
GeoNode Documentation

Versão 3.2.1

GeoNode Development Team

29 jul., 2021

1	Índice	3
1.1	O que é GeoNode	3
1.1.1	Galeria	4
1.1.2	Links mais úteis	4
1.2	Licenciamento	4
1.3	Versão Actual e Recursos	5
1.4	Entre em contacto com a comunidade	5
1.5	Roteiro	5
1.6	Fundamentos do GeoNode	5
1.6.1	<i>Com o GeoNode, utilizadores não especializados podem partilhar dados e criar mapas interactivos.</i>	7
1.6.2	Armazenamento de dados geoespaciais	7
1.6.3	Mistura de dados, criação de mapas	9
1.6.4	GeoNode como um bloco de construção	9
1.6.5	Convencido! Onde eu assino?	10
1.7	Navegadores Suportados	10
1.7.1	Internet Explorer	10
1.7.2	A testar no Internet Explorer	10
1.8	Demonstração Online	13
1.9	Guia de Instalação Rápido	13
1.9.1	Guia de Instalação Rápido	13
1.10	Manual de Utilizador do GeoNode	17
1.10.1	Contas e Perfil de Utilizador	17
1.10.2	Interagindo com Utilizadores e Grupos	28
1.10.3	Dados	36
1.10.4	Fazendo a Gestão de Documentos	47
1.10.5	Fazendo a Gestão de Camadas	69
1.10.6	Gestão de Mapas	116
1.10.7	Publicação de Dados	170
1.10.8	Usando o GeoNode com Outras Aplicações	170
1.11	GeoNode Basic Installation	190
1.11.1	Overview	190
1.11.2	First Step: Deploy GeoNode on a local server (e.g.: http://localhost/)	190
1.11.3	Second Step: Deploy GeoNode on a production server (e.g.: https://my_geonode.geonode.org/)	193
1.11.4	Third Step: Customize <i>.env</i> to match your needs	198

1.11.5	Fourth Step: Secure your production deployment; change the <i>admin</i> passwords and <i>OAuth2</i> keys	198
1.11.6	Further Production Enhancements	204
1.12	GeoNode Advanced Installation	213
1.12.1	GeoNode Core	213
1.12.2	GeoNode Project	265
1.13	GeoNode Settings	273
1.13.1	Settings	273
1.14	Customize the Look and Feel	318
1.14.1	GeoNode Themes	318
1.14.2	Theming your GeoNode Project	319
1.15	GeoNode permissions	326
1.15.1	Permissions	326
1.16	Read-Only and Maintenance Mode	337
1.16.1	Read-Only and Maintenance Modes	337
1.17	Monitoring	341
1.17.1	Monitoring	341
1.17.2	Monitoring: API	360
1.17.3	Monitoring: User Analytics	381
1.17.4	Monitoring: Notifications	393
1.17.5	Web API	394
1.18	GeoNode Backup and Restore	402
1.18.1	Full GeoNode Backup & Restore	402
1.19	Viewer and Hooksets	431
1.19.1	GXP	431
1.19.2	MapStore 2	431
1.19.3	Leaflet	431
1.20	GeoNode Components and Architecture	431
1.20.1	Overview	431
1.20.2	Django	432
1.20.3	WebServers	432
1.20.4	GeoServer	432
1.20.5	Databases	432
1.20.6	OAuth2 Security: Authentication and Authorization	432
1.21	Hardening GeoNode	482
1.21.1	Publish on HTTPS	482
1.21.2	OAuth2 Fixtures Update and Base URL Migration	482
1.21.3	GeoNode Security Subsystem	482
1.21.4	OAuth2 Tokens and Sessions	483
1.22	Social Login	483
1.22.1	GeoNode Social Accounts	483
1.23	GeoNode Django Contrib Apps	503
1.23.1	Geonode auth via LDAP	503
1.23.2	Geonode Logstash for centralized monitoring/analytics	508
1.24	GeoNode Admins Guide	515
1.24.1	Accessing the panel	515
1.24.2	Reset or Change the admin password	516
1.24.3	Simple Theming	517
1.24.4	Add a new user	532
1.24.5	Activate/Disable a User	535
1.24.6	Change a User password	536
1.24.7	Promoting a User to Staff member or superuser	536
1.24.8	Creating a Group	538
1.24.9	Managing a Group	543

1.24.10	Group based advanced data workflow	554
1.24.11	Manage profiles using the admin panel	557
1.24.12	Manage layers using the admin panel	557
1.24.13	Manage the maps using the admin panel	559
1.24.14	Manage the documents using the admin panel	561
1.24.15	Manage the base metadata choices using the admin panel	562
1.24.16	Announcements	571
1.24.17	Menus, Items and Placeholders	574
1.24.18	OAuth2 Access Tokens	582
1.25	GeoNode Management Commands	587
1.25.1	Migrate GeoNode Base URL	587
1.25.2	Update Permissions, Metadata, Legends and Download Links	589
1.25.3	Loading Data into GeoNode	594
1.25.4	Create Users and Super Users	623
1.25.5	Batch Sync Permissions	626
1.25.6	Delete Certain GeoNode Resources	629
1.26	Changing the default Languages	631
1.26.1	Changing the Default Language	631
1.26.2	GeoNode Configuration	631
1.26.3	Additional Steps	631
1.26.4	Restart	632
1.27	GeoNode Upgrade from older versions	632
1.27.1	Upgrade from 3.1.x	632
1.27.2	Upgrade from 2.10.x / 3.0	633
1.27.3	Upgrade from 2.4.x	634
1.28	GeoNode Async Signals	636
1.28.1	Supervisord and Systemd	636
1.28.2	Celery	636
1.28.3	Rabbitmq and Redis	636
1.28.4	How to: Async Upload via API	636
1.29	GeoNode add a thesaurus	637
1.29.1	Loading a thesaurus	637
1.29.2	Admin panel	638
1.29.3	Command line	638
1.29.4	Configure a thesaurus in GeoNode	639
1.29.5	Apply a thesaurus to a resource	641
1.30	Participate in the Discussion	641
1.30.1	Join the community, ask for help or report bugs	641
1.31	Write Documentation	642
1.31.1	How to contribute to GeoNode's Documentation	642
1.32	Provide Translations	644
1.32.1	Contribute to Translations	644
1.33	Write Code	650
1.34	Frontend Development	650
1.34.1	Frontend development	650
1.35	API Schema	652
1.35.1	GeoNode API Schema	652
1.36	How to Develop	683
1.36.1	Start to develop with Docker	683
1.36.2	How to Install GeoNode-Core for development	684
1.36.3	How to run GeoNode Core for development	693
1.36.4	How to run GeoNode Project for development	693
1.36.5	Start MapStore2 client in development mode	694
1.36.6	Workshops	695

Bem-vindo à documentação do GeoNode.

GeoNode é um sistema de gestão de conteúdo (CMS) de código aberto para dados geoespaciais. É uma aplicação e plataforma baseada na web para o desenvolvimento de sistemas de informação geoespaciais (SIG) e para implantação de infraestruturas de dados espaciais (SDI).

1.1 O que é GeoNode



GeoNode é um sistema de gestão de conteúdo geoespacial , uma plataforma para a gestão e publicação de dados geoespaciais. Reúne projectos de software de código aberto maduros e estáveis em uma interface consistente e fácil de usar, permitindo aos utilizadores não especializados partilhar dados e criar mapas interactivos.

As ferramentas de gestão de dados incorporadas no GeoNode permitem a criação integrada de dados, metadados e visualização de mapas. Cada conjunto de dados no sistema pode ser partilhado publicamente ou restringido para permitir o acesso a apenas utilizadores específicos. Funcionalidades sociais como perfis de utilizador e sistemas de comentários e classificação permitem o desenvolvimento de comunidades em cada plataforma para facilitar o uso, gestão e controlo de qualidade dos dados que a instância do GeoNode contém.

É também projetado para ser uma plataforma flexível, que os desenvolvedores de software podem estender, modificar ou integrar para atender aos requisitos em seus próprios aplicativos.

1.1.1 Galeria

Vários outros projectos de código aberto estendem a funcionalidade do GeoNode, aproveitando a capacidade de reutilização de aplicativos Django. Visite a nossa galeria para ver como a comunidade usa o GeoNode: [Galeria GeoNode](#).

A comunidade de desenvolvimento dá muito apoio a novos projectos e contribui com ideias e orientações para os recém-chegados.

Para uma demonstração em tempo real, veja também [Demonstração Online](#)

1.1.2 Links mais úteis

Geral

- Página inicial do projecto: <https://geonode.org>
- Repositório: <https://github.com/GeoNode/geonode>
- Demonstração oficial: <http://master.demo.geonode.org>
- Wiki do GeoNode: <https://github.com/GeoNode/geonode/wiki>
- Centro de suporte: <https://github.com/GeoNode/geonode-project/issues>

No caso de bugs sensíveis como vulnerabilidades de segurança, por favor, entre em contacto com um Desenvolvedor Principal do GeoNode directamente, ao invés de usar o centro de suporte. Valorizamos seu esforço para melhorar a segurança e a privacidade deste projecto!

Projectos relacionados

- Projecto GeoNode: <https://github.com/GeoNode/geonode-project>
- GeoNode no Docker: <https://hub.docker.com/u/geonode>
- OSGeo-Live GeoNode: <https://live.osgeo.org/en/>

1.2 Licenciamento

GeoNode é Copyright 2018 Open Source Geospatial Foundation (OSGeo).

GeoNode é um software livre: você pode redistribuí-lo e/ou modificá-lo sob os termos da GNU General Public License (Licença Pública Geral GNU) conforme publicado pela Free Software Foundation, seja a versão 3 da Licença, ou (por sua opção) qualquer versão posterior. O GeoNode é distribuído na esperança de que seja útil, mas SEM NENHUMA GARANTIA; sem mesmo a garantia implícita de COMERCIALIDADE ou ADEQUABILIDADE A UM DETERMINADO FIM. Consulte a Licença Pública Geral GNU para obter mais detalhes.

Você deveria ter recebido uma cópia da Licença Pública Geral GNU junto com o GeoNode. Caso contrário, consulte <http://www.gnu.org/licenses>.

1.3 Versão Actual e Recursos

Versão actual do GeoNode: 3.2.0

Principais Recursos: [Estado do GeoNode](#)

1.4 Entre em contacto com a comunidade

GeoNode é um projecto de código aberto e colaboradores são necessários para manter este projecto em andamento. Saiba mais sobre como contribuir, no nosso [Estatuto da Comunidade](#).

- Mailing list de Utilizadores: <https://lists.osgeo.org/cgi-bin/mailman/listinfo/geonode-users>
- Mailing list de Desenvolvedores: <https://lists.osgeo.org/cgi-bin/mailman/listinfo/geonode-devel>
- Chat do Gitter: <https://gitter.im/GeoNode/general>

1.5 Roteiro

O roteiro de desenvolvimento do GeoNode está documentado em uma série de Projectos de Melhoria GeoNode (GNIPS). Eles estão documentados no [Wiki do GeoNode](#).

GNIPS são considerados grandes empreendimentos que irão adicionar uma grande quantidade de recursos ao projeto. Como tal, são o tópico de discussão e orientação da comunidade.

A comunidade discute isto na mailing list de desenvolvedores: <http://lists.osgeo.org/pipermail/geonode-devel/>

1.6 Fundamentos do GeoNode



é uma plataforma para a gestão e publicação de dados geoespaciais. Reúne projectos maduros de software de código aberto em uma interface fácil de usar.

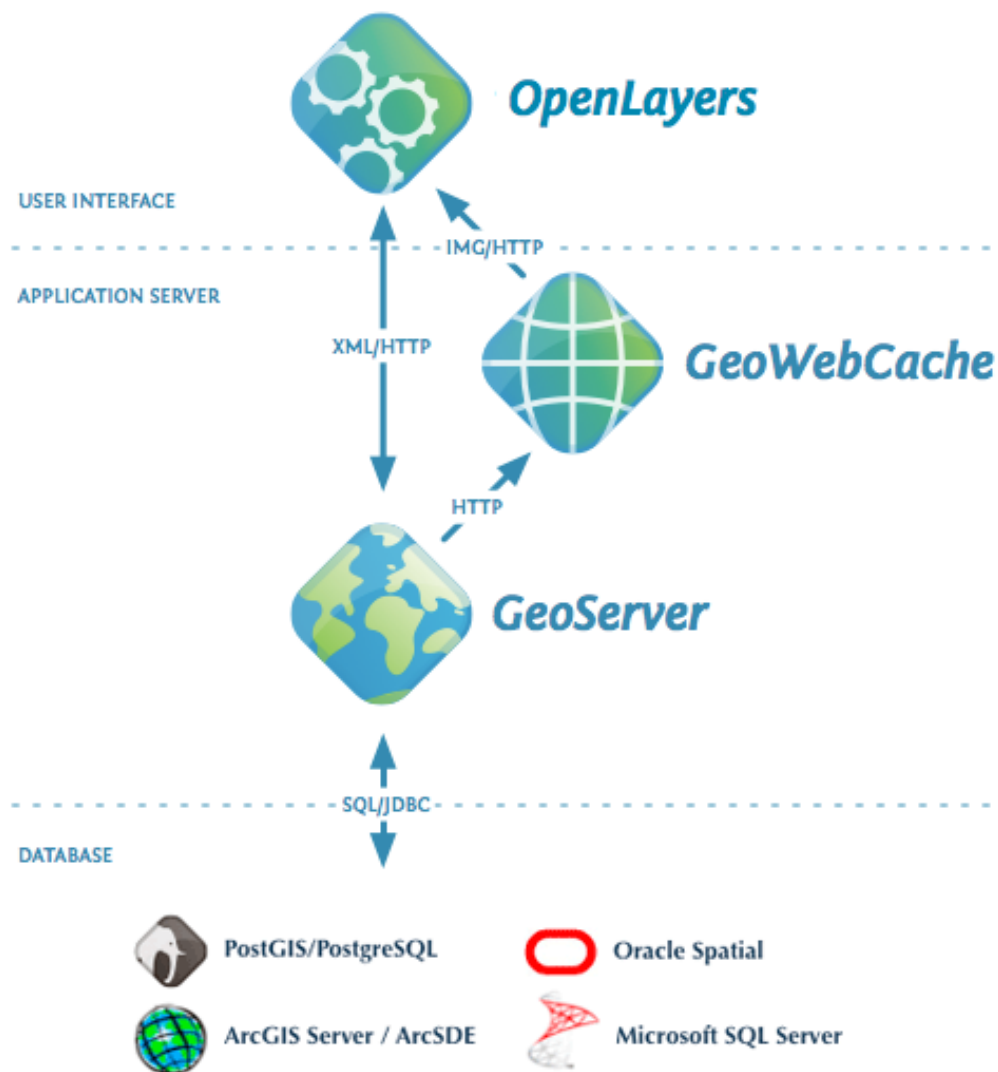
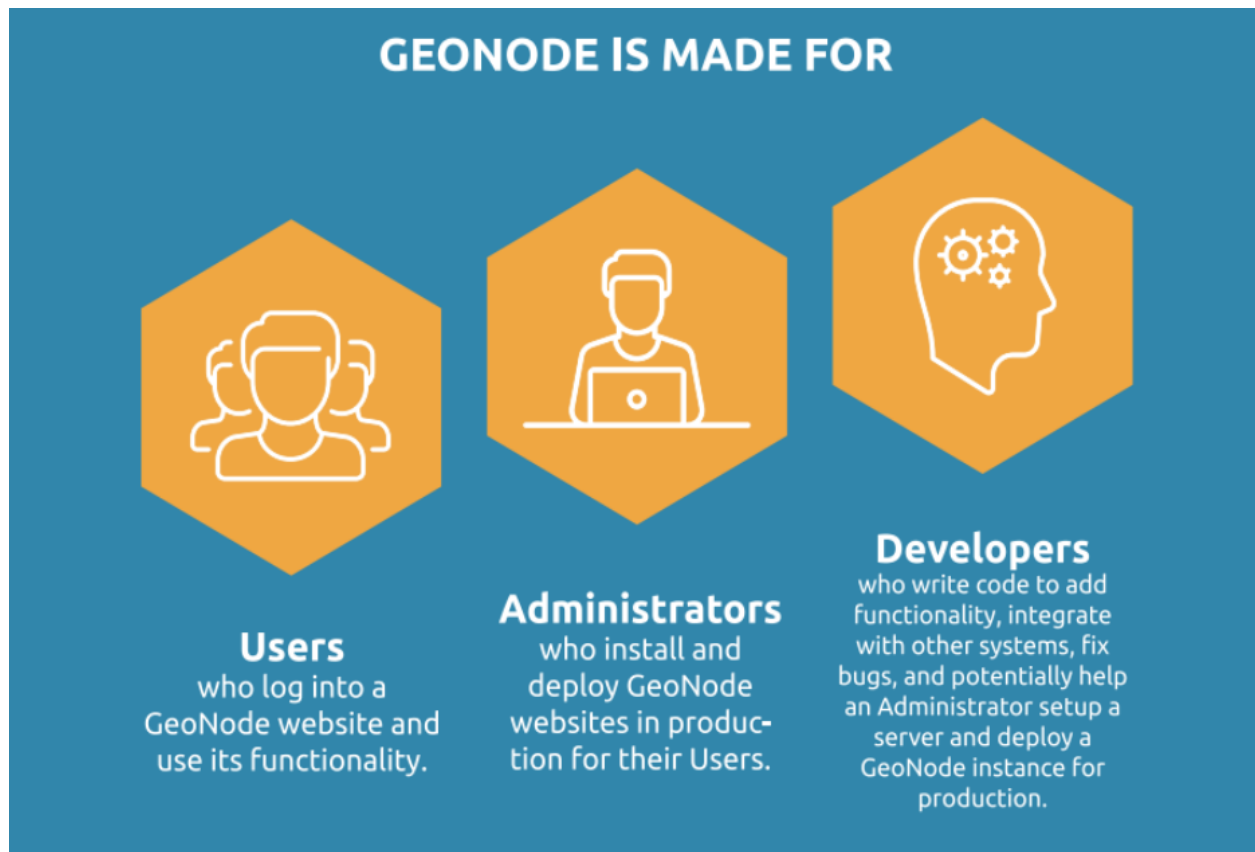


Fig. 1: *GeoNode simplified architecture*

1.6.1 Com o GeoNode, utilizadores não especializados podem partilhar dados e criar mapas interactivos.



1.6.2 Armazenamento de dados geoespaciais

GeoNode permite aos utilizadores fazer upload de dados vectoriais (actualmente shapefiles, json, csv, kml e kmz) e dados matriciais (raster) nas suas projecções originais usando um formulário web.

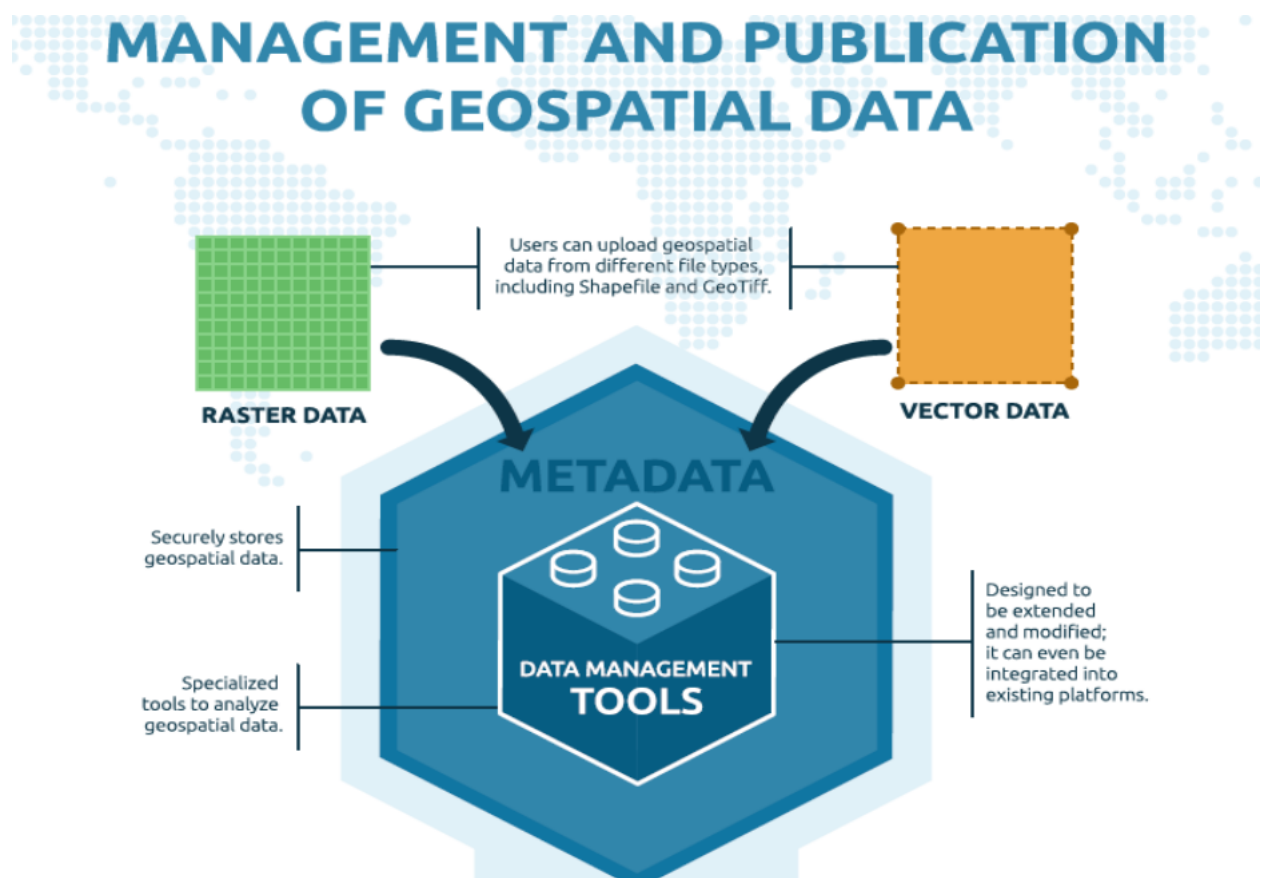
Os dados vectoriais são convertidos em tabelas geoespaciais em um banco de dados, imagens de satélite e outros tipos de dados raster são guardados como GeoTIFFs.

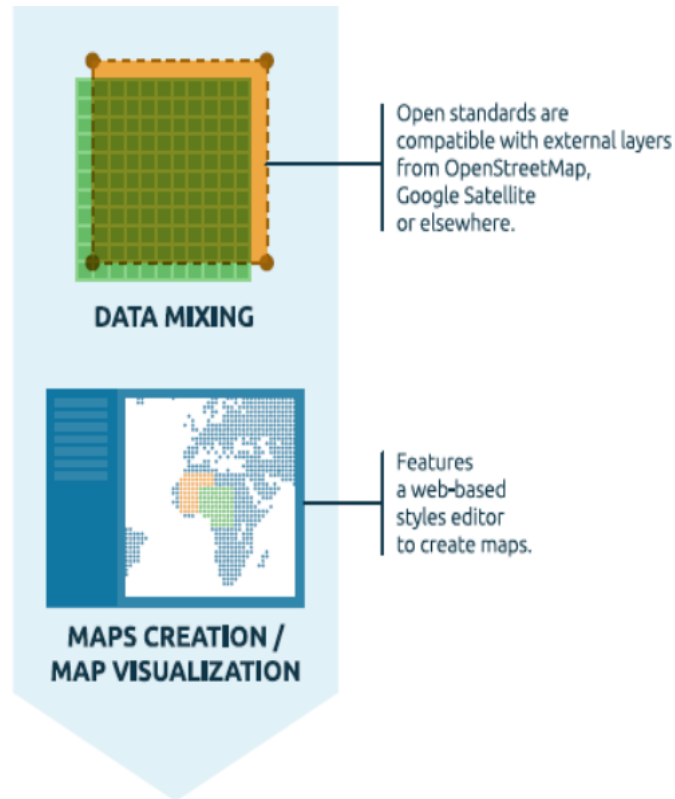
É dada importância especial aos formatos padrão de metadados, como os padrões de metadados ISO 19139: 2007 / ISO 19115.

Assim que o upload for concluído, o utilizador pode preencher os metadados do recurso para torná-lo repentinamente disponível através dos terminais [CSW](#) (Serviço de Catálogo OGC) e APIs.

Os utilizadores podem também, fazer upload de um documento XML de metadados (formato ISO, FGDC e Dublin Core) para preencher os principais elementos de metadados do GeoNode automaticamente.

Da mesma forma, o GeoNode fornece um estilizador baseado na web que permite aos utilizadores alterar as representações dos dados e visualizar as alterações em tempo real.





1.6.3 Mistura de dados, criação de mapas

Uma vez que os dados tenham sido carregados, o GeoNode permite ao utilizador pesquisá-los geograficamente ou por meio de palavras-chave para criar mapas sofisticados.

Todas as camadas são automaticamente reprojectadas para Web Mercator para exibição de mapas, tornando possível o uso de diferentes camadas de base populares, como Open Street Map, Google Satellite ou Bing layers.

Depois de salvar os mapas, é possível inseri-los em qualquer página da web ou obter uma versão em PDF para impressão.

1.6.4 GeoNode como um bloco de construção

Vários outros projectos de código aberto estendem a funcionalidade do GeoNode, aproveitando a reusabilidade de aplicativos Django.

Visite a nossa galeria para ver como a comunidade usa o GeoNode: [Projectos GeoNode](#).

A comunidade de desenvolvimento apoia muito a novos projectos e contribui com ideias e orientação para os recém-chegados.

1.6.5 Convencido! Onde eu assino?

As próximas etapas são:

1. Faça um passeio pela *Demonstração Online*
2. Siga o *Guia de Instalação Rápido* de modo a brincar com a sua própria instância local e aceder a todas as funcionalidades de administração
3. Leia a documentação iniciando pelo *guia do utilizador* ao *guia do administrador*
4. Inscreva-se nas listas de discussão [utilizadores-geonode](#) e/ou [desenvolvimento-geonode](#) para se juntar à comunidade. Veja também a secção [entre em contacto](#) para mais informações.

Obrigado pelo seu interesse!

1.7 Navegadores Suportados

GeoNode é conhecido por funcionar em todos os navegadores modernos.

Esta lista inclui (mas não está limitada a):

- [Google Chrome](#).
- [Apple Safari](#).
- [Mozilla Firefox](#).
- [Microsoft Edge](#).
- [Microsoft Internet Explorer](#).

Nota: A grande maioria dos desenvolvedores GeoNode prefere usar o Google Chrome.

1.7.1 Internet Explorer

As versões do Microsoft Internet Explorer anteriores a 10, apresentam problemas conhecidos quando usados para navegar em um site GeoNode. Assim, uma mensagem é exibida avisando ao utilizador que ele deve actualizar o seu navegador.

1.7.2 A testar no Internet Explorer

Ao trabalhar no código de front-end, os desenvolvedores devem ter o cuidado de testar cuidadosamente com o Microsoft Internet Explorer, para garantir que os recursos em que estão a trabalhar realmente funcionem correctamente neste navegador. É uma boa prática testar em todos os navegadores disponíveis, mas o uso de bibliotecas modernas de front-end, como bootstrap e jQuery, torna muito mais provável que o código funcione perfeitamente em todos os navegadores.

Para testar no Internet Explorer, os desenvolvedores podem usar o site [Modern IE](#) para baixar máquinas virtuais para uso no [Oracle VM Virtual Box](#).

Depois de fazer o download da VM, siga as instruções para configurá-la na sua instalação do VirtualBox.

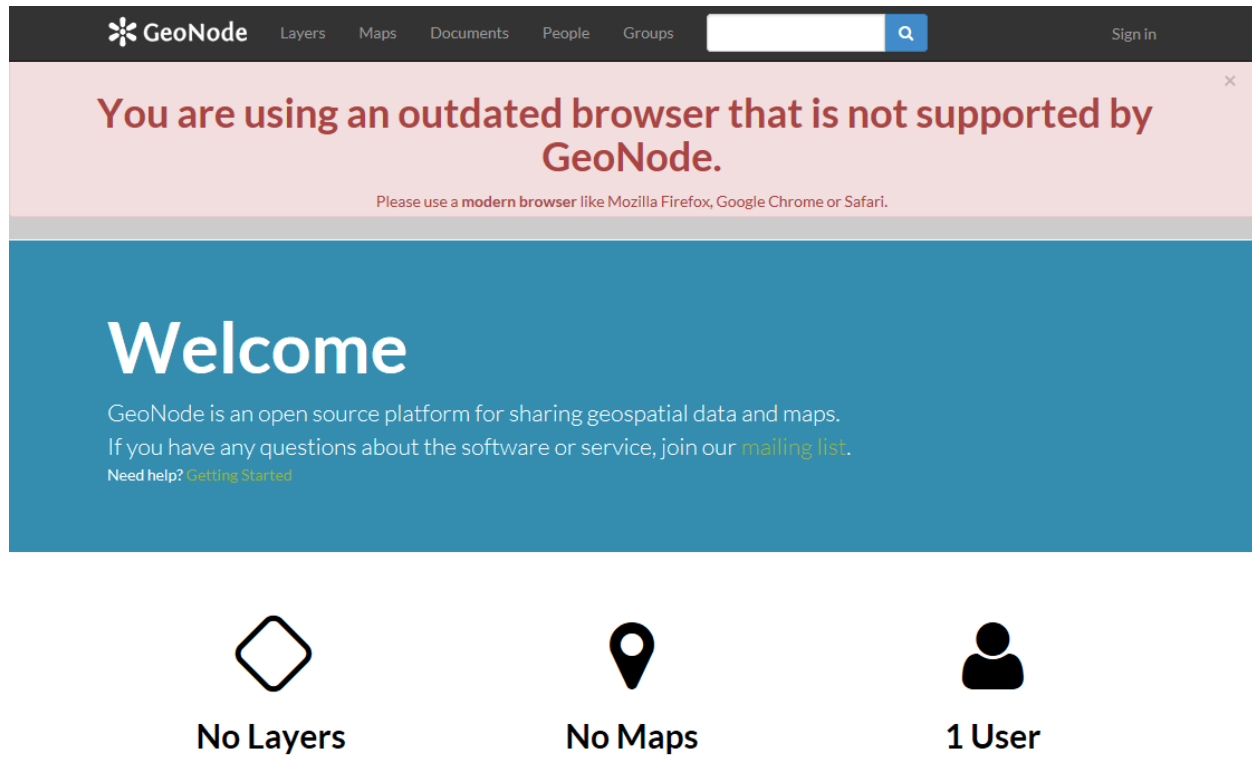


Fig. 2: Mensagem de erro do Internet Explorer

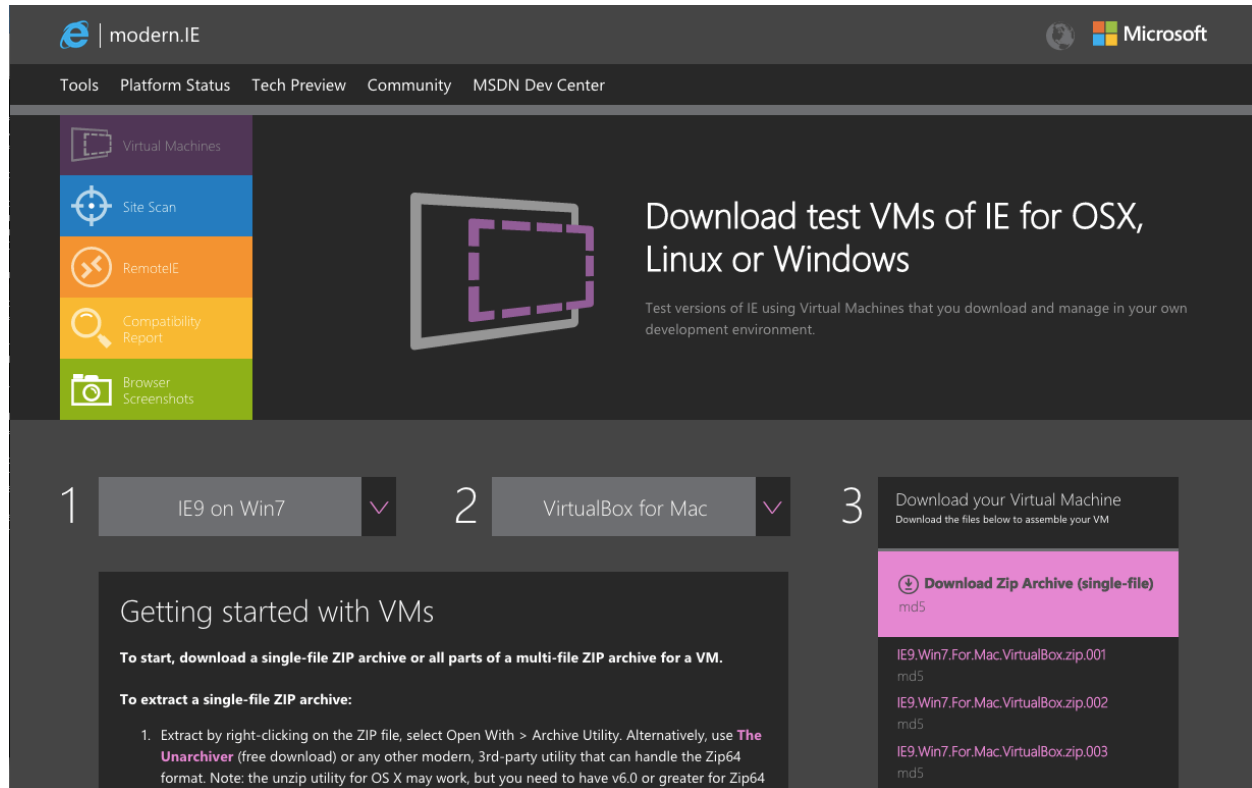


Fig. 3: A testar no Internet Explorer

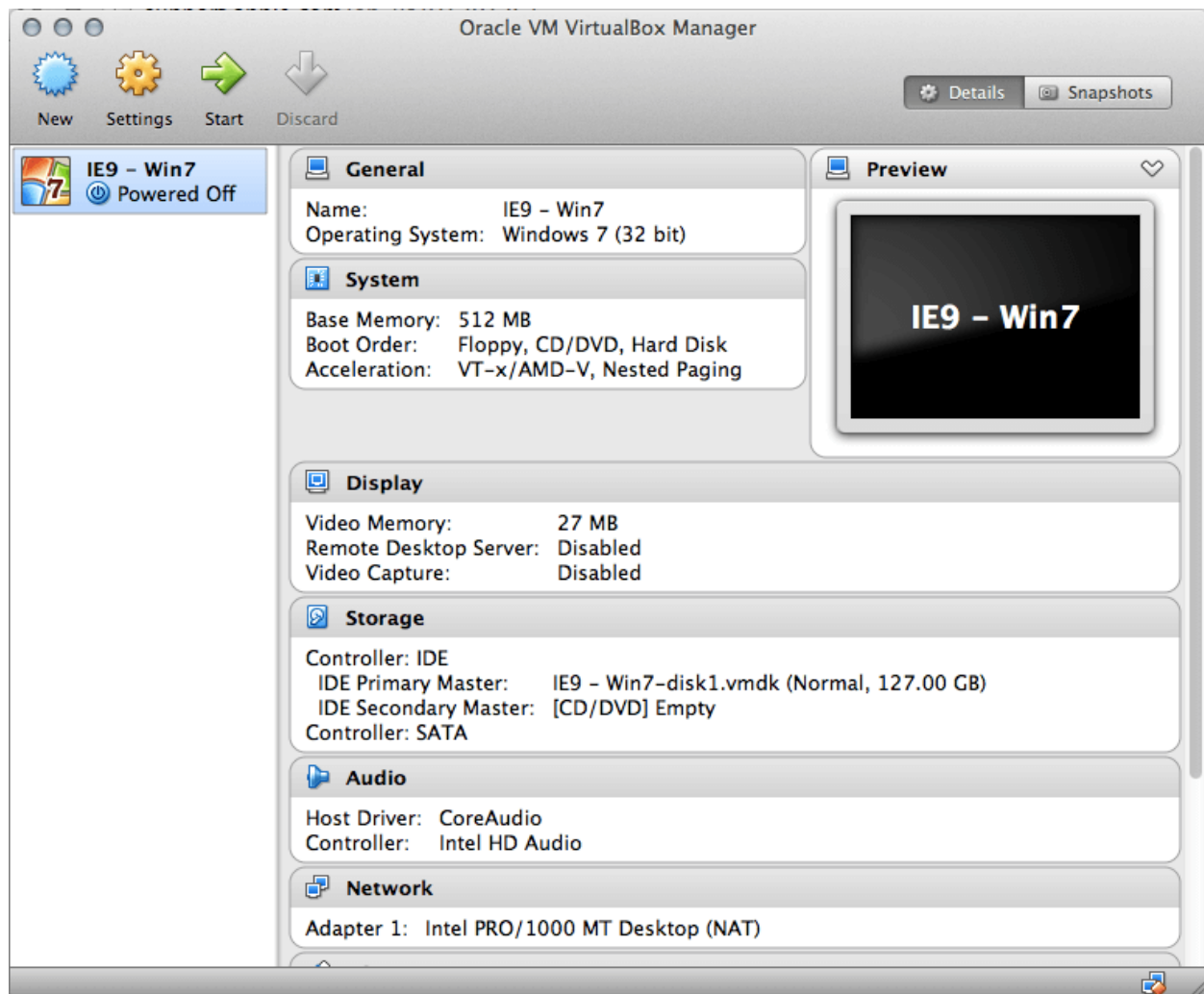


Fig. 4: Interface de administração do Oracle VirtualBox

Depois que a VM for configurada, pode aceder à sua instância de desenvolvimento do GeoNode, visitando o endereço IP de sua máquina anfitriã ou na interface no modo bridge (geralmente 10.0.2.2) e começar os seus testes.

1.8 Demonstração Online

Nota: Isenção de responsabilidade não fornecemos garantia por quaisquer dados publicados neste Site de Demonstração. Publique os dados por sua própria conta e risco. Cada conjunto de dados será removido automaticamente todos os domingos. Se encontrar algum conjunto de dados que não deveria lá estar, escreva imediatamente para a comunidade que desenvolve e mantém o GeoNode.

Veja a secção *Entre em contacto com a comunidade* para detalhes.

Uma demonstração em tempo real da versão estável mais recente está disponível em <http://master.demo.geonode.org/>.

Qualquer pessoa pode se inscrever para uma conta de utilizador, fazer upload e modelar dados, criar e partilhar mapas e alterar permissões.

Por ser um site de demonstração, aos domingos, todos os conjuntos de dados serão apagados. Utilizadores, senhas e grupos serão preservados.

Isto deve permitir que faça uma incursão pelos principais recursos do GeoNode de forma rápida e fácil.

Aviso: Esta instância do GeoNode está configurada com definições padrão e um nível de segurança muito baixo. Esta é uma demonstração apenas, não deve ser considerado um sistema realmente pronto para a produção. Para uma lista completa de configurações, consulte a secção: *Settings*

1.9 Guia de Instalação Rápido

1.9.1 Guia de Instalação Rápido

Introdução

A seguir, um guia rápido para começar a usar o GeoNode na maioria dos sistemas operativos comuns.

Nota: Para uma configuração completa e implantação, consulte os *guias de instalação completos*

Isto deve ser executado em uma máquina nova, sem pacotes ou versões do GeoNode previamente instalados.

Aviso: Os métodos apresentados aqui, devem ser usados apenas para uma demonstração interna limitada. Antes de expor sua instância GeoNode a um servidor público, leia com atenção o *guia de protecção*

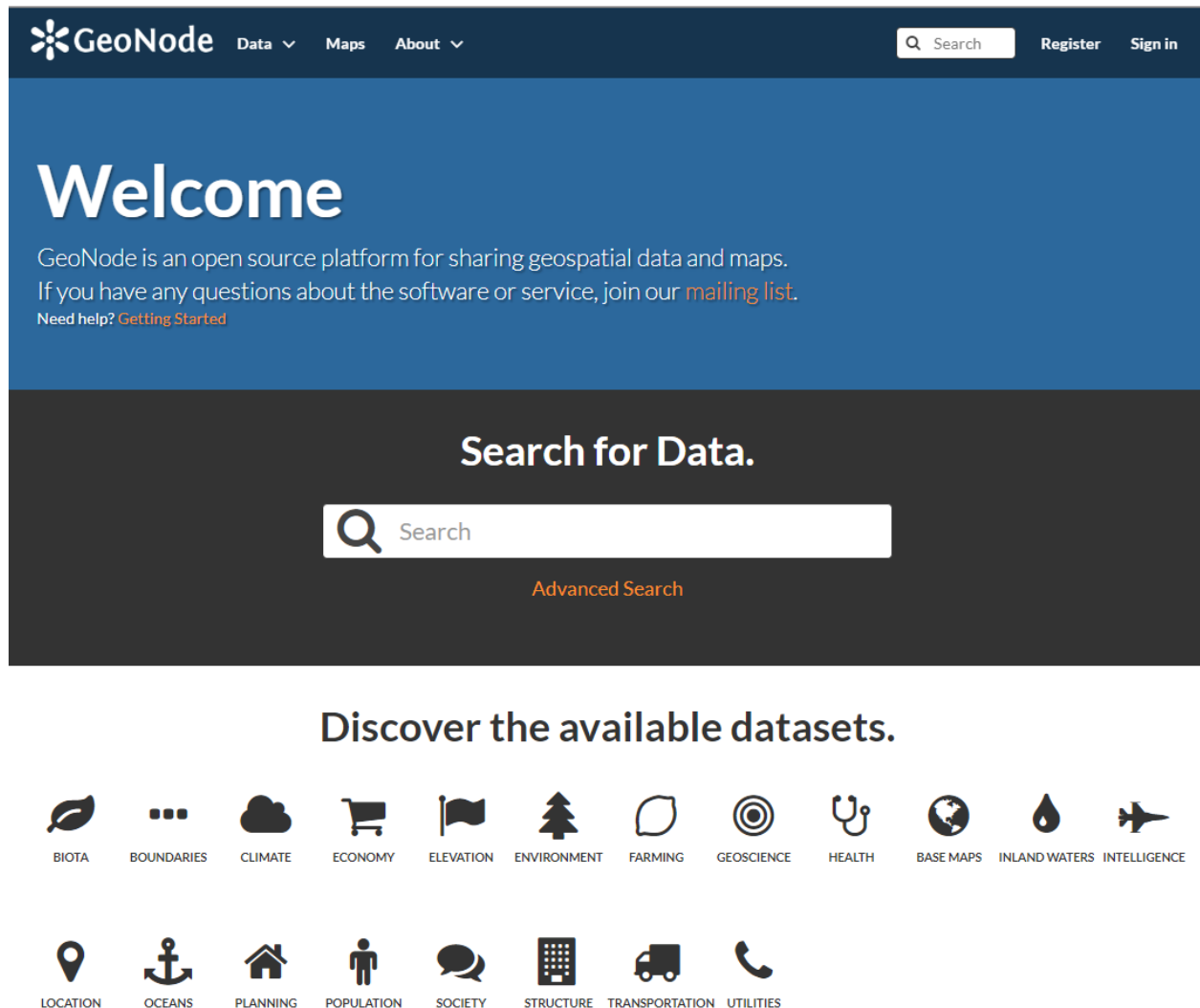


Fig. 5: *Demonstração Online @ master.demo.geonode.org*

Requisitos Mínimos Recomendados do Sistema

Uma especificação definitiva de requisitos técnicos é difícil de recomendar. O desempenho aceite é altamente subjetivo. Além disso, o desempenho depende de factores como utilizadores em simultâneo, registos na base de dados ou a conectividade de rede da sua infraestrutura.

Para a implantação do GeoNode num único servidor, os seguintes são os requisitos *mínimos* do sistema:

- 8GB de RAM (16GB ou mais de preferência para uma implantação de produção).
- Processador de 2,2GHz com 4 núcleos. (Poder de processamento adicional pode ser necessário para várias renderizações de estilo simultâneas)
- 30 GB de capacidade de disco ocupado pelo software (Reservado apenas ao Sistema Operativo e ao código-fonte).
- Espaço adicional em disco para quaisquer dados hospedados com o GeoNode, dados armazenados na Base de Dados e blocos armazenados em cache com GeoWebCache. Para a bd, dados espaciais, blocos em cache e «espaço temporário» úteis para administração, um tamanho de base decente para implantações GeoNode está entre 50 GB e 100 GB.
- Hardware de 64 bits **fortemente** recomendado.

CD OSGEO Executável



OSGeoLive é um DVD independente auto-inicializável, pen drive USB ou máquina virtual baseada em Ubuntu, que lhe permite experimentar uma grande variedade de software geoespacial de código aberto sem instalar nada.

É composto inteiramente de software livre, permitindo que seja livremente distribuído e duplicado.

Ele fornece aplicativos pré-configurados para uma variedade de casos de uso geoespaciais, incluindo armazenamento, publicação, visualização, análise e manipulação de dados.

Ele também contém conjuntos de dados de amostra e documentação.

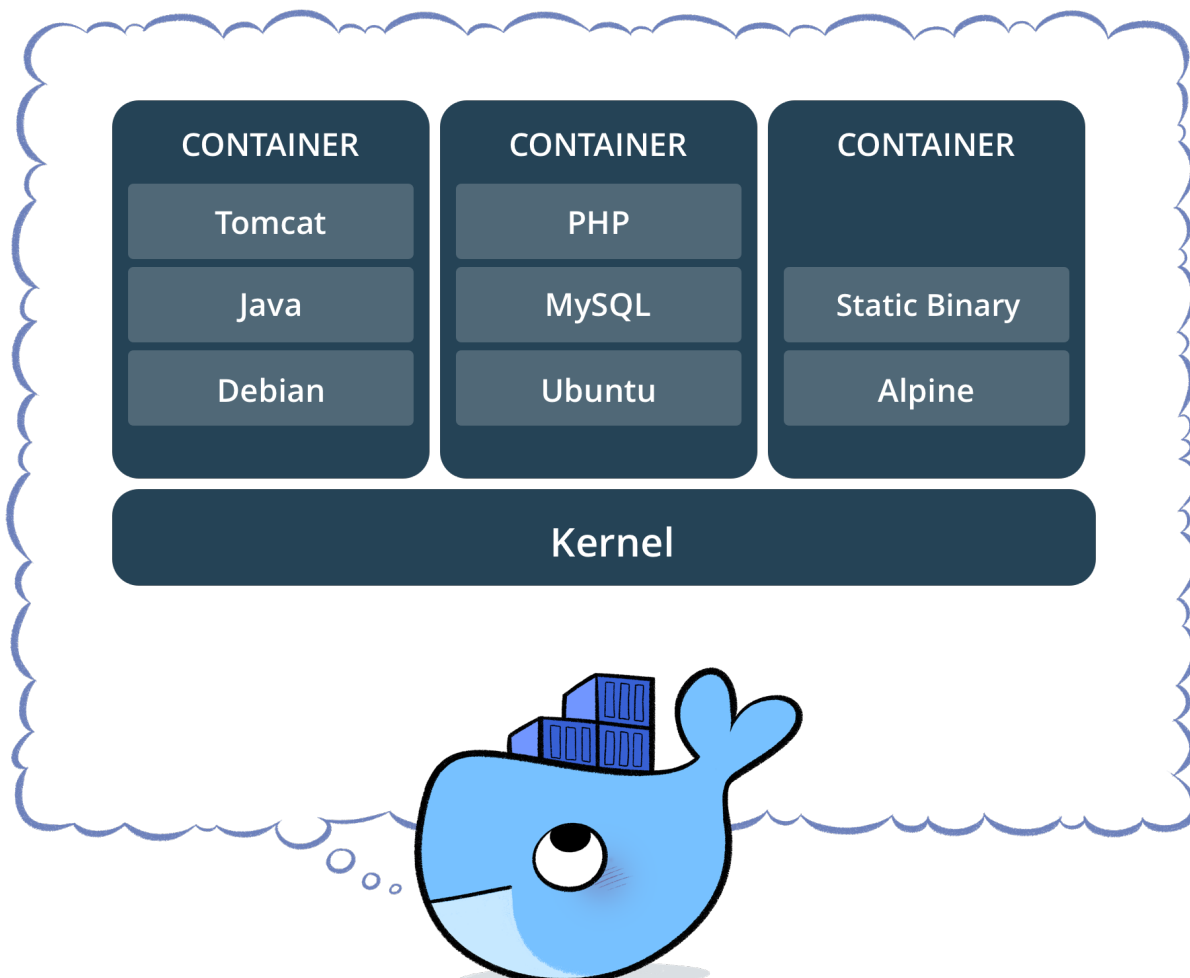
Para experimentar os aplicativos, basta:

- Inserir o DVD ou pen drive USB no computador ou máquina virtual.
- Reinicialize o computador. (verifique a ordem do dispositivo de inicialização, se necessário)
- Pressione *Enter* para iniciar e fazer o login.
- Selecione e execute aplicativos a partir do menu *Geoespacial*.

OSGeoLive is an [OSGeo Foundation](#) project. The [OSGeo Foundation](#) is a not-for-profit supporting Geospatial Open Source Software development, promotion and [education](#).

Instalar via Docker

Docker é uma plataforma de software livre usado para empacotar software em unidades padronizadas para desenvolvimento, remessa e implantação.



Nota: créditos para Docker

Apresentando os conceitos principais

Uma imagem de container é um pacote leve, autônomo e executável de um software que inclui tudo o que é necessário para executá-lo: código, bibliotecas de execução, ferramentas do sistema, bibliotecas do sistema, configurações.

Os containers do Docker em execução numa única máquina compartilham o kernel do sistema operativo dessa máquina; eles iniciam instantaneamente e usam menos computação e RAM.

Os containers podem compartilhar um único kernel e as únicas informações que precisam estar em uma imagem do container são o executável e suas dependências de pacote, que nunca precisam ser instalados no sistema anfitrião.

Vários containers podem ser executados na mesma máquina e compartilhar o kernel do sistema operativo com outros containers, cada um a executar como processos isolados no espaço do utilizador.

Os seguintes tutoriais irão apresentar o uso do Docker community edition em:

- *Ubuntu 18.04*
- *CentOS 7.0*

1.10 Manual de Utilizador do GeoNode

1.10.1 Contas e Perfil de Utilizador

No GeoNode, muitos conteúdos são públicos pelo que utilizadores não registados têm acesso de leitura a mapas, camadas e documentos públicos. Para criar mapas, adicionar camadas ou documentos, editar os dados e partilhar esses recursos com outros utilizadores, é necessário fazer login.

GeoNode é principalmente uma plataforma *social*, portanto, um componente principal de qualquer instância GeoNode é a conta de utilizador.

Esta secção vai guiá-lo através do registo da conta, actualização das informações e preferências da sua conta, ligações com redes sociais e endereços de email.

Criar uma Nova Conta

Para tirar o máximo proveito de todos os recursos do GeoNode, precisa de uma conta de utilizador. Siga estes passos para criar uma nova conta.

1. Em qualquer página da interface da web verá um link *Register*. Clique nesse link e o formulário de registo irá aparecer

Nota: Os registos no GeoNode devem ser abertos. Caso não veja o link de registo, então não é possível o registo a menos que o administrador do site o faça para si.

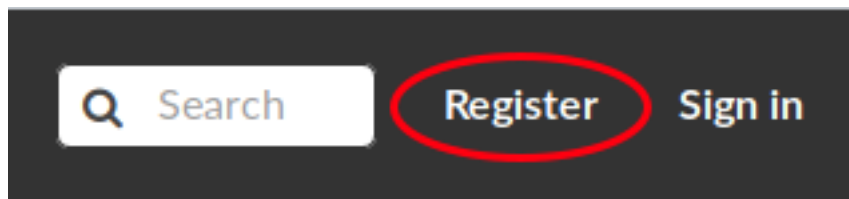
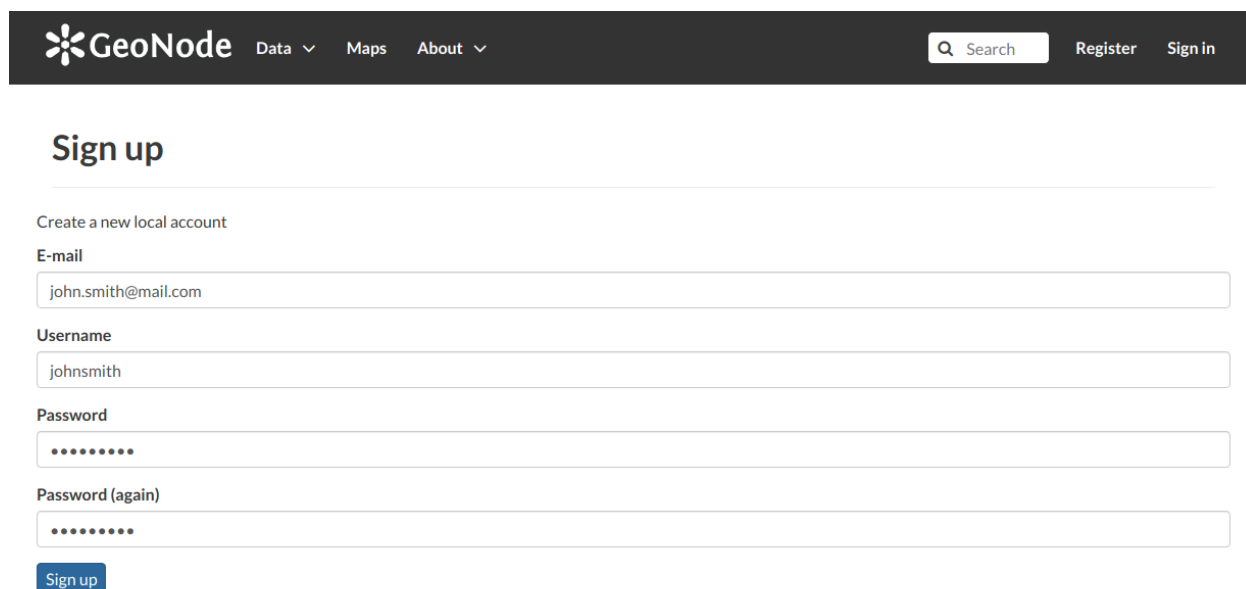


Fig. 6: *Ecrã de login*

2. Na próxima página, preencha o formulário. Insira um nome de utilizador e senha nos campos. Além disso, insira seu endereço de email para verificação.
3. Você será automaticamente conectado e redirecionado para a página de Perfil. Um email será enviado a confirmar a sua inscrição. Se nenhum erro ocorrer durante o registo, os seguintes alertas irão aparecer no ecrã:

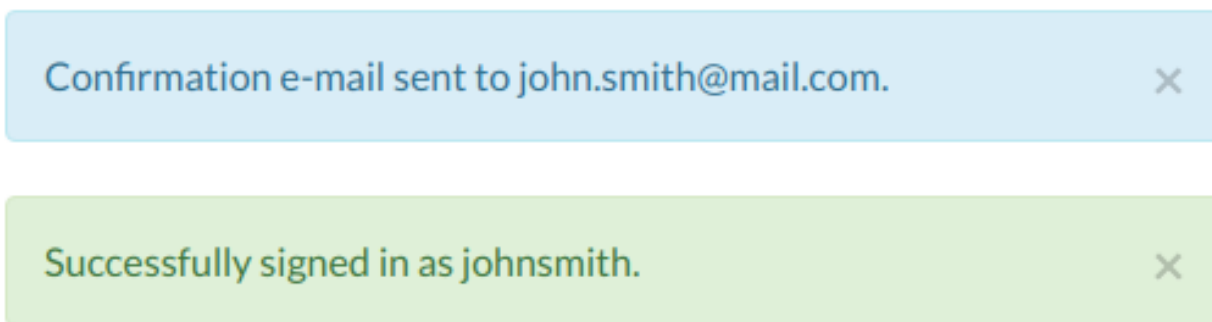
Para sair, clique na ligação *Log out* do menu do utilizador.

Deve confirmar esta acção conforme descrito na imagem abaixo.



The image shows the 'Sign up' page of the GeoNode application. At the top is a dark navigation bar with the GeoNode logo, links for 'Data', 'Maps', and 'About', a search bar, and 'Register' and 'Sign in' buttons. The main heading is 'Sign up'. Below it is the instruction 'Create a new local account'. The form contains four input fields: 'E-mail' (with the value 'john.smith@mail.com'), 'Username' (with the value 'johnsmith'), 'Password' (masked with dots), and 'Password (again)' (also masked with dots). A blue 'Sign up' button is located at the bottom left of the form area.

Fig. 7: *Registando-se para uma nova conta*



The image displays two alert messages. The first is a light blue box with the text 'Confirmation e-mail sent to john.smith@mail.com.' and a close button (X) on the right. The second is a light green box with the text 'Successfully signed in as johnsmith.' and a close button (X) on the right.

Fig. 8: *Alertas*

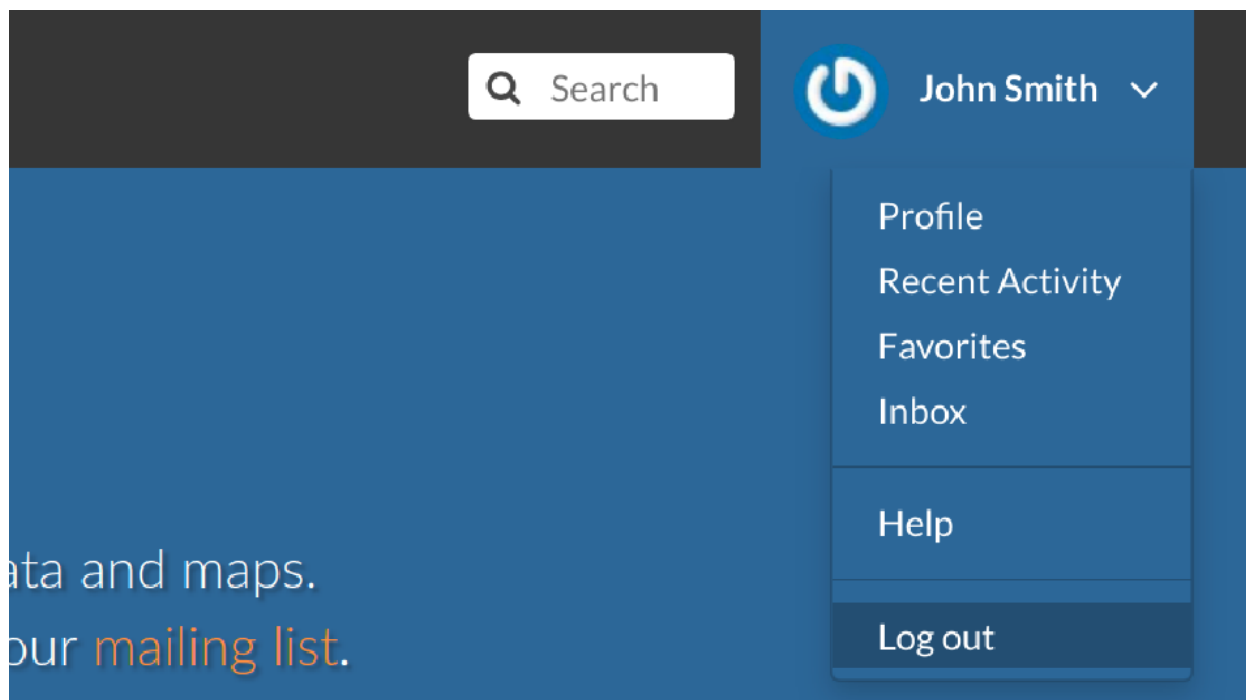


Fig. 9: *Ligação de logout*



Fig. 10: *Confirmar Log out*

Actualização do Perfil

Depois de ter uma conta, pode completar o seu perfil com informações úteis. Pode também editar ou excluir as existentes. Pode conectar a conta à sua rede social, associar muitos endereços de e-mail a ela e fazer a gestão de várias opções, como preferências sobre notificações.

Pode actualizar essas informações a qualquer momento na página de *Perfil*, que pode ser acedida a partir do menu do utilizador.

Então, clique em seu nome de utilizador no canto superior direito do ecrã. Uma lista suspensa será exibida. Clique em *Profile* para entrar na página de configurações do *Perfil*.

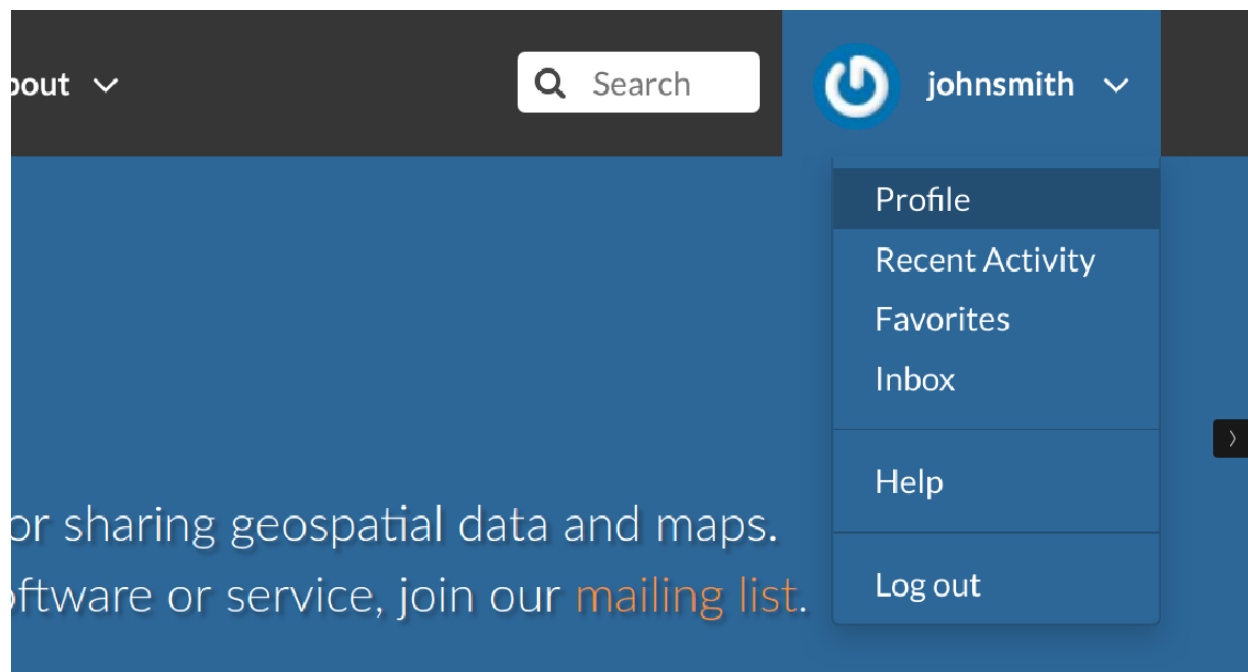


Fig. 11: Link para o seu perfil

A página de *Perfil* é similar à imagem em baixo.

As suas informações pessoais são mostradas sob o nome de utilizador. Na parte inferior da página estão listados todos os recursos associados ao seu *Perfil*. Pode decidir visualizar apenas camadas ou mapas ou documentos clicando na aba correspondente.

Através do link `User layers WMS GetCapabilities document` é possível recuperar um documento XML com a lista das camadas disponíveis.

No lado direito da página, há muitos links úteis para editar informações pessoais, fazer upload e criar camadas ou mapas, actualizar as suas configurações de *Perfil* e entrar em contacto com outros utilizadores do GeoNode.

O link *Favorites*, também acessível a partir do menu de utilizador, leva à lista de recursos marcados como seus favoritos.

Clique no botão *Delete from Favorites* para remover o recurso da lista.

O link *My Activities* permite ver todas as suas actividades recentes no GeoNode, como upload de camadas e criação de mapas.

Este link está também disponível no menu do utilizador.

Todos os outros links e suas funcionalidades serão descritos em detalhe nas secções seguintes.

 **GeoNode** [Data](#) [Maps](#) [About](#)

Q Search  johnsmith

johnsmith



johnsmith

Position	Not provided.
Organization	Not provided.
Location	Not provided.
Voice	Not provided.
Fax	Not provided.
Description	Not provided.
Keywords	Not provided

 User layers WMS GetCapabilities document

[Message User](#)
[Edit profile](#)
[Connected social accounts](#)
[Associated e-mails](#)
[Set/Change password](#)
[Upload new layers](#)
[Create a new layer](#)
[Create a new map](#)
[My Activities](#)
[Favorites](#)
[Notifications](#)
[Invite Users](#)

Resources

[All contents](#) [Layers](#) [Maps](#) [Documents](#)



OSM Railways

Railways extracted from OSM

 johnsmith  4 Jun 2019  0  0  0

[View Map](#)



railways

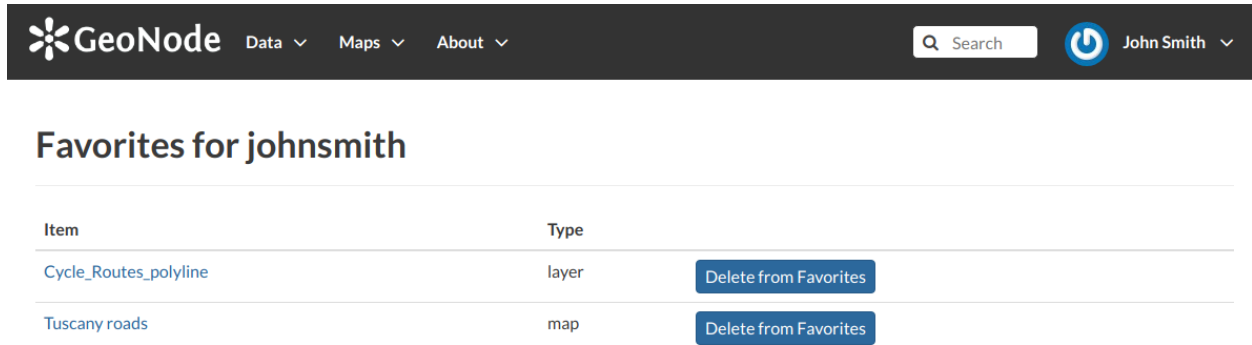
No abstract provided

 johnsmith  4 Jun 2019  0  0  0

[Create a Map](#)

< page 1 of 1 >

Fig. 12: Página de perfil do utilizador

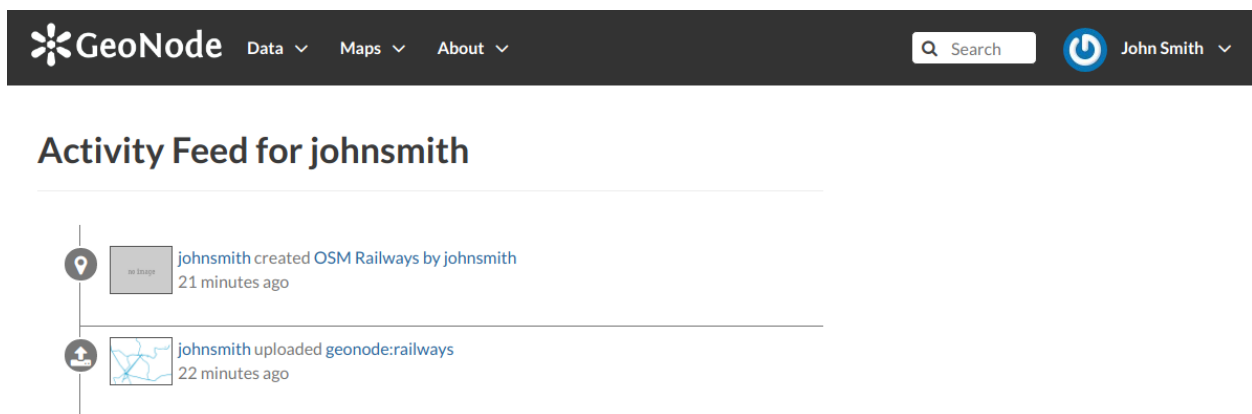


GeoNode Data ▾ Maps ▾ About ▾ John Smith ▾

Favorites for johnsmith

Item	Type	
Cycle_Routes_polyline	layer	Delete from Favorites
Tuscany roads	map	Delete from Favorites

Fig. 13: *Favoritos*



GeoNode Data ▾ Maps ▾ About ▾ John Smith ▾

Activity Feed for johnsmith

-   johnsmith created [OSM Railways by johnsmith](#)
21 minutes ago
-   johnsmith uploaded [geonode:railways](#)
22 minutes ago

Fig. 14: *Actividades recientes*

Editando as Informações de Perfil

O seu *Perfil* contém informações pessoais como endereço, número de telefone, empresa e assim por diante, mas está vazio por defeito no início.

Através do botão *Edit profile* da página *Perfil* (ver [Actualização do Perfil](#)) pode definir os seus detalhes, incluindo seu avatar.

Quando terminar, clique em *Update profile*. Será redireccionado para a página de *Perfil*. Uma mensagem irá confirmar que o perfil foi actualizado correctamente.

Ligando a sua Conta com Redes Sociais

De momento, não possui contas de redes sociais associadas a esta conta.

Associando a sua conta a um email

A sua conta é automaticamente associada ao email que você utilizou para se registar na plataforma.

Ao clicar em *Associated e-mails* da página *Perfil* (veja [Actualização do Perfil](#)), terá a possibilidade de preencher um novo endereço de email. Digite no campo *E-mail* e clique em *Add E-mail* para fazer uma nova associação.

Pode torná-lo primário se necessário, para que receba a notificação neste endereço. Para isso, seleccione o email que deseja e clique em *Make Primary*.

Gestão da Senha

Para alterar a sua senha, clique no link *Set/Change password* da página de *Perfil* (veja [Actualização do Perfil](#)). Será solicitado para inserir a sua senha actual e a nova (duas vezes). Clique em *Change my password* para fazer a alteração.

Se não ocorrer nenhum erro, verá uma mensagem de confirmação.

Na próxima vez que fizer login, terá que usar a nova senha.

Definir Preferências de Notificação

Por padrão, o GeoNode envia notificações aos utilizadores para eventos que podem subscrever, tais como uma nova camada carregada ou uma nova classificação adicionada a um mapa. Você pode ajustar as suas configurações de notificação clicando no link *Notifications* da página de *Perfil* (veja [Actualização do Perfil](#)).

Nota: Certifique-se de ter um endereço de email verificado para o qual os avisos podem ser enviados. Se não, veja [Associando a sua conta a um email](#).

Agora active/desactive os tipos de notificação que deseja ou não receber. É possível ser notificado para os eventos mostrados na imagem em baixo.

 Data ▾ Maps ▾ About ▾

Search  johnsmith ▾

Edit Your Profile


Change your avatar

First name

Last name

Email address

Organization Name

name of the responsible organization

Profile

John Smith profile

Position Name

role or position of the responsible person

Voice

telephone number by which individuals can speak to the responsible organization or individual

Facsimile

telephone number of a facsimile machine for the responsible organization or individual

Delivery Point

physical and email address at which the organization or individual may be contacted

City

city of the location

Administrative Area

state, province of the location

Postal Code

ZIP or other postal code

Country

country of the physical address

Keywords

A space or comma-separated list of keywords

Language

Timezone

Update profile

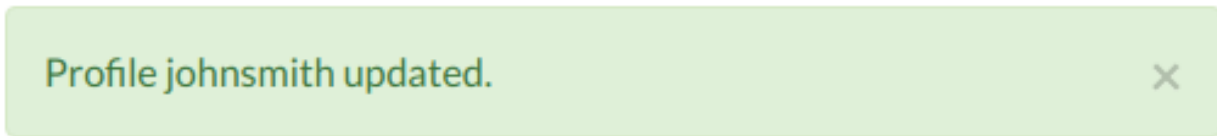


Fig. 16: Atualização do Perfil finalizada correctamente

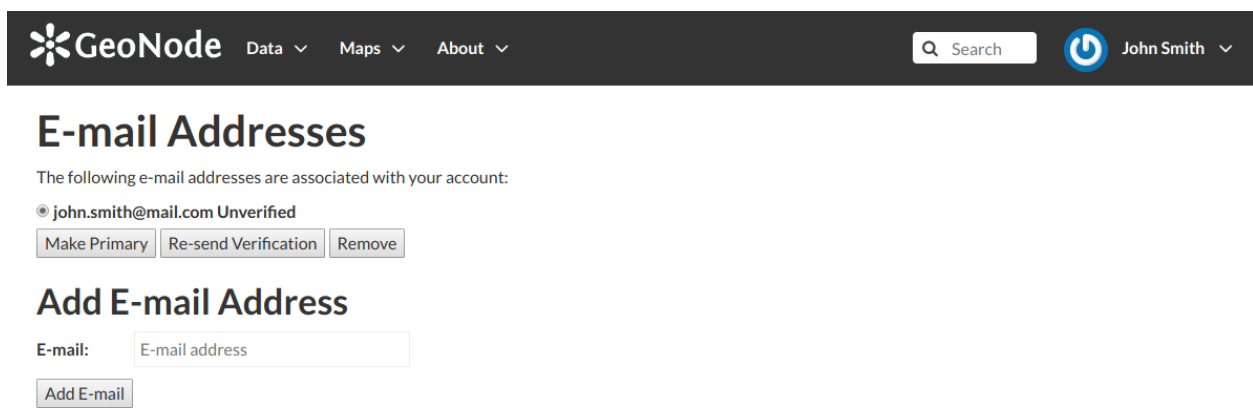


Fig. 17: Contas de email

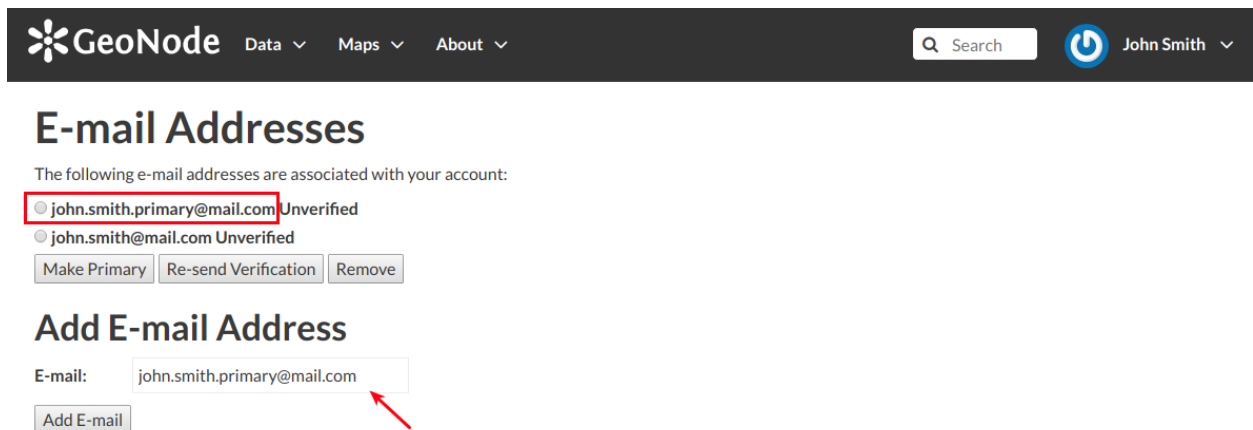
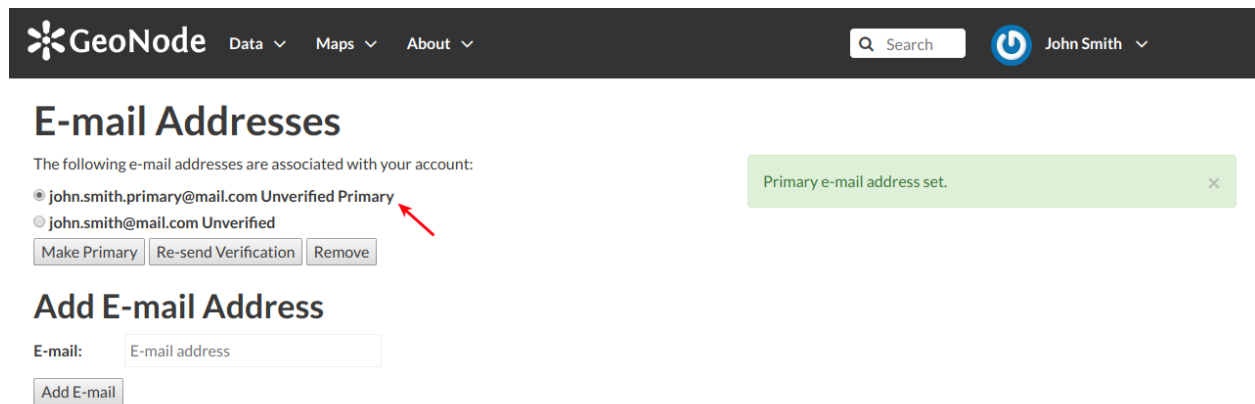


Fig. 18: Nova associação de e-mail



E-mail Addresses

The following e-mail addresses are associated with your account:

- john.smith.primary@mail.com Unverified Primary
- john.smith@mail.com Unverified

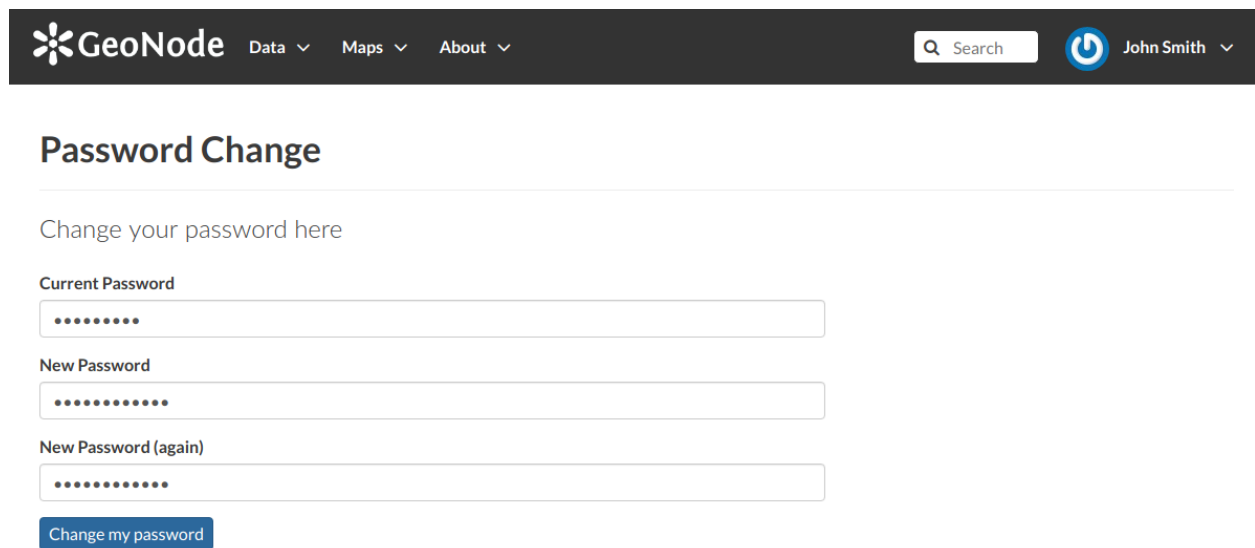
Make Primary Re-send Verification Remove

Add E-mail Address

E-mail:

Add E-mail

Fig. 19: Endereço de email primário



Password Change

Change your password here

Current Password

.....

New Password

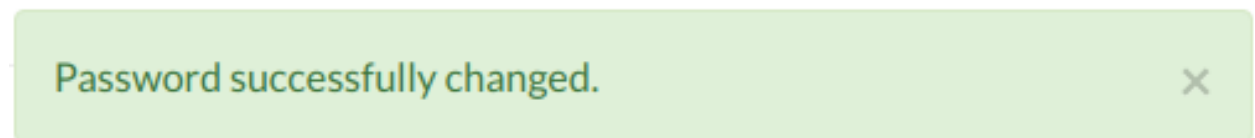
.....

New Password (again)

.....

Change my password

Fig. 20: Mude a sua senha



Password successfully changed.

Fig. 21: Confirmação de alteração de senha


Data ▾
Maps ▾
About ▾




John Smith ▾

Notification Settings

Notification Type	Email
Request to download a resource A request for downloading a resource was sent	☒
Layer Created A Layer was created	☒
Layer Updated A Layer was updated	☒
Layer Approved A Layer was approved by a Manager	☒
Layer Published A Layer was published	☒
Layer Deleted A Layer was deleted	☒
Comment on Layer A layer was commented on	☒
Rating for Layer A rating was given to a layer	☒
Map Created A Map was created	☒
Map Updated A Map was updated	☒
Map Approved A Map was approved by a Manager	☒
Map Published A Map was published	☒
Map Deleted A Map was deleted	☒
Comment on Map A map was commented on	☒
Rating for Map A rating was given to a map	☒
Document Created A Document was created	☒
Document Updated A Document was updated	☒
Document Approved A Document was approved by a Manager	☒
Document Published A Document was published	☒
Document Deleted A Document was deleted	☒
Comment on Document A Document was commented on	☒
Document for Map A rating was given to a document	☒
User following you Another user has started following you	☒
User requested access A new user has requested access to the site	☒
Account activated This account is now active and can log in the site	☒
Layer Uploaded A layer was uploaded	☒
Monitoring alert Alert situation reported by monitoring	☒

Change

Fig. 22: Configurações de notificações

1.10.2 Interagindo com Utilizadores e Grupos

A plataforma GeoNode permite que comunique por mensagem com outros utilizadores do GeoNode e grupos de utilizadores.

Pode, também, convidar utilizadores externos para se inscrever no seu GeoNode. Para fazer isso, clique em *Invite Users* na página *Perfil* (veja *Actualização do Perfil*) ou no menu *About* na página *Home*.

Pode convidar seus contactos digitando seus endereços de email no campo de entrada conforme mostrado na imagem em baixo. Clique em *Submit* para enviar os convites.

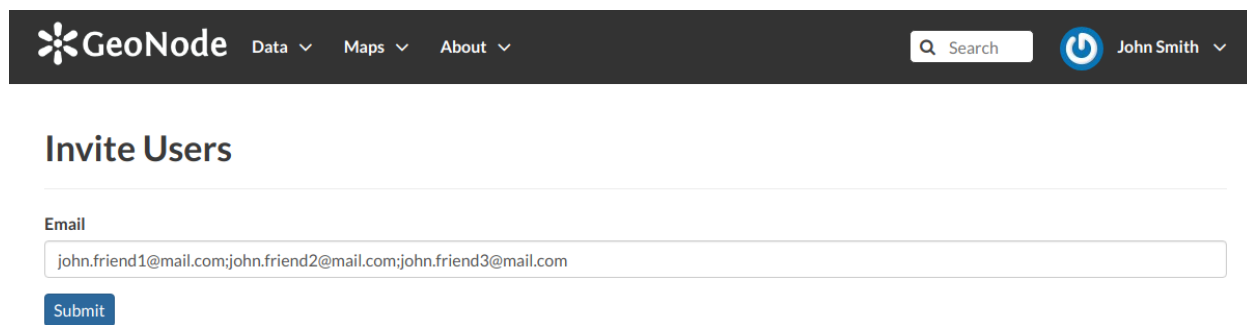


Fig. 23: Convide utilizadores para se juntarem ao GeoNode

Uma mensagem irá confirmar que os convites foram enviados correctamente.



Invitations successfully sent to 'john.friend1@mail.com, john.friend2@mail.com, john.friend3@mail.com'

Fig. 24: Convites confirmam mensagem

Nota: Pode convidar mais de um utilizador ao mesmo tempo, digitando os endereços de email seguidos com ponto e vírgula (;) a separar.

As próximas secções irão mostrar como visualizar informações sobre outros utilizadores e como contactá-los.

Visualização de informações de outros utilizadores

Quando a sua conta é criada, poderá ver outras contas no sistema.

Para ver informações sobre outros utilizadores no sistema, clique no link *People* do menu *About* na página *Home*.

Você irá ver uma lista de utilizadores registados no sistema.

A ferramenta *Pesquisar* é muito útil no caso de muitos utilizadores registados, digite o nome do utilizador que está à procura no campo de texto de entrada para filtrar a lista de utilizadores.

Selecione um utilizador e clique em seu *nome de utilizador* para aceder à página de detalhes do utilizador.

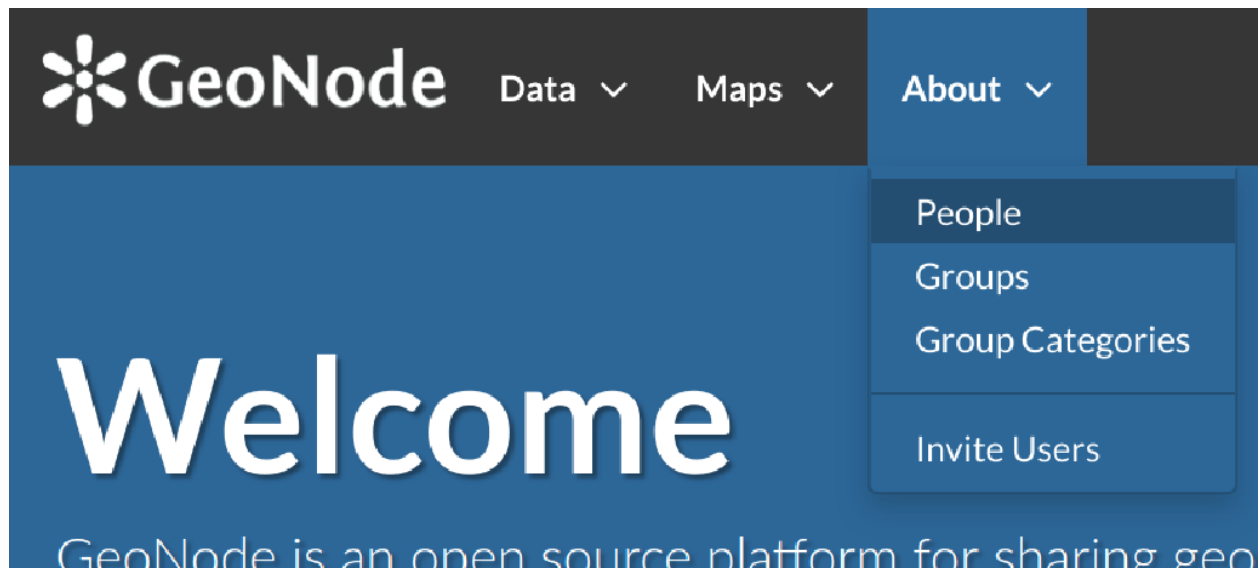


Fig. 25: Menu Sobre - ligar Pessoas

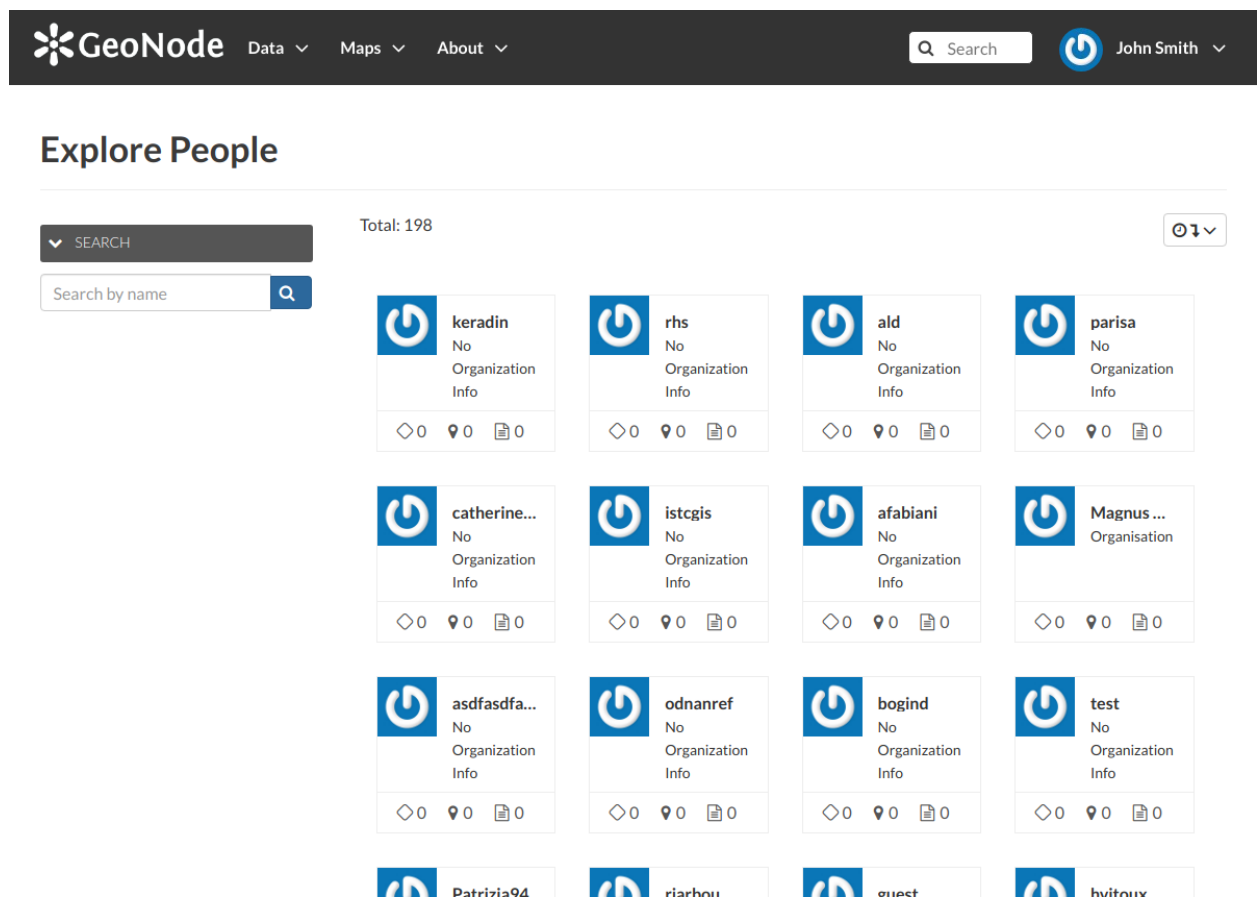


Fig. 26: Lista dos utilizadores registados

 Data ▾ Maps ▾ About ▾

Search

 John Smith ▾

**Mario**

Position	Not provided.
Organization	Mario Rossi Transport
Location	Not provided.
Voice	Not provided.
Fax	Not provided.
Description	Not provided.
Keywords	Not provided

[Message User](#)[User Activities](#)[User layers WMS GetCapabilities document](#)[All contents](#) [Layers](#) [Maps](#) [Documents](#)

no image

Tuscany roads
Roads of Tuscany
Mario Rossi 3 Jun 2019 0 0 0
[View Map](#)



roads
No abstract provided
Mario Rossi 3 Jun 2019 0 0 0
[Create a Map](#)

< page 1 of 1 >

Fig. 27: Detalhes do utilizador

Nesta página são apresentadas as principais informações do utilizador: dados pessoais (nome, apelido, organização, etc.) e os recursos que o utilizador possui (camadas, mapas e documentos).

Através do link *User Activities*, no lado direito da página, é possível visualizar todas as actividades realizadas pelo utilizador.

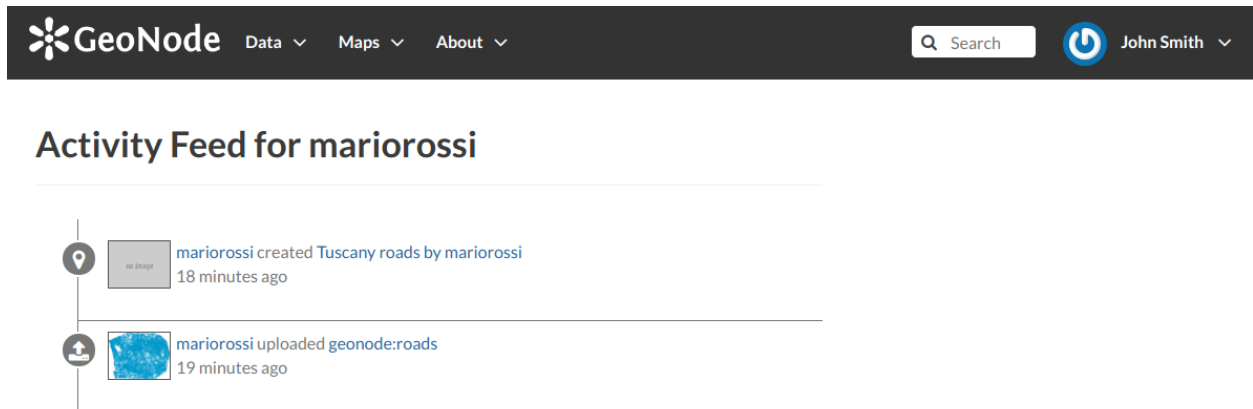


Fig. 28: Actividades do utilizador

A ligação *Message User* permite que contacte outros utilizadores, consulte a próxima secção para ler mais sobre isso. É também possível, no GeoNode, ver as actividades recentes de todos os utilizadores através da ligação *Recent Activities* do menu do utilizador.

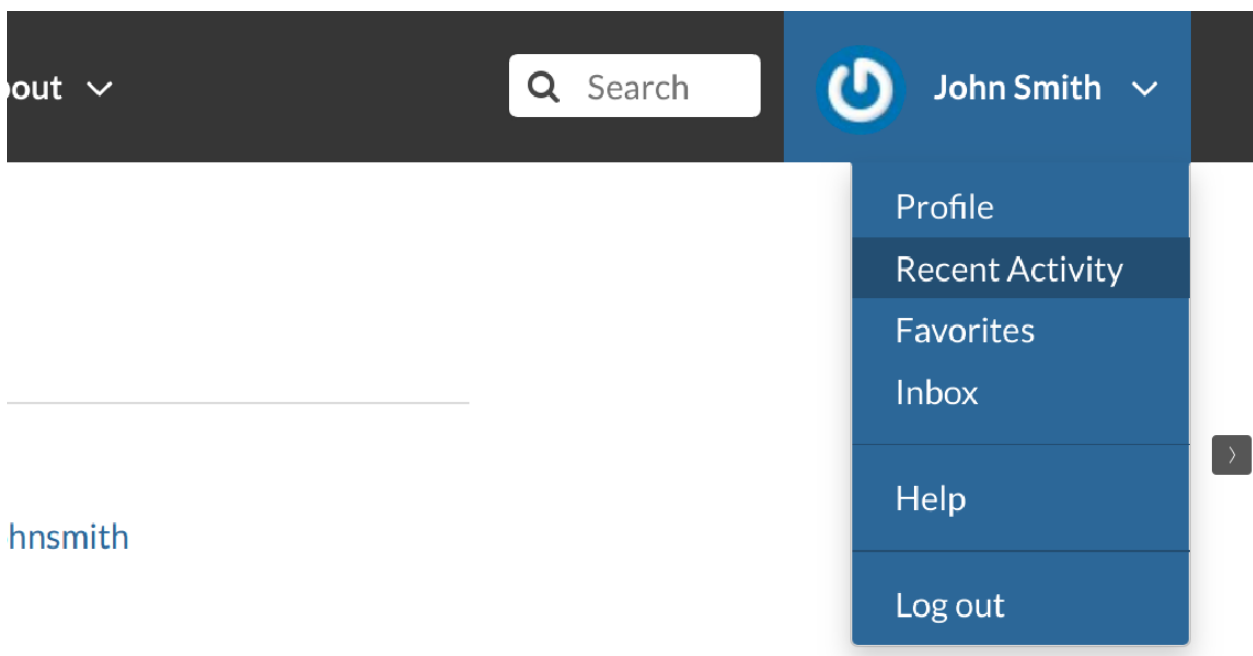


Fig. 29: Ligação de Actividades Recentes

Na foto em baixo há um exemplo.

Como pode ver, pode decidir se deseja ver apenas as actividades relacionadas a camadas ou aquelas relacionadas a mapas ou comentários alternando as guias.



[Data](#) [Maps](#) [About](#)

 [John Smith](#)

Recent activity

[All](#) [Layers](#) [Maps](#) [Comments](#)



 [johnsmith](#) created [OSM Railways](#) by [johnsmith](#)
24 minutes ago



 [johnsmith](#) uploaded [geonode:railways](#)
25 minutes ago



 [cbardalesc](#) subido [geonode:](#)
14 hours, 18 minutes ago



 [cbardalesc](#) subido [geonode:Crhc0](#)
14 hours, 18 minutes ago



 [cbardalesc](#) subido [geonode:Crhc](#)
14 hours, 19 minutes ago



 [Palma](#) creato [SELVA](#) by [Palma](#)
14 hours, 20 minutes ago



 [Palma](#) aggiornato [geonode:latina3](#)
14 hours, 36 minutes ago



 [Palma](#) aggiornato [geonode:viterbo](#)
15 hours, 28 minutes ago

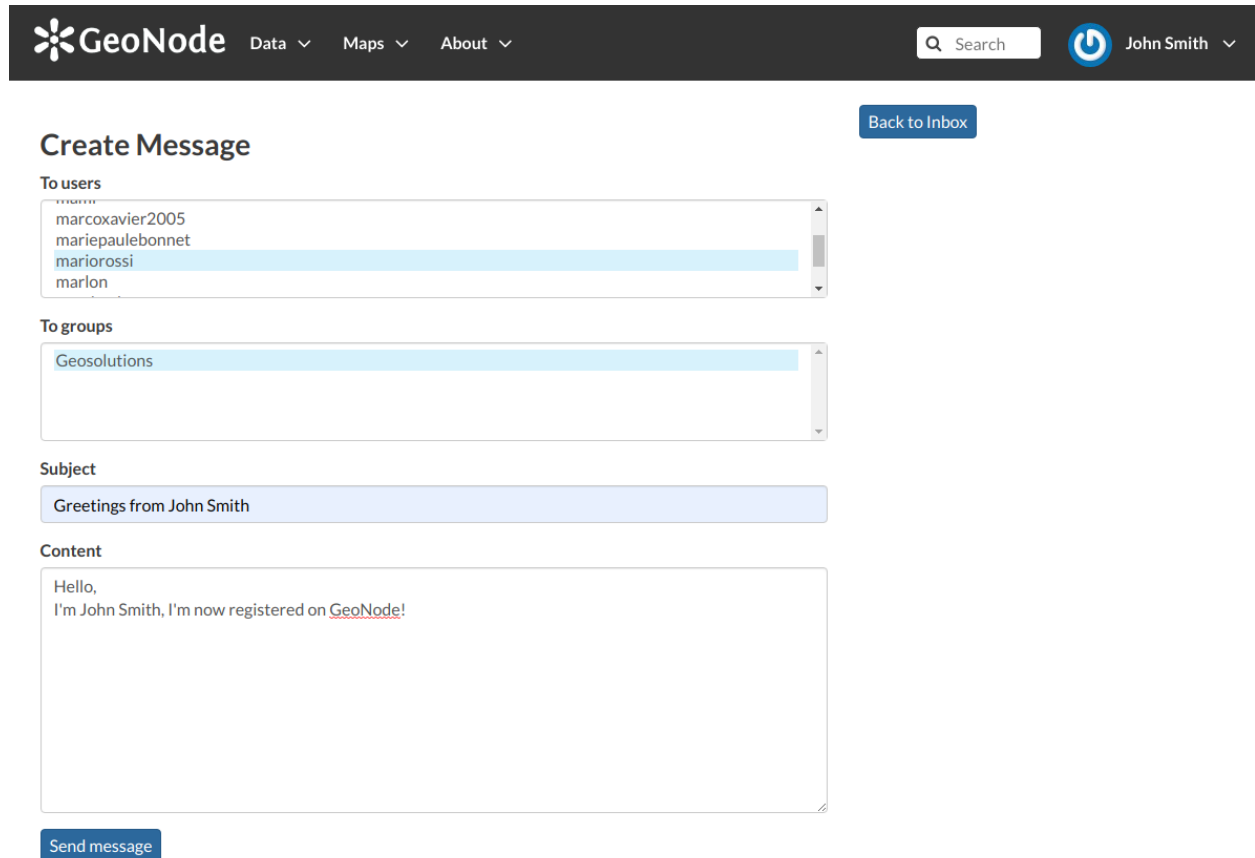


 [pedroalves10](#) criado [teste](#) by [pedroalves10](#)

Fig. 30: *Actividades Recentes*

Contacto com outros utilizadores

O GeoNode permite que um utilizador comunique por mensagem com outros utilizadores e grupos registados. Para enviar uma mensagem para outro utilizador e/ou grupos, você pode seguir o link *Message User* da sua página de *Perfil* (veja *Actualização do Perfil*) ou da página de detalhes de *Perfil* (veja a secção anterior *Visualização de informações de outros utilizadores*) desse utilizador.



The screenshot shows the 'Create Message' interface on the GeoNode website. At the top is a dark navigation bar with the GeoNode logo, links for 'Data', 'Maps', and 'About', a search bar, and a user profile for 'John Smith'. Below the navigation bar is a 'Back to Inbox' button. The main form is titled 'Create Message' and includes several sections: 'To users' with a list of usernames (marcoxavier2005, mariepaulebonnet, mariorossi, marlon) where 'mariorossi' is selected; 'To groups' with a list of group names (Geosolutions) where 'Geosolutions' is selected; 'Subject' with a text input field containing 'Greetings from John Smith'; and 'Content' with a large text area containing the message 'Hello, I'm John Smith, I'm now registered on [GeoNode!](#)'. At the bottom of the form is a 'Send message' button.

Fig. 31: Envie uma mensagem para utilizadores e grupos

Insira o seu conteúdo, digite um assunto e clique em *Send message* para enviar a mensagem aos utilizadores e grupos que seleccionou.

Será redireccionado para a página de detalhes da *Conversa* referente ao assunto.

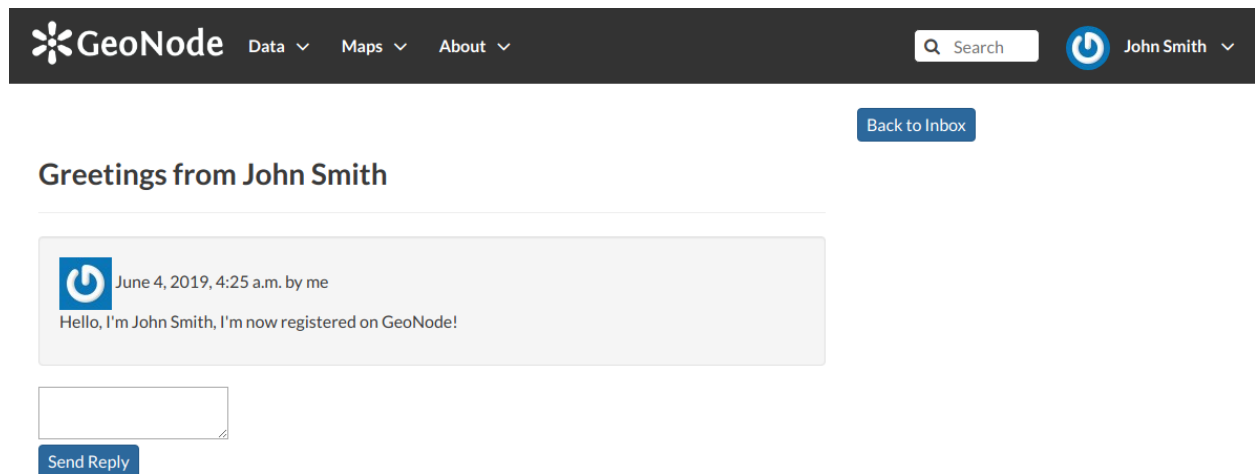


Fig. 32: A sua mensagem

A página da caixa de Entrada

Pode ver suas conversas na página *Caixa de Entrada*, acessível através do botão *Back to inbox* (veja a imagem em cima) ou da ligação *Inbox* do menu de utilizador.

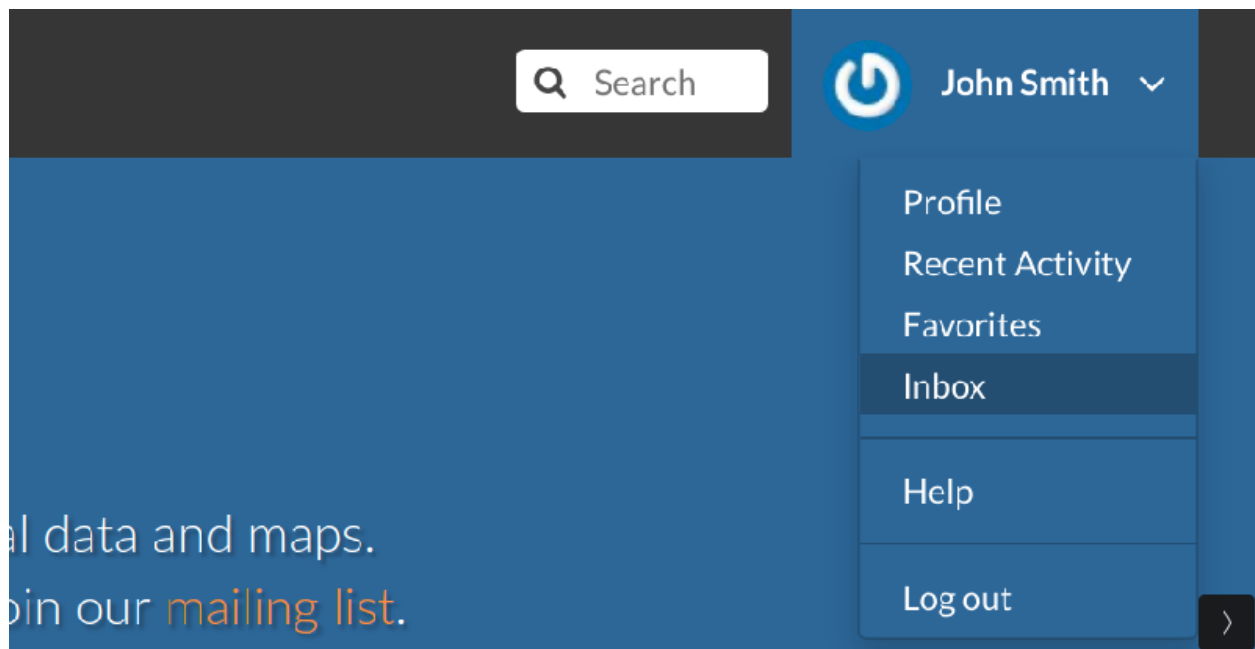


Fig. 33: Link da caixa de entrada

A imagem em baixo mostra como deve ser a aparência da sua página *Caixa de Entrada*.

Na *Caixa de entrada* todas as mensagens não lidas são listadas. Ainda não recebeu uma mensagem, pelo que a sua *Caixa de entrada* está vazia. Se alternar para a guia *Todas*, poderá ver todas as conversas nas quais está envolvido.

Quando algum utilizador envia uma resposta à sua mensagem, a sua *Caixa de Entrada* mostra-a. Veja a imagem em baixo como exemplo.

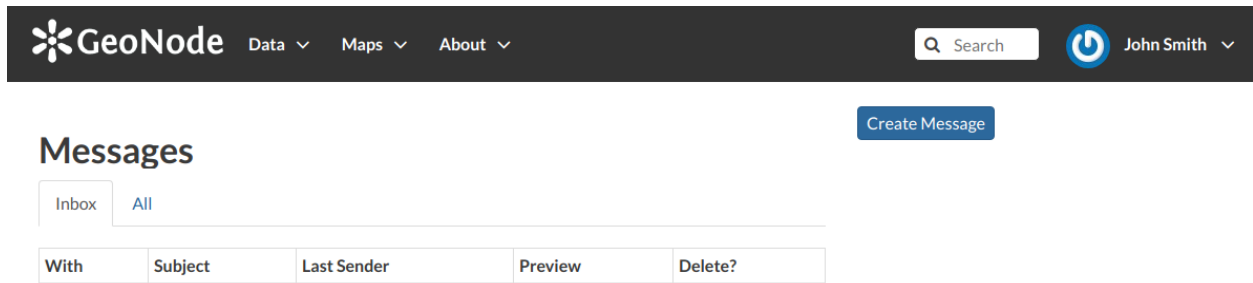


Fig. 34: Página da caixa de entrada

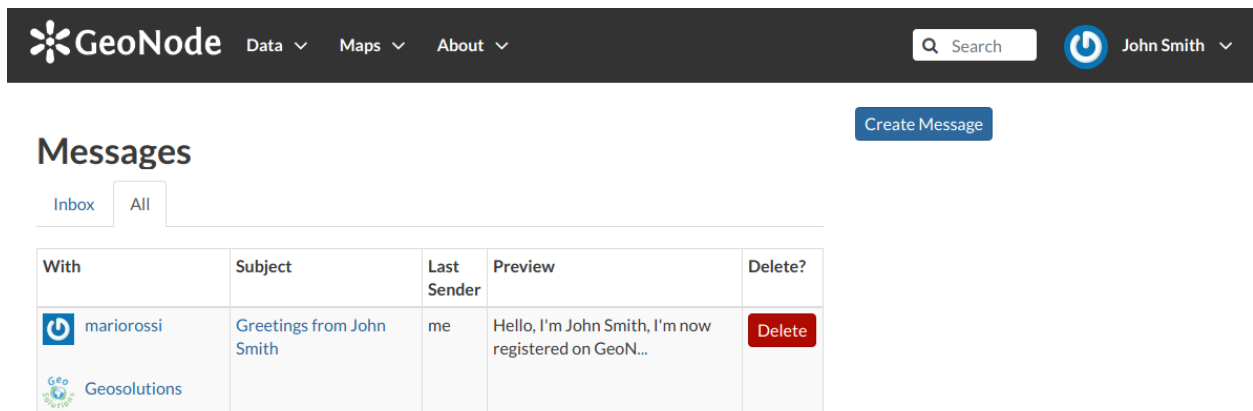


Fig. 35: Todas as suas conversas

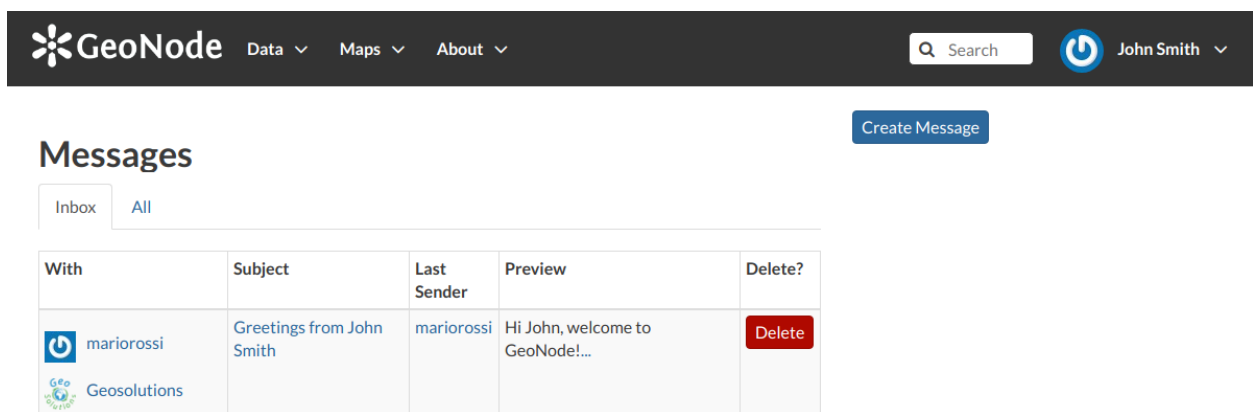


Fig. 36: Resposta à sua mensagem

Pode abrir os detalhes da *Conversa* clicando no link *Assunto*.

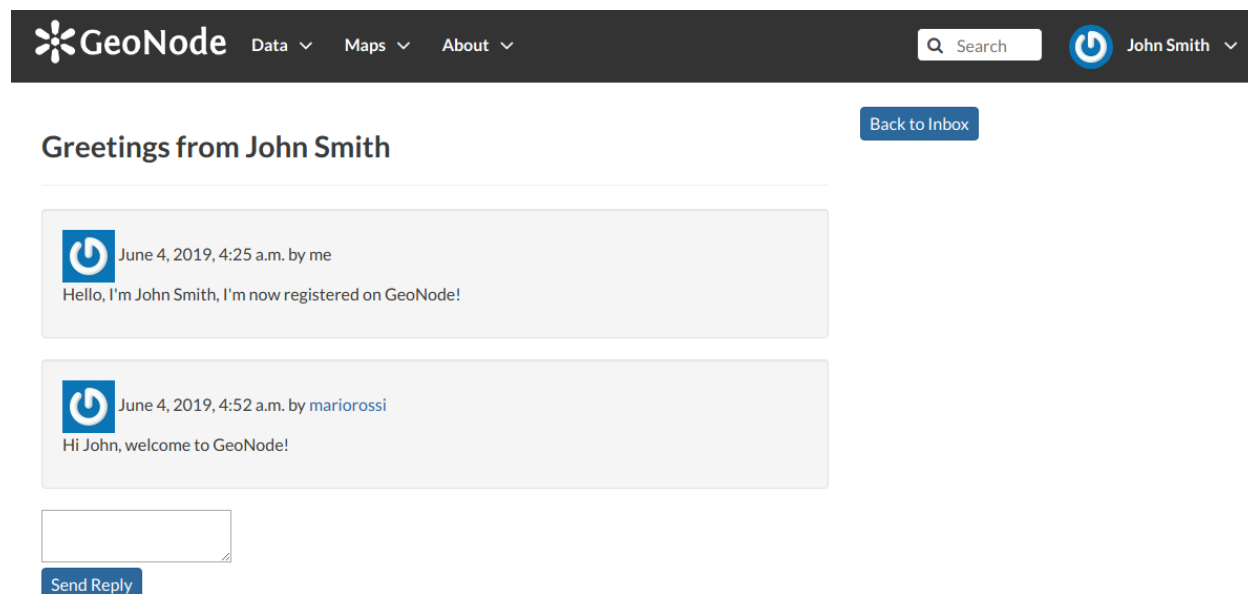


Fig. 37: Detalhes da conversa

Como pode ver na imagem em cima, na página *Conversa* pode escrever uma resposta rápida. Digite a sua mensagem na caixa de texto e clique em *Send Reply* para enviar.

Na página *Caixa de Entrada* há também o botão *Create Message* que fornece uma ligação rápida para o formulário de criação de mensagem.

1.10.3 Dados

As ferramentas de gestão de dados incorporadas no GeoNode permitem a criação integrada de dados, documentos, ligações para documentos externos e visualizações de mapas. Cada conjunto de dados no sistema pode ser partilhado publicamente ou restrito para permitir o acesso apenas a utilizadores específicos. Recursos sociais como perfis de utilizador e sistemas de comentários e classificação permitem o desenvolvimento de comunidades em torno de cada plataforma para facilitar o uso, gestão e controle de qualidade dos dados que a instância GeoNode contém.

As secções a seguir irão explicar mais detalhadamente quais dados podem ser geridos no GeoNode e como encontrar esses dados facilmente.

Tipos de Dados

A página de boas-vindas do GeoNode mostra uma variedade de informações sobre a instância actual do GeoNode. Você pode explorar os dados existentes usando muitas ferramentas de pesquisa e filtros (veja [Procurando Dados](#)) ou através das ligações da barra de navegação no topo da página.

Existem três tipos principais de recursos que o GeoNode pode fazer a gestão:

1. Documentos
2. Camadas
3. Mapas

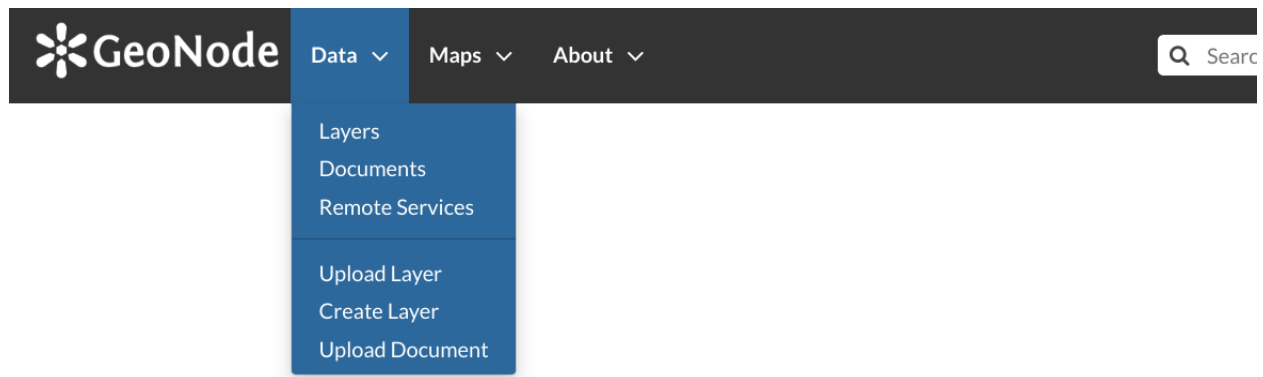


Fig. 38: Menu de dados

Documentos e camadas podem ser acessados a partir do menu *Data* da barra de navegação.

O menu *Maps* permite que faça a gestão dos mapas.

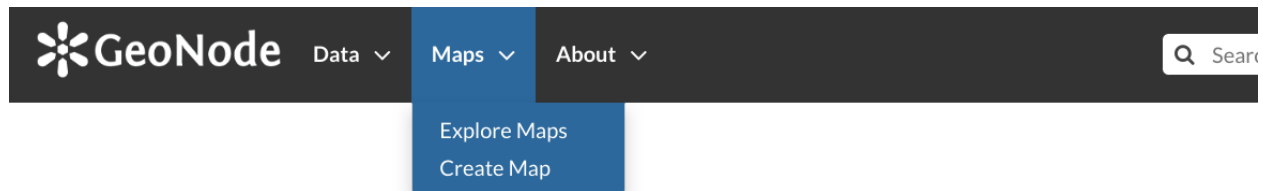


Fig. 39: Menu de mapas

Documentos

GeoNode permite publicar dados tabulares e de texto e fazer a gestão de metadados e documentos associados.

Documentos podem ser carregados directamente do seu disco (veja [Carregando Documentos](#) para mais informações).

Os seguintes tipos de documentos são permitidos: *.doc*, *.docx*, *.gif*, *.jpg*, *.jpeg*, *.ods*, *.odt*, *.odp*, *.pdf*, *.png*, *.ppt*, *.pptx*, *.rar*, *.sld*, *.tif*, *.tiff*, *.txt*, *.xls*, *.xlsx*, *.xml*, *.zip*, *.gz*, *.qml*.

Através da página detalhada do documento é possível visualizar, descarregar e gerir um documento.

Camadas

As camadas são um componente principal do GeoNode.

Camadas são recursos publicáveis que representam uma fonte de dados espaciais raster ou vectoriais. As camadas podem também ser associadas a metadados, classificações e comentários.

Ao clicar no link Camadas, obterá uma lista de todas as camadas publicadas. Se estiver ligado como administrador, você poderá também, ver as camadas não publicadas na mesma lista.

GeoNode permite ao utilizador fazer carregamento de dados vectoriais e raster nas suas projecções originais usando um formulário da web.

Os dados vectoriais podem ser enviados em formatos diferentes (ESRI Shapefile, KML e assim por diante...). Imagens

de satélite e outros tipos de dados raster podem ser carregados como GeoTIFFs.

Mapas

Os mapas são um componente principal do GeoNode.

Os mapas são compostos por várias camadas e seus estilos. As camadas podem ser tanto camadas locais no GeoNode quanto camadas remotas, publicadas por outros servidores WMS ou por camadas de serviço da web, como Google ou MapQuest.

Os mapas GeoNode também contêm outras informações, como zoom e extensão do mapa, ordenação de camadas e estilo.

Pode criar um mapa baseado em camadas carregadas, combiná-las com algumas camadas existentes e uma camada de serviço remoto da web, compartilhar o mapa resultante para exibição pública. Uma vez que os dados tenham sido carregados, o GeoNode permite ao utilizador pesquisá-los geograficamente ou por meio de palavras-chave e criar mapas. Todas as camadas são reprojectadas automaticamente para o web mercator para exibição de mapas, tornando possível o uso de mapas básicos populares como o [OpenStreetMap](#).

Procurando Dados

Esta secção irá guiá-lo para navegar o GeoNode para encontrar camadas, mapas e documentos usando diferentes rotas, filtros e funções de pesquisa.

Na página *Home* poderá encontrar algumas ferramentas de pesquisa rápida.

A caixa *Pesquisar* na barra de navegação (veja a imagem abaixo) permite que digite umas palavras e encontre todos os dados que têm semelhança com esse texto.



Fig. 40: Ferramenta de pesquisa na página de boas-vindas do GeoNode

Ao iniciar uma pesquisa, é levado à página *Pesquisar*, que mostra o resultado da pesquisa em todos os tipos de dados.

Esta página contém uma variedade de opções para personalizar uma busca por várias informações no GeoNode. Este formulário de pesquisa permite pesquisas muito mais ajustadas do que a simples caixa de pesquisa disponível no topo de cada página.

É possível pesquisar dados por Texto, Categorias, Tipo, Palavras-chave, Proprietários, Data, Regiões ou Extensão.

Tente definir algum filtro e ver como a lista de dados resultante muda de acordo. Um tipo interessante de filtro é *EXTENSÃO*: pode aplicar um filtro espacial movendo ou ampliando um mapa dentro de uma caixa, conforme mostrado na imagem abaixo.

Os dados podem ser ordenados por data, nome e popularidade.

1.10. Manual de Utilizador do GeoNode



Fig. 42: *Filtro de pesquisa por EXTENSÃO*

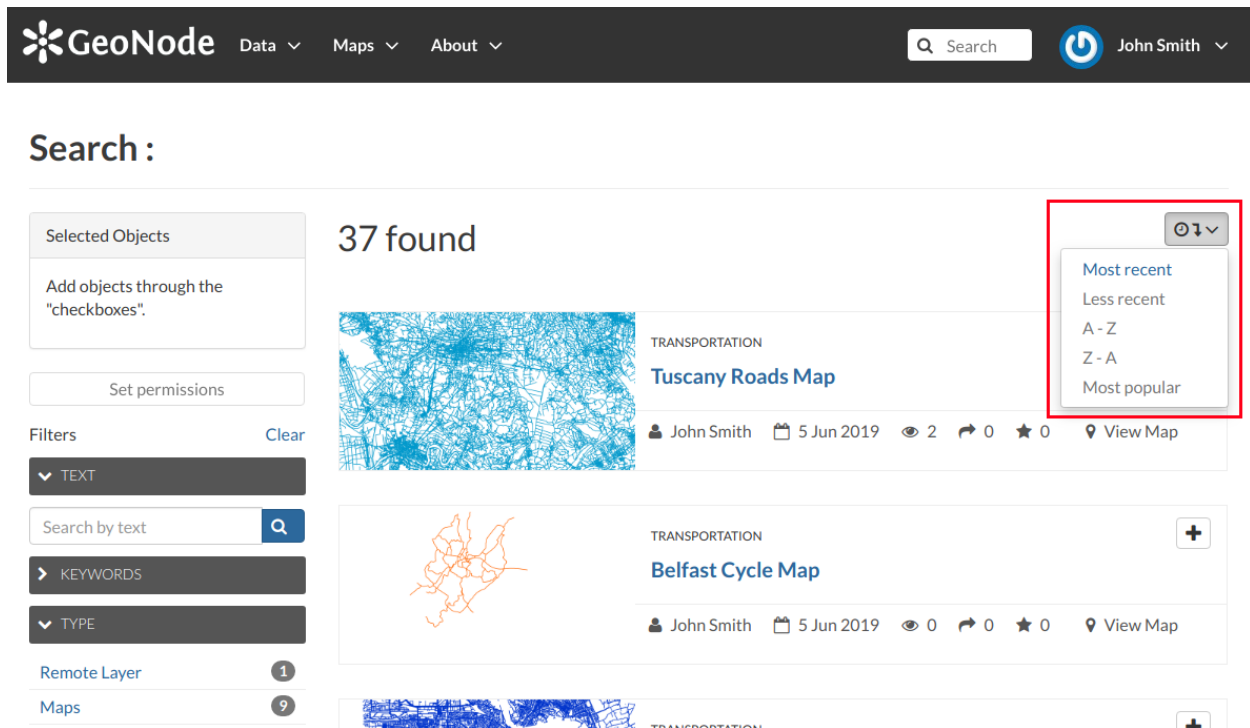


Fig. 43: Ordenando Dados

A página de boas-vindas do GeoNode oferece muitas outras opções para encontrar recursos.

- A ferramenta *Search for data* permite pesquisar dados por nome.

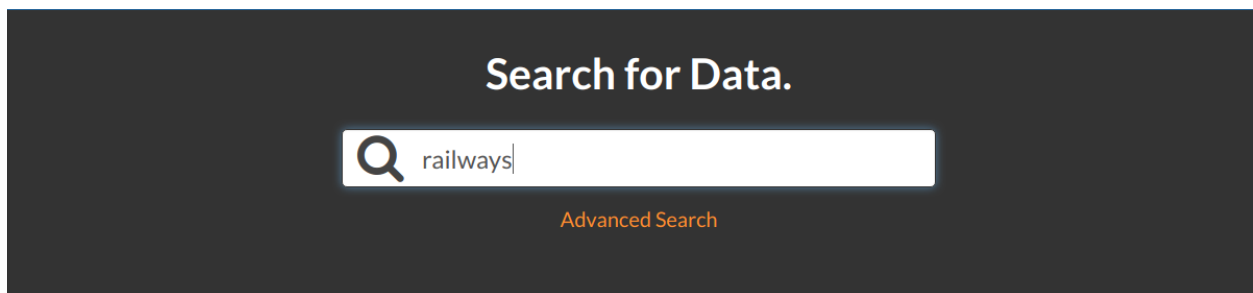


Fig. 44: Pesquisando dados

A página *Search*, para a qual será redireccionado, já terá o filtro de TEXTO configurado com o nome que digitou na caixa de busca (veja a imagem abaixo). Se desejar aceder à *Página de pesquisa* directamente, sem qualquer entrada de texto, pode clicar no link *Advanced Search*.

- Na secção da página *Home* mostrada abaixo, estão listadas todas as categorias disponíveis na instância GeoNode que está a usar. Pode pesquisar dados por categoria clicando nela.

Na página *Pesquisar*, os dados serão filtrados por essa categoria.

- A secção *Featured Datasets* (veja a imagem abaixo) mostra dados agregados sobre *Camadas*, *Mapas*, *Documentos* e *Utilizadores*. Pode accionar uma busca por camadas clicando no ícone *Layers*, o mesmo acontece para *Mapas*, *Documentos* e *Utilizadores*. O *Explore all datasets* o leva para a página *Pesquisar* sem filtro nos tipos de dados.

The screenshot shows the GeoNode search results for the query 'railways'. The top navigation bar includes the GeoNode logo, links for Data, Maps, and About, a search bar, and a user profile for John Smith. The search results section displays '2 found' items. On the left, a sidebar contains filters for TEXT, KEYWORDS, TYPE, CATEGORIES, OWNERS, DATE, REGIONS, and EXTENT. The 'railways' keyword is highlighted in the search bar. The first result is 'OSM Railways', described as 'Railways extracted from OSM', by John Smith, dated 4 Jun 2019, with 7 views and 0 likes. The second result is 'railways', described as 'No abstract provided', also by John Smith, dated 4 Jun 2019, with 9 views and 0 likes. A map thumbnail is visible for the second result. The page is on 'page 1 of 1'.

Fig. 45: Resultados da pesquisa feita por nome

The screenshot shows the 'Discover the available datasets' page on GeoNode. A callout box highlights 'Legal land descriptions. Examples: political and administrative boundaries.' The page features a grid of 20 icons representing different data categories: BIOTA, BOUNDARIES, CLIMATE, ECONOMY, ELEVATION, ENVIRONMENT, FARMING, GEOSCIENCE, HEALTH, BASE MAPS, INLAND WATERS, INTELLIGENCE, LOCATION, OCEANS, PLANNING, POPULATION, SOCIETY, STRUCTURE, TRANSPORTATION, and UTILITIES.

Fig. 46: Pesquisando conjuntos de dados por categoria

The screenshot shows the GeoNode web interface. At the top is a dark navigation bar with the GeoNode logo, links for Data, Maps, and About, a search bar, and a user profile for John Smith. Below the navigation bar, the page is titled "Search :". On the left side, there is a sidebar with various filter options: "Selected Objects", "Add objects through the 'checkboxes'", "Set permissions", "Filters" (with a "Clear" link), "TEXT", "KEYWORDS", "TYPE", "Vector Layer" (with a count of 1), "CATEGORIES", "OWNERS", "DATE", "REGIONS", and "EXTENT". The "CATEGORIES" section is expanded, showing a list of categories: "Boundaries" (with a count of 1 and a red circle around it), "climatologyMeteorolo...", and "Transportation" (with a count of 1). The main content area displays "1 found" at the top right. Below this, there is a map of Italy with a red circle around the word "BOUNDARIES" in the legend. The legend entry is "Com2016_WGS84_g" with the description "Administrative boundaries of Italian municipalities". Below the legend, there is a row of metadata: "John Smith", "5 Jun 2019", "0" views, "0" shares, and "0" stars. At the bottom right of the main content area, there is a pagination bar showing "page 1 of 1".

Fig. 47: Resultados da pesquisa feita por categoria

Nesta secção, também, há links rápidos úteis para adicionar novos recursos: o *Add layers* leva-o para a página de envio de camadas, o *Add documents* para a página de envio de documentos e o *Create maps* para a criação do mapa.

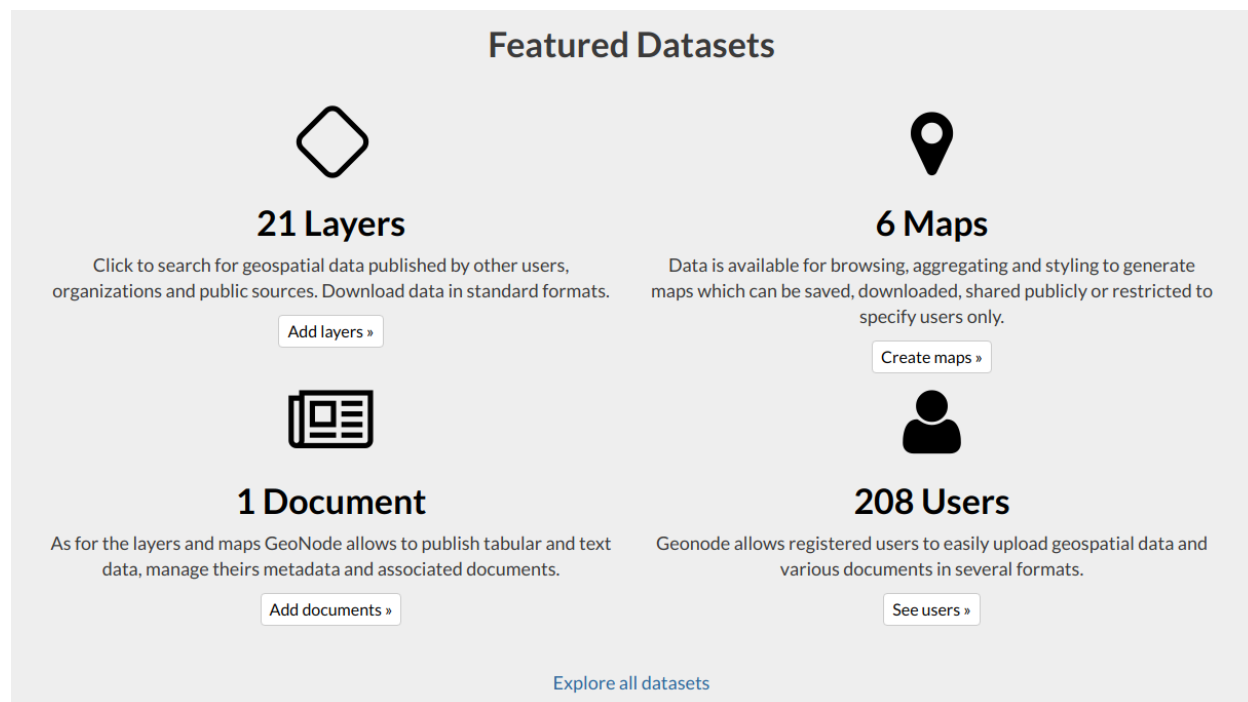


Fig. 48: Conjuntos de Dados em Destaque

Para cada tipo de dados, o GeoNode disponibiliza uma página de *Pesquisa* individual, os próximos parágrafos irão explicar isso em profundidade. Para *Utilizadores* veja [Visualização de informações de outros utilizadores](#).

Documentos

Ao pesquisar por *Documentos*, pode:

- usar o link rápido de *Documentos* da secção *Featured Datasets* conforme descrito em cima
- clicar no link *Documents* do menu *Data* na barra de navegação

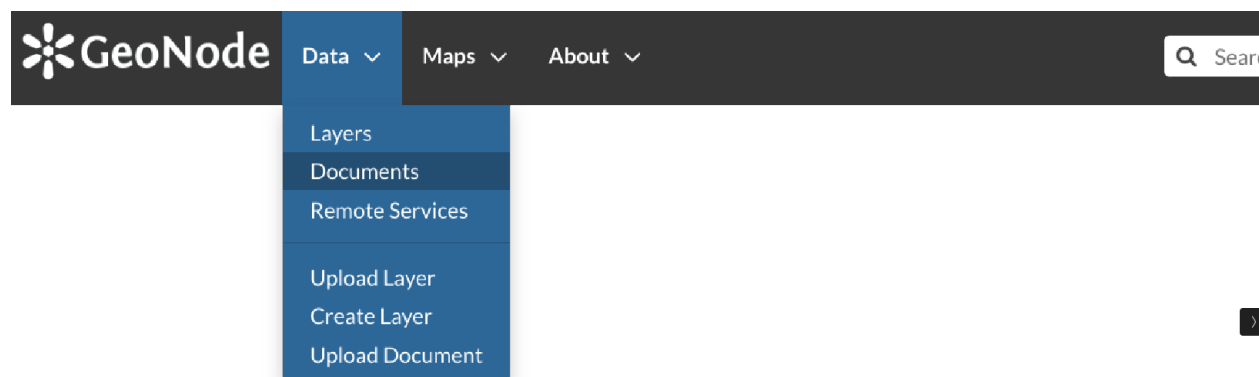


Fig. 49: Link para Documentos

A página de pesquisa *Documentos* parece genérica, mas apenas *Documento* é considerado um tipo de dados. Pode filtrar documentos por CATEGORIAS, como no exemplo em baixo, ou por TEXTO, PALAVRAS-CHAVE e assim por diante. Também pode usar mais de um filtro ao mesmo tempo.

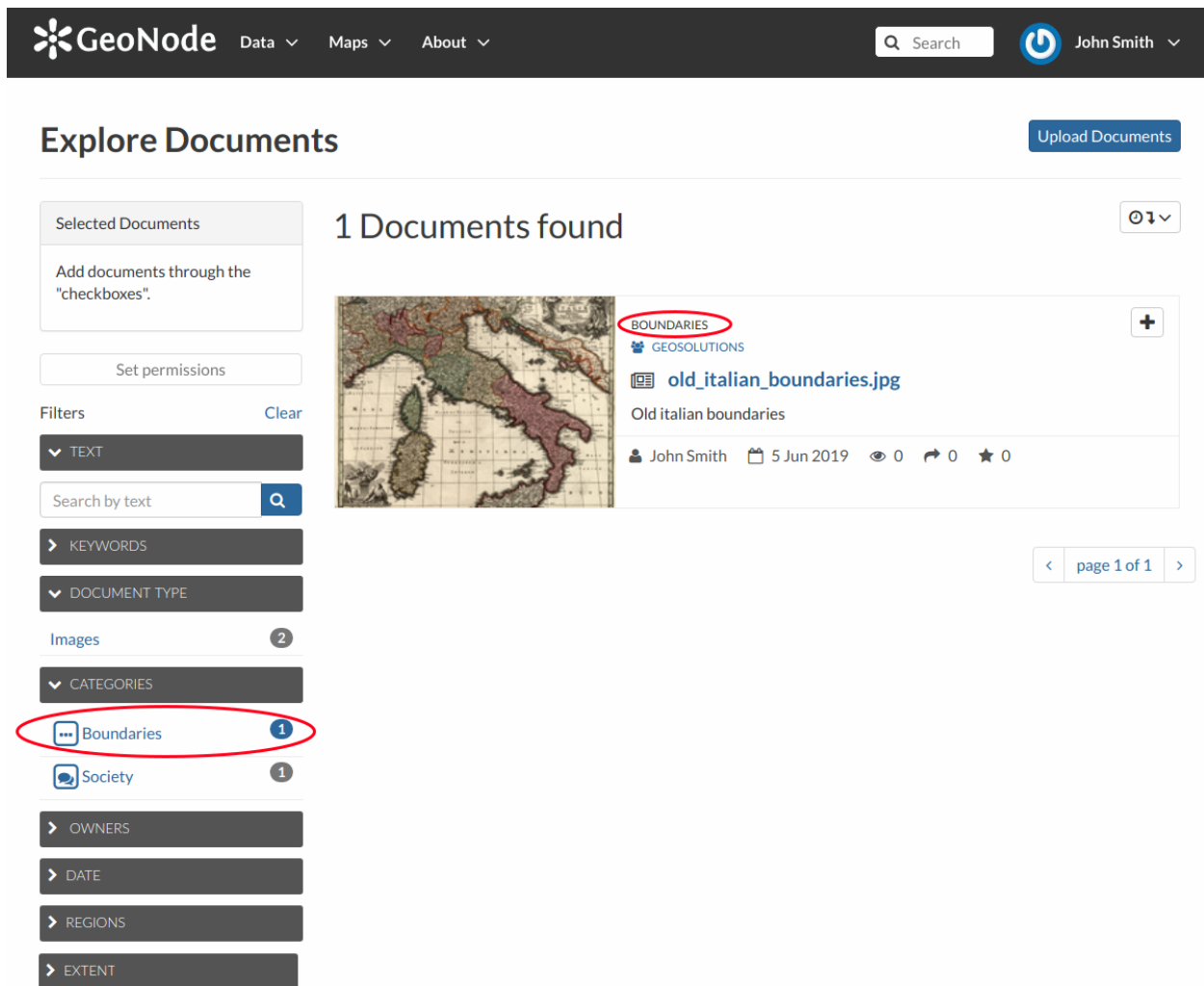


Fig. 50: Documentos filtrados por categorias

Camadas

Para encontrar *Camadas*, pode:

- use o link rápido *Camadas* dos *Featured Datasets*
- clique no link *Layers* do menu *Data* na barra de navegação

Na página de pesquisa *Layers*, apenas *Camada* será considerado como tipo de dado. Pode definir um ou mais filtros para refinar a pesquisa. No exemplo em baixo, as camadas foram filtradas por EXTENSÃO e CATEGORIAS.

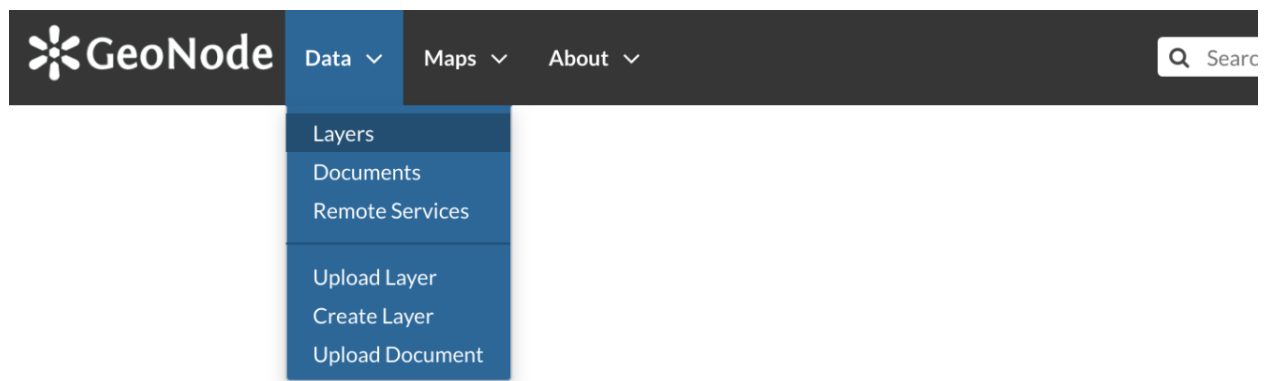


Fig. 51: Link para Camadas

Fig. 52: Camadas filtradas por extensão

Mapas

Se estiver a procurar por *Mapas*, poderá:

- usar o link rápido *Mapas* da secção *Featured Datasets* conforme descrito em cima
- clicar no link *Explore Maps* do menu *Maps* na barra de navegação

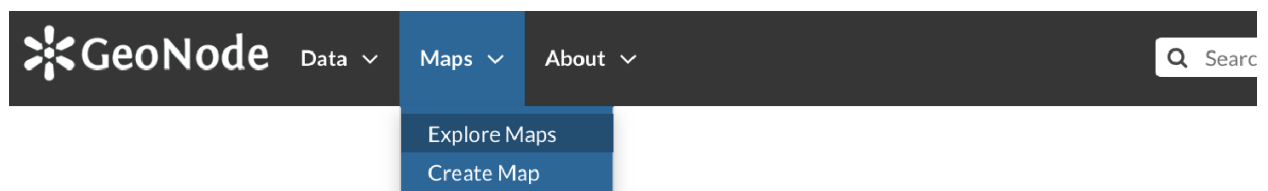


Fig. 53: Link para Mapas

Como visto para os outros tipos de dados, a página de pesquisa *Mapas* permite que filtre os seus mapas por uma combinação de critérios. O exemplo em baixo mostra mapas filtrados por REGIÕES.

Fig. 54: Mapas filtrados por regiões

1.10.4 Fazendo a Gestão de Documentos

Nesta secção, todos os aspectos relativos a *Documentos* serão discutidos.

Aprenderá a carregar um documento e a inspecionar seus metadados e detalhes. Todas as ferramentas de edição também serão explicadas.

Carregando Documentos

O GeoNode permite compartilhar relatórios, notas de concepção, posters, folhas de cálculo, etc. Uma grande variedade de arquivos de documentos podem ser hospedados na plataforma, incluindo arquivos de texto (.doc, .docx, .txt, .odt), planilhas (.xls, .xlsx, .ods), apresentações (.ppt, .pptx, .odp), imagens (.gif, .jpg, .png, .tif, .tiff), PDF, arquivos zip (.rar, .zip, .gz), Arquivos SLD, XML ou QML.

Aviso: Apenas utilizadores autenticados podem carregar dados no GeoNode.

O carregamento de *Documentos* pode ser acedido de duas formas:

- o botão *Upload Documents* da página *Pesquisa de Documentos* (veja *Documentos*)

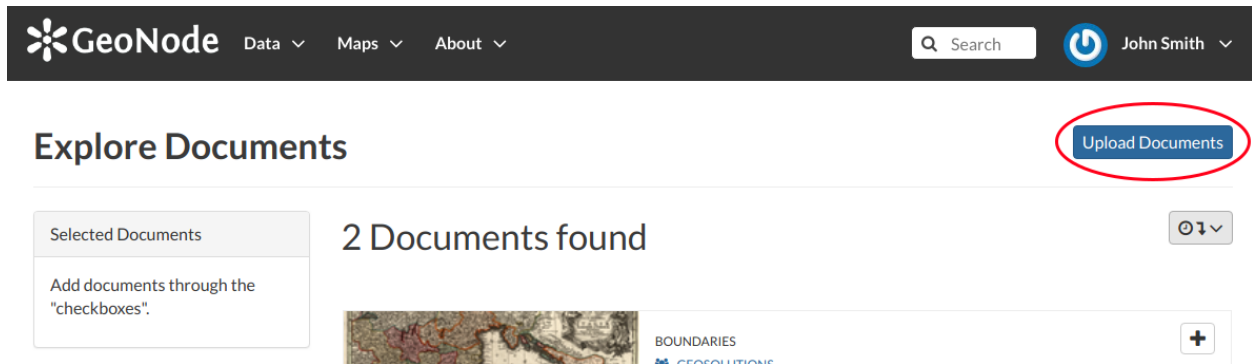


Fig. 55: Botão Carregar Documentos

- o link *Upload Document* do menu *Data* na barra de navegação

A página de *Carregamento de Documentos* é parecida com a mostrada na imagem em baixo.

Para carregar um documento:

1. seleccione um arquivo do seu disco ou insira um endereço URL se o documento estiver armazenado na internet
2. insira o título do documento
3. seleccione um ou mais recursos publicados aos quais o documento pode ser vinculado (opcional)
4. clique no botão vermelho *Upload*

No final do processo de carregamento, será direccionado para a página *Metadados* para preencher outras informações sobre o documento. Veja a próxima secção para saber mais sobre isto.

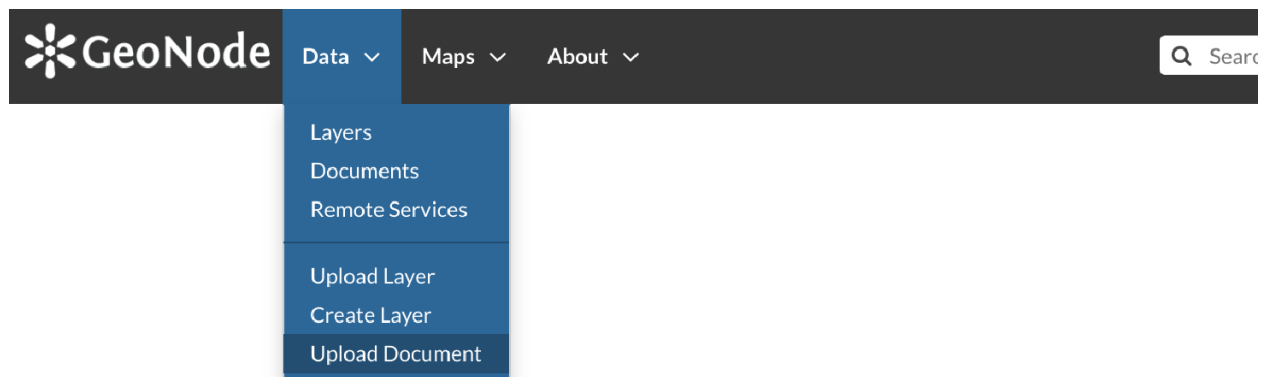


Fig. 56: Link Carregar Documento

Upload Documents [Explore Documents](#)

Allowed document types:

.doc .docx .gif .jpg .jpeg .ods .odt .odp .pdf .png .ppt .pptx .rar .sld .tif .tiff .txt .xls .xlsx .xml .zip .gz .qml

Title:

 name by which the cited resource is known

File:
 john_smith_foundation_annual_report.odt

URL:

 The URL of the document if it is external.

Link to:

Permissions

Who can view it?

☒ Anyone
 The following users:

 The following groups:

Who can download it?

Who can change metadata for it?

Who can manage it? (update, delete, change permissions, publish/unpublish it)

Fig. 57: Página de Carregamento de documentos

Preenchendo os Metadados do Documento

Os metadados contêm todas as informações relativas ao documento: são o seu cartão de identificação. Eles fornecem informações essenciais para sua identificação e compreensão. Os metadados também tornam o documento mais facilmente recuperável por meio de pesquisas por outros utilizadores.

A edição dos metadados de um documento é feita em três etapas (*Metadados Básicos*, *Localização e Licenças*, *Metadados Opcionais*). As duas primeiras etapas são obrigatórias (nenhum documento será publicado se as informações exigidas não forem fornecidas), enquanto a última é opcional.

1. Na página **Metadados Básicos**, as informações essenciais que devem ser preenchidas são:

- O *Título* do documento, que deve ser claro e compreensível;
- Os *Recursos* aos quais o documento deve estar relacionado;
- Um *Resumo* no documento;
- As datas de *Criação/Publicação/Revisão* que definem o período de tempo coberto pelo documento;
- As *Palavras-chave*, que devem ser escolhidas dentro da lista disponível. O contribuinte pesquisa as palavras-chave disponíveis clicando na barra de busca, ou no logótipo da pasta que representa, ou digitando as primeiras letras da palavra desejada. As palavras-chave devem ser relevantes para o documento importado;
- A *Categoria* à qual o documento pertence;
- O *Grupo* ao qual o documento está vinculado.

Assim que todos os campos estiverem preenchidos, clique no botão azul *Next >>* no canto inferior direito da página.

Nota: Quando um documento está vinculado a alguns recursos, pode ver essa ligação na *Página de Recursos*.

Metadata for My New Document

Edit
Settings

Mandatory
Mandatory
Optional

1

Basic Metadata

Thumbnail

no image

Choose file

2

Location and Licenses

Title

My New Document

Link to

Select an option

Abstract

File Edit View Insert Format Tools

Table Help

↶ ↷ B I U ↺ ...

No abstract provided

P 3 WORDS POWERED BY TINY

3

Optional Metadata

Free-text Keywords

Date type
Date

Publication
2020-11-1

Category

Group

Completeness

Check Schema mandatory fields

46 %

Update
Next >>

Fig. 58: Metadados Básicos do Documento

50

Capítulo 1. Índice


[Data](#)
[Maps](#)
[About](#)

John Smith



[Download Layer](#)
[Metadata Detail](#)
[Editing Tools](#)
[View Layer](#)
[Download Metadata](#)

Legend

A dark orange polygon style

- dark orange polygon

Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

[Create a Map](#)

Add the layer to an existing map

Click the button below to add the layer to the selected map.

[Add to Map](#)

Documents related to this layer

List of documents related to this layer:

[old_italian_boundaries.jpg](#)

Styles

The following styles are associated with this layer. Choose a style to view it in the preview map.

⊙ (default style) A dark orange polygon style

Refresh Attributes and Statistics of this layer

Click the button below to allow GeoNode refreshing the list of available Layer Attributes. If the option 'WPS_ENABLED' has been also set on the backend, it will recalculate their statistics too.

[Refresh Attributes and Statistics](#)

Clear the Server Cache of this layer

Click the button below to wipe the tile-cache of this layer.

[Empty Tiled Layer Cache](#)

Info **Attributes** **Share** **Ratings** **Comments** **Favorite**

Title Com2016_WGS84_g
License Not Specified
Abstract No abstract provided
Publication Date June 25, 2019, 5:28 a.m.
Type Vector Data
Keywords Com2016_WGS84_g, features
Regions Global
Owner johnsmith

[More info](#)

[Layer WMS GetCapabilities document](#)

Fig. 59: Documentos vinculados a uma Camada

Também estará visível na página *Informação do Documento*.

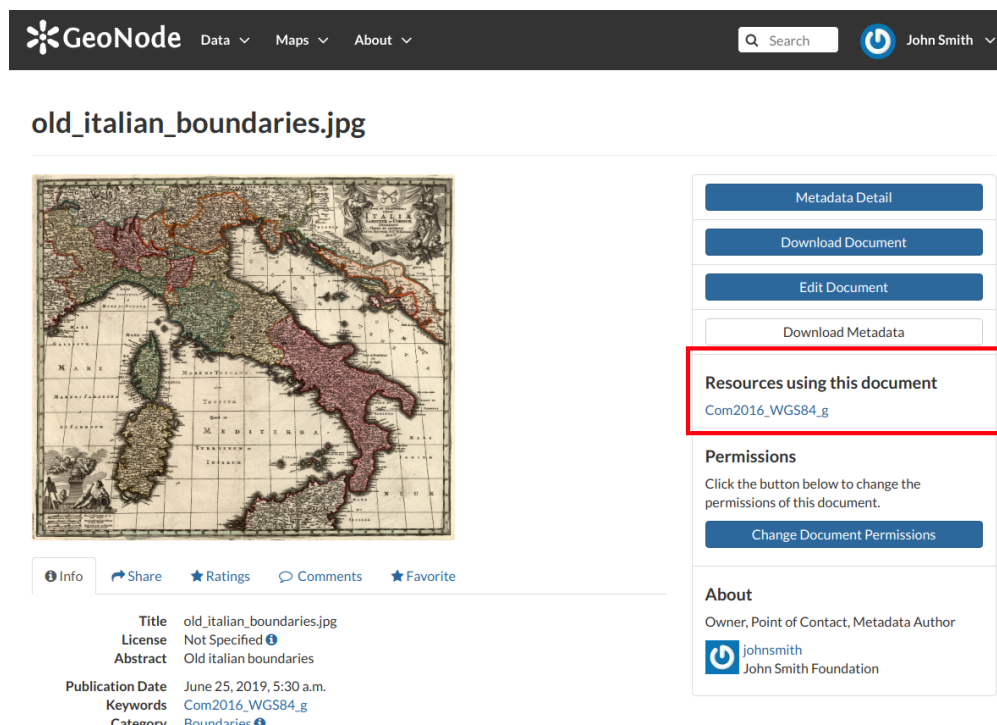


Fig. 60: Recursos vinculados a um Documento

2. Na página **Local e Licenças**, as seguintes informações devem ser preenchidas:

- O *Idioma* do documento;
- As *Regiões*, que informam sobre a extensão espacial coberta pelo documento. As extensões propostas cobrem as seguintes escalas: global, continental, regional, nacional;
- A *Declaração de Qualidade de Dados* (explicação geral do conhecimento do produtor de dados sobre a linhagem de um conjunto de dados);
- Potenciais *Restrições* à partilha do documento devem ser fornecidas na caixa Restrições.

Clique no botão azul *Next >>* para avançar para a próxima etapa.

3. Na página **Metadados Opcionais**, informações complementares podem ser adicionadas:

- A *Edição* para indicar a referência ou a fonte do documento;
- O *Propósito* do documento e seus objectivos;
- Qualquer *Informação suplementar* que possa fornecer uma melhor compreensão do documento carregado;
- A *Frequência de manutenção* do documento;
- O *Tipo de representação espacial* usado.

Partes Responsáveis, *Proprietário* e *Permissões* estão listados no lado direito da página, você pode editá-los.

Se todas as informações obrigatórias forem preenchidas, o documento pode ser publicado, caso contrário a barra de progresso *Completeness* avisa que algo está faltando.

Metadata for My New Document

Edit
Settings

Mandatory
Mandatory
Optional

1
2
3

Basic Metadata
Location and Licenses
Optional Metadata

Language
English
License
Not Specified
DOI
* Field declared Mandatory by the Metadata Schema
Attribution
* Field declared Mandatory by the Metadata Schema

Regions
Global
Data quality statement
File Edit View Insert Format Tools
Table Help
P 0 WORDS POWERED BY TINY
* Field declared Mandatory by the Metadata Schema

Restrictions
* Field declared Mandatory by the Metadata Schema
Other constraints
File Edit View Insert Format Tools
Table Help
P 0 WORDS POWERED BY TINY

<< Back
Update
Next >>

Fig. 61: Localização do Documento e Licenças

1.10. Manual de Utilizador do GeoNode

53

Metadata for My New Document

Completeness

✖ Check Schema mandatory fields

46 %

Edit

Settings

Mandatory

Mandatory

Optional

1

Basic Metadata

Other, Optional, Metadata

Edition

Purpose

File Edit View Insert Format Tools

Table Help

↶ ↷ B I U S ...

P 0 WORDS POWERED BY TINY

Supplemental information

File Edit View Insert Format Tools

Table Help

↶ ↷ B I U S ...

No information provided

2

Location and Licenses

temporal extent start

temporal extent end

Maintenance frequency

Spatial representation type

3

Optional Metadata

Responsible Parties

Point of Contact

admin x

Responsible and Permissions

Owner

admin x

Metadata Author

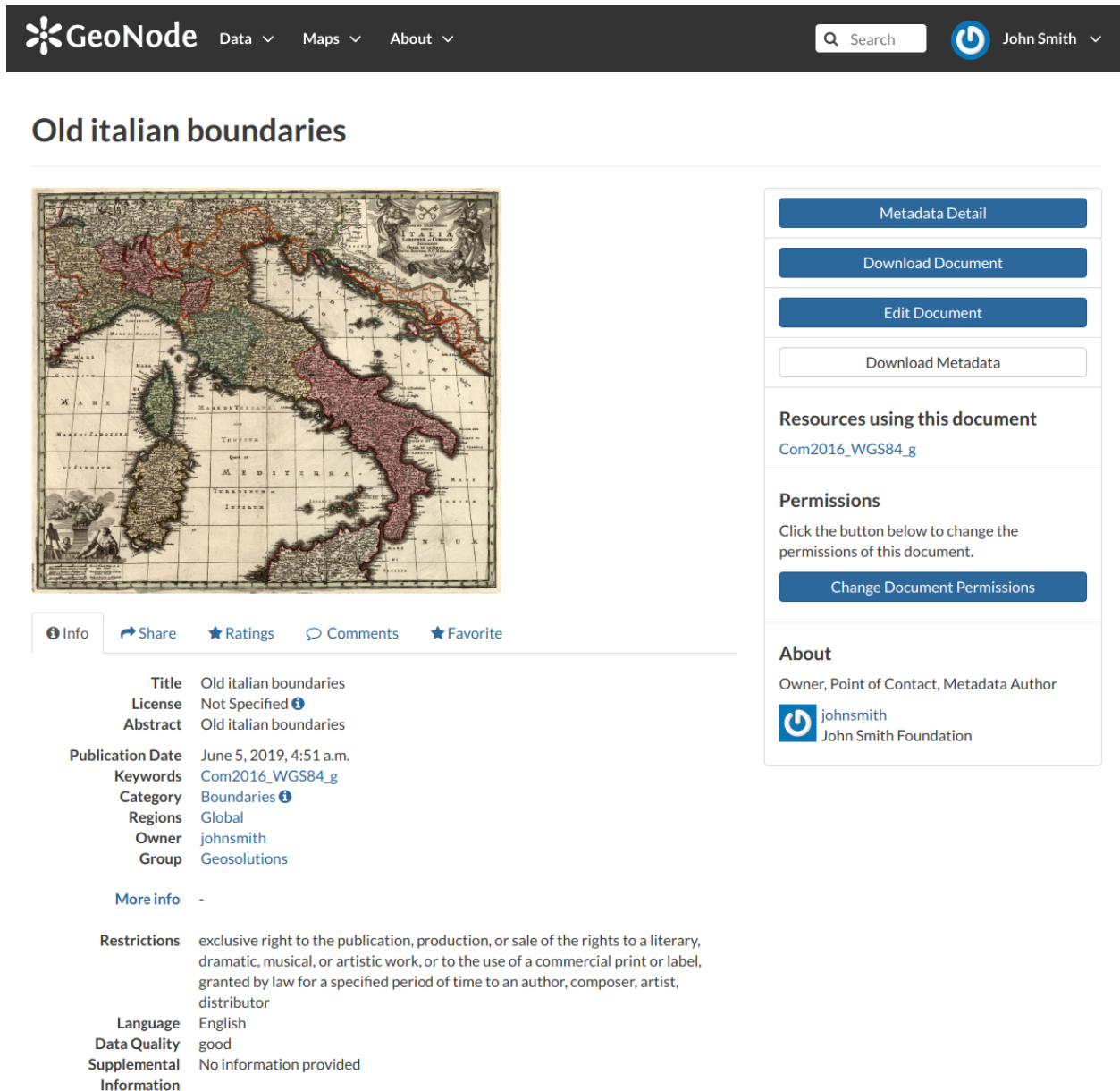
admin x

Fig. 62: Metadados Opcionais do Documento

Clique no botão azul *Update* para guardar as informações no sistema.

Informação do Documento

Na *Página de Pesquisa de Documentos* (veja [Documentos](#)), pode seleccionar o documento no qual está interessado e ver algumas informações básicas sobre ele. Pode aceder a página de detalhes do documento clicando no seu nome. A página é parecida com a mostrada na imagem em baixo.



The screenshot shows the GeoNode interface for a document titled "Old italian boundaries". The top navigation bar includes the GeoNode logo, links for Data, Maps, and About, a search bar, and a user profile for John Smith. The document title "Old italian boundaries" is prominently displayed. Below the title is a thumbnail image of a historical map of Italy. To the right of the map is a sidebar with several action buttons: "Metadata Detail", "Download Document", "Edit Document", and "Download Metadata". Below these buttons are sections for "Resources using this document" (linking to "Com2016_WGS84_g"), "Permissions" (with a "Change Document Permissions" button), and "About" (listing the owner as John Smith and the metadata author as John Smith Foundation). Below the map, there is a row of icons for Info, Share, Ratings, Comments, and Favorite. The "Info" icon is selected, leading to a detailed metadata section. This section includes fields for Title, License, Abstract, Publication Date, Keywords, Category, Regions, Owner, and Group. Below this is a "More info" link. At the bottom, there is a "Restrictions" section with a detailed text description, and "Language", "Data Quality", and "Supplemental Information" sections.

GeoNode Data ▾ Maps ▾ About ▾ Search John Smith ▾

Old italian boundaries

Metadata Detail
Download Document
Edit Document
Download Metadata

Resources using this document
[Com2016_WGS84_g](#)

Permissions
Click the button below to change the permissions of this document.
Change Document Permissions

About
Owner, Point of Contact, Metadata Author
 johnsmith
John Smith Foundation

Info Share ★ Ratings Comments ★ Favorite

Title Old italian boundaries
License Not Specified ⓘ
Abstract Old italian boundaries
Publication Date June 5, 2019, 4:51 a.m.
Keywords [Com2016_WGS84_g](#)
Category [Boundaries](#) ⓘ
Regions [Global](#)
Owner [johnsmith](#)
Group [Geosolutions](#)

More info -

Restrictions exclusive right to the publication, production, or sale of the rights to a literary, dramatic, musical, or artistic work, or to the use of a commercial print or label, granted by law for a specified period of time to an author, composer, artist, distributor
Language English
Data Quality good
Supplemental Information No information provided

Fig. 63: *Página de Informações do documento*

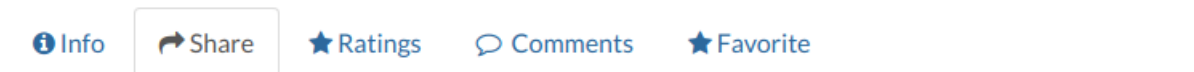
Na página de um documento, o recurso é exibido directamente na página ou acessível clicando no link fornecido no título.

Explorando as Secções das Guias

Há uma *Secção de Guia* em baixo no documento, onde pode primeiro ver *Informações* sobre o documento.

A secção **Guia Info** mostra os metadados do documento, como título, resumo, data de publicação, etc. Os metadados também indicam o utilizador responsável por enviar e fazer a gestão desse conteúdo, bem como o grupo ao qual ele está vinculado .

A **Guia Share** fornece as ligações para as redes sociais para partilhar o documento. Também existe uma ligação para partilhar o documento por email.



Share This Document

- [Email](#)
- [Facebook](#)
- [Twitter](#)
- [Google +](#)

Fig. 64: *Partilha de Documentos*

Pode **Classificar** o documento por intermédio do *Sistema de Classificações*.

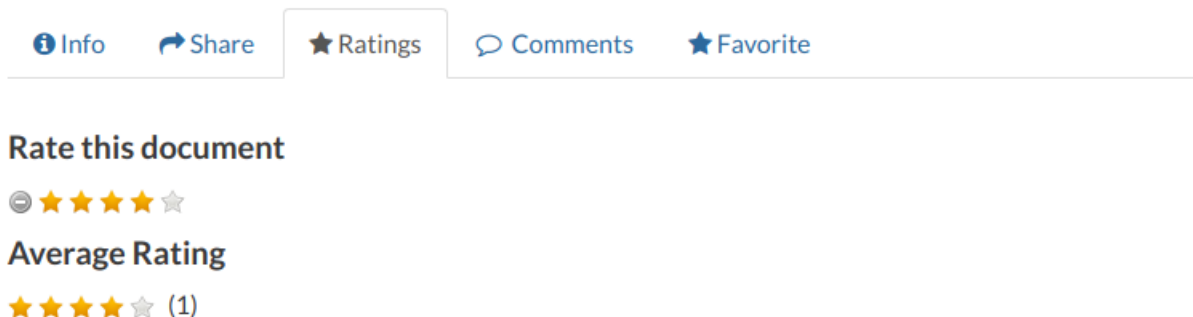


Fig. 65: *Classifique o Documento*

Na secção **Guia Comentários** pode colocar o seu comentário. Clique em *Add Comment*, insira o seu comentário e clique em *Submit Comment* para o colocar.

O seu comentário será adicionado depois do último comentário já existente. Se desejar removê-lo, clique no botão vermelho *Delete*.

Se deseja este documento nos seus *Favoritos* (veja *Actualização do Perfil*), abra a **Guia Favoritos** e clique em *Add to Favorites*.

O GeoNode também suporta o *EXIF* (*EXchangeable Image Format*) para documentos de imagem *jpeg* e *tiff*. O *EXIF* significa que informações adicionais (metadados) são armazenadas no ficheiro da imagem, então o GeoNode permite que veja essas informações na **Guia EXIF**.

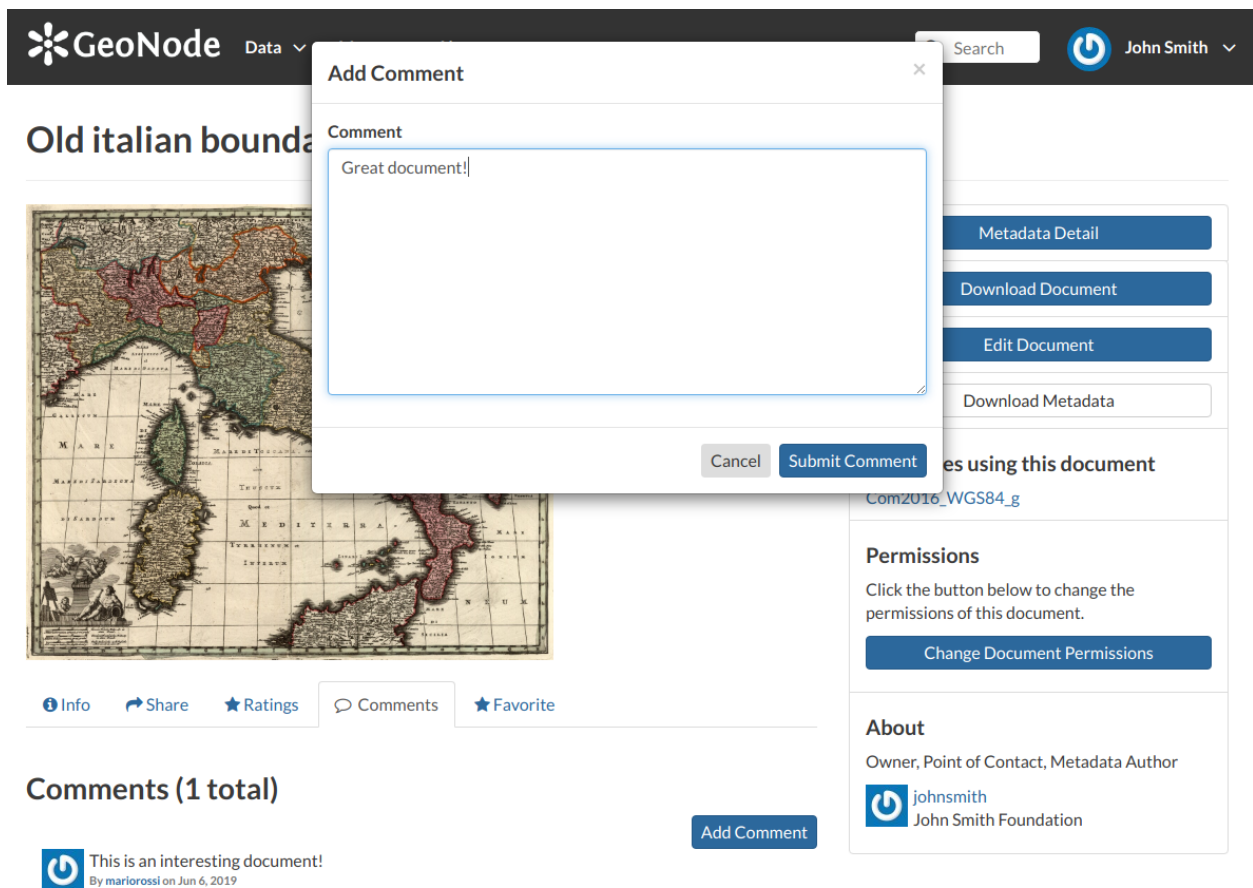


Fig. 66: Comentários do Documento

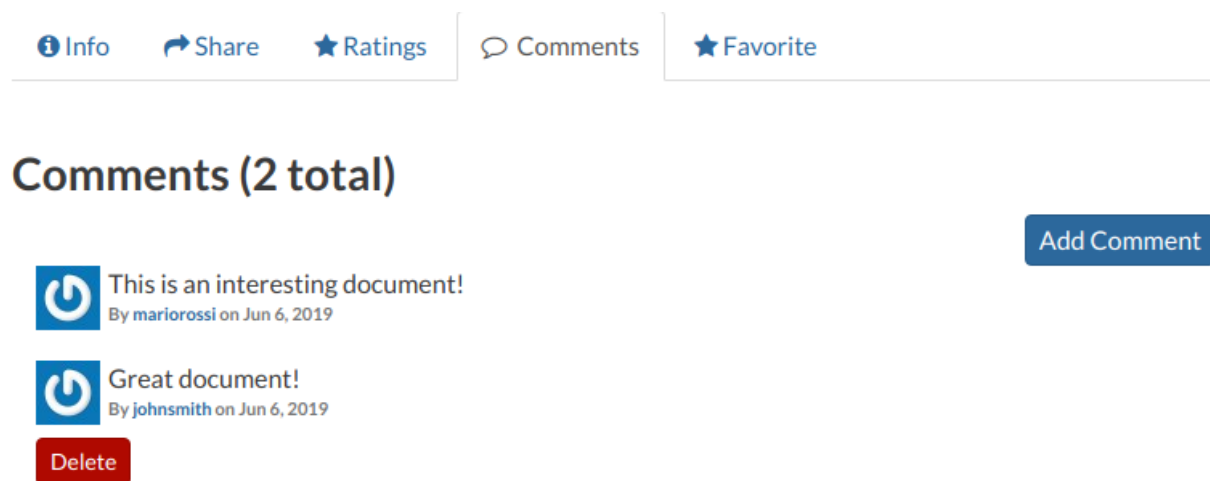


Fig. 67: O seu Comentário

[Info](#)
[Share](#)
[★ Ratings](#)
[Comments](#)
[★ Favorite](#)

Favorite

Add to Favorites

Go to Favorites

Fig. 68: O seu Comentário Favorito


[Data](#)
[Maps](#)
[About](#)

 John Smith


[Info](#)
[Share](#)
[★ Ratings](#)
[Comments](#)
[Exif](#)
[★ Favorite](#)

Metadata Detail

Download Document

Edit Document

Download Metadata

Resources using this document

This document is not related to any maps or layers

Permissions

Click the button below to change the permissions of this document.

Change Document Permissions

About

Owner, Point of Contact, Metadata Author


 johnsmith
 John Smith Foundation

Fig. 69: A guia EXIF

A Secção de Ferramentas

No lado direito da *Página do Documento* poderá ver outras informações úteis, como os links para os recursos vinculados ao documento, o documento *Proprietário*, o *Ponto de Contacto* e o *Autor de Metadados*.

Na mesma secção da *Página do Documento*, pode encontrar a seguinte ferramenta útil:

- *Metadata Detail* para explorar em detalhes os metadados do documento (veja o próximo parágrafo)
- *Download Document* para descarregar o documento
- *Edit Document* para alterar os metadados do documento, substituir o arquivo etc (veja *Edição de Documentos*)
- *Download Metadata* para descarregar todo o conjunto de metadados em vários formatos
- *Change Document Permissions* para atribuir permissões no documento a utilizadores e grupos (veja *Alterar as Permissões do Documento*).

Explorando Detalhes de Metadados

Ao clicar no botão *Metadata Detail* a *Página de Detalhes de Metadados* será aberta.

Exibe todo o conjunto de metadados disponíveis sobre o documento.

Os metadados são agrupados para mostrar os seguintes tipos de informações:

- *Identificação* para identificar em exclusivo o documento
- *Proprietário*, o utilizador que é proprietário do documento
- *Informações*, a imagem de identificação, a extensão espacial, sistema de projecção e assim por diante
- *Recursos*, restrições, Idioma e assim por diante
- *Pontos de Contacto*, o utilizador disponível para ter contacto
- *Referências*, várias ligações para as informações do recurso
- *Autor de Metadados*, as informações do autor de metadados

Edição de Documentos

A página *Informação do Documento* disponibiliza ferramentas úteis para a edição de documentos. Clique no botão *Edit Document* para ver o que pode fazer para efectuar alterações. A imagem em baixo mostra o *Painel de Edição* que aparecerá no ecrã.

Pode *Substituir* o ficheiro por outro clicando em *Replace*. Isso levá-lo-á à página *Carregar documentos* (consulte *Carregando Documentos*) onde pode carregar um novo arquivo.

O botão *Remove* permite apagar o documento. Terá que confirmar essa escolha.

O *Painel de Edição* mostra também alguns links para editar os metadados e a miniatura. Estas acções serão explicadas com mais detalhe nos próximos parágrafos.

[Metadata Detail](#)

[Download Document](#)

[Edit Document](#)

[Download Metadata](#)

Resources using this document

[Com2016_WGS84_g](#)

Permissions

Click the button below to change the permissions of this document.

[Change Document Permissions](#)

About

Owner, Point of Contact, Metadata Author

 **johnsmith**
John Smith Foundation

Fig. 70: Documentar ferramenta útil

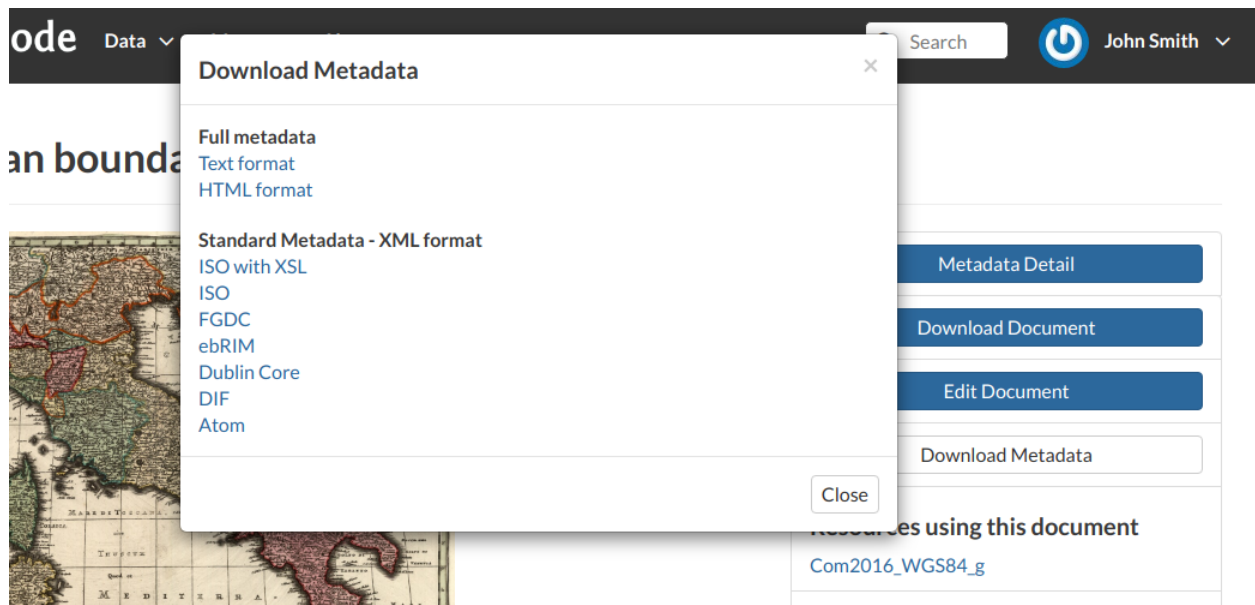


Fig. 71: Descarregar Metadados dos Documentos

Configuração da Miniatura do Documento

Do *Painel de Edição*, é também possível *Definir a Miniatura* do documento. Clique em *Set* para abrir a página *Carregamento de Miniaturas* e escolha a imagem que irá ilustrar o seu documento. Pode arrastar e soltar na caixa *Solte os arquivos aqui* ou seleccionar nas suas pastas clicando em *Escolher arquivos*. Feito isto, clique no botão vermelho *Upload files*. Se a miniatura foi carregada com sucesso, pode vê-la voltando para a lista de documentos. Clique no botão *Explore Documents* para verificar.

Se nenhum erro ocorrer, a seguinte mensagem será exibida.

Edição dos Metadados do Documento

Pode editar os metadados do seu documento através dos botões mostrados no rectângulo vermelho na imagem abaixo.

O botão *Wizard* o leva ao assistente descrito na secção *Preenchendo os Metadados do Documento*. O botão *Advanced Edit* leva-o a um formulário extenso onde todos os metadados disponíveis do documento podem ser editados. Algumas informações são obrigatórias, como o *Título* ou a *Categoria* a que o documento pertence, outras são opcionais.

No exemplo mostrado na imagem em cima, as informações dentro dos rectângulos vermelhos foram alteradas. Para salvar as alterações clique em *Update*, e será redireccionado para a página do documento.


Data ▾
Maps ▾
About ▾

John Smith ▾

Fig. 72: Página de Detalhes de Metadados do Documento

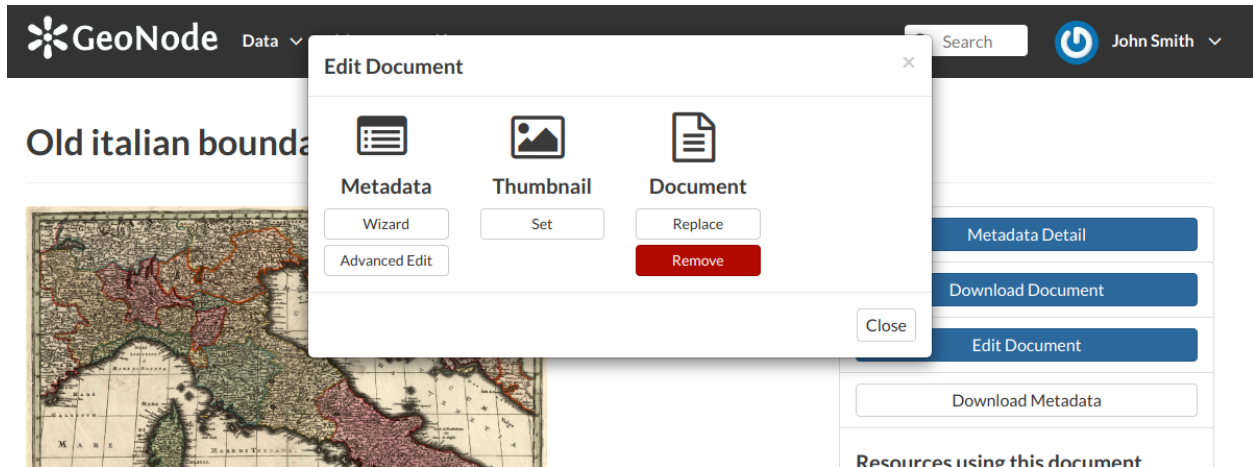


Fig. 73: Painel de Edição de documentos

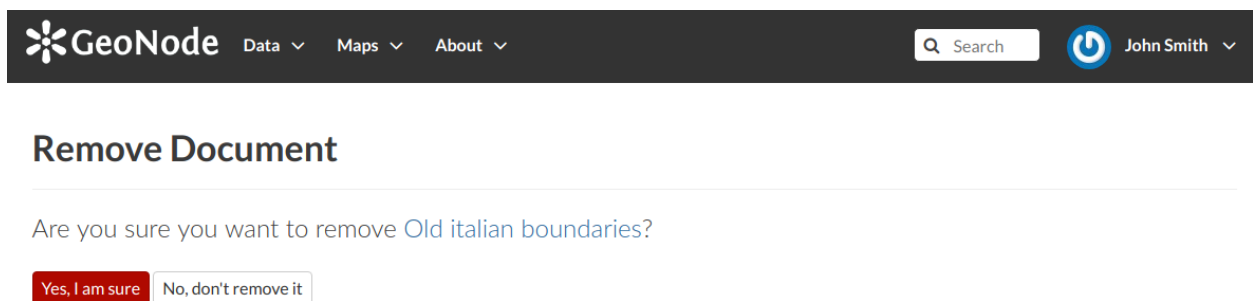


Fig. 74: Confirmação de Remoção de documento



[Data](#) [Maps](#) [About](#)

 [John Smith](#)

Upload Document's Thumbnail

[Explore Documents](#)



Drop files here

or select them one by one:

Choose Files

Files to be uploaded

420884656472

JPEG

420884656472.jpg

[Remove](#)

Clear

Upload files

Permissions

Who can view it?

☒ Anyone

The following users:

The following groups:

Who can download it?

Who can change metadata for it?

Who can manage it? (update, delete, change permissions, publish/unpublish it)

Fig. 75: Carregar Miniatura do Documento

Your document was successfully updated

Document Info

Edit Metadata

Fig. 76: Carregamento bem-sucedido

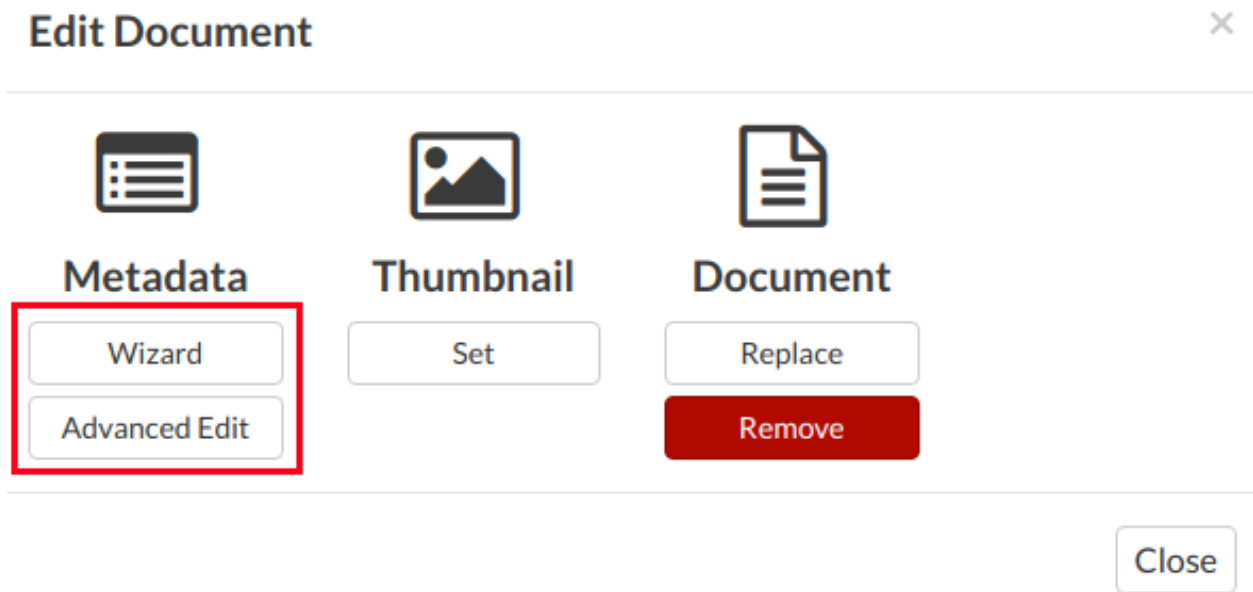


Fig. 77: Edição de Metadados

Alterar as Permissões do Documento

O GeoNode incentiva a publicar, partilhar e disponibilizar para descarregamento as informações carregadas na plataforma. Por padrão, qualquer pessoa pode ver e descarregar um documento. No entanto, o responsável pelo documento pode optar por limitar o acesso ao documento a alguns colaboradores e/ou grupos.

Através do botão mostrado na figura em baixo é possível fazer a gestão das permissões do documento.

O botão *Change Document Permissions* no lado direito da página do documento permite definir quem pode:

- Ver o documento;
- Descarregá-lo;
- Editar os metadados;
- Fazer a gestão (actualizar, excluir, alterar permissões, publicar/cancelar publicação).

Veja um exemplo na imagem em baixo.

Normalmente, a edição de metadados e a gestão de um documento ficam a cargo do responsável pelo documento, ou seja, o colaborador que o carregou e quem tem essas permissões por defeito.

Depois de definir as permissões, clique em *Apply changes* para guardá-las.

[illegible]

 Data ▾ Maps ▾ About ▾

Search

John Smith ▾

Nouvelle Carte de l'Italie



[Info](#) [Share](#) [Ratings](#) [Comments](#) [Favorite](#)

Title	Nouvelle Carte de l'Italie
License	Not Specified ⓘ
Abstract	A map from the Moll's map collection
Publication Date	June 5, 2019, 4:51 a.m.
Keywords	Com2016_WGS84_g
Category	Boundaries ⓘ
Regions	Global , Italy
Owner	johnsmith

[Metadata Detail](#)
[Download Document](#)
[Edit Document](#)
[Download Metadata](#)

Resources using this document
[Com2016_WGS84_g](#)

Permissions
Click the button below to change the permissions of this document.
[Change Document Permissions](#)

About
Owner, Point of Contact, Metadata Author
 johnsmith
John Smith Foundation

Fig. 79: O botão para alterar as permissões

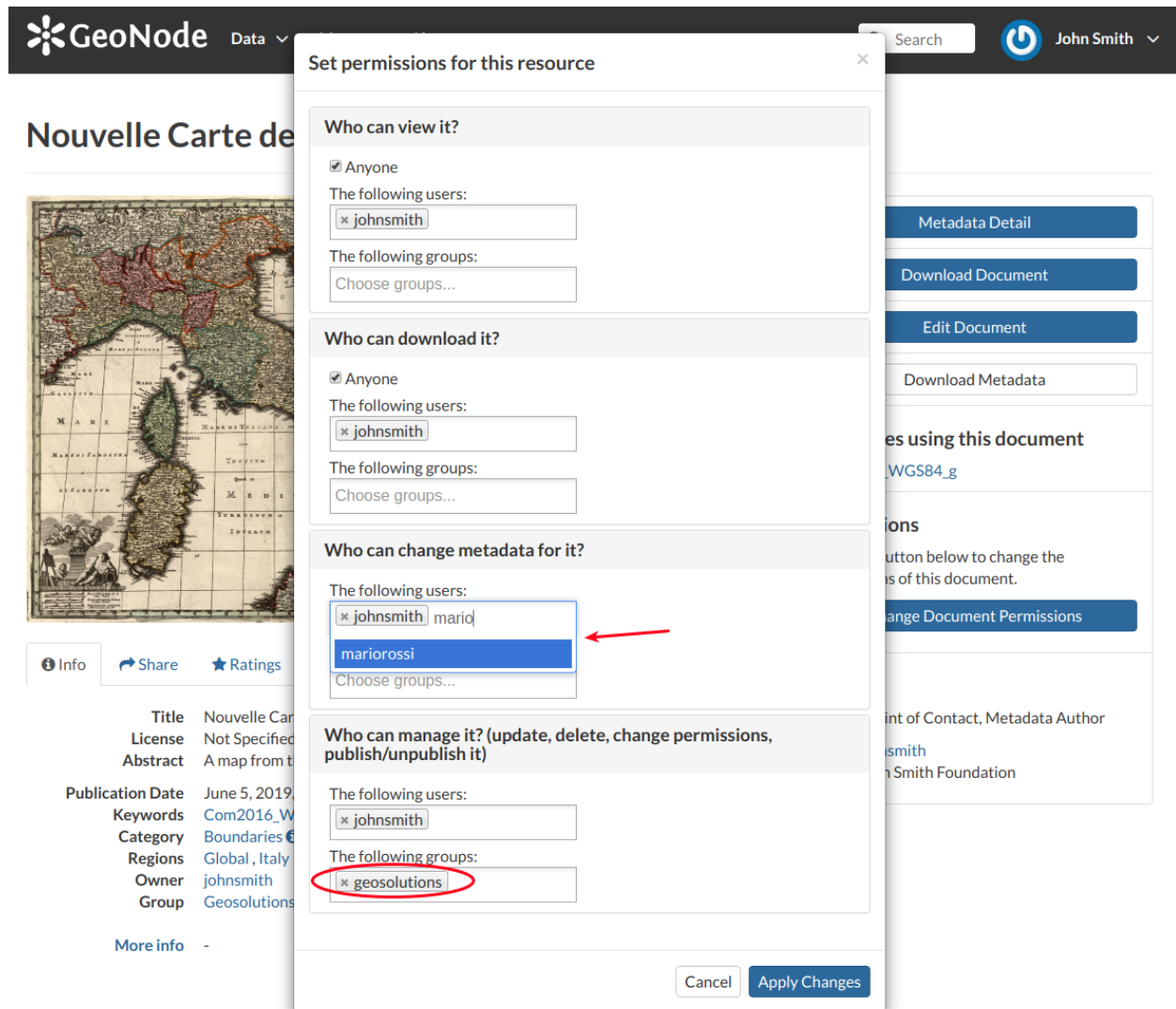


Fig. 80: Alterando as Permissões do documento

1.10.5 Fazendo a Gestão de Camadas

Camadas são recursos publicados que representam fontes de dados espaciais raster ou vectoriais. As camadas também podem ser associadas a metadados, classificações e comentários.

Nesta secção, irá aprender como criar uma nova camada carregando um conjunto de dados local, adicionar informações de camada, alterar o estilo da camada e partilhar os resultados.

Carregamento de Camadas

O tipo de recurso mais importante no GeoNode é a *Camada*. Uma camada representa informações espaciais para que possam ser exibidas num mapa.

Para entender melhor do que estamos a falar, vamos carregar a sua primeira camada.

A página de *Carregamento de Camada* pode ser acedida a partir do link *Upload Layer* do menu *Data* na barra de navegação.

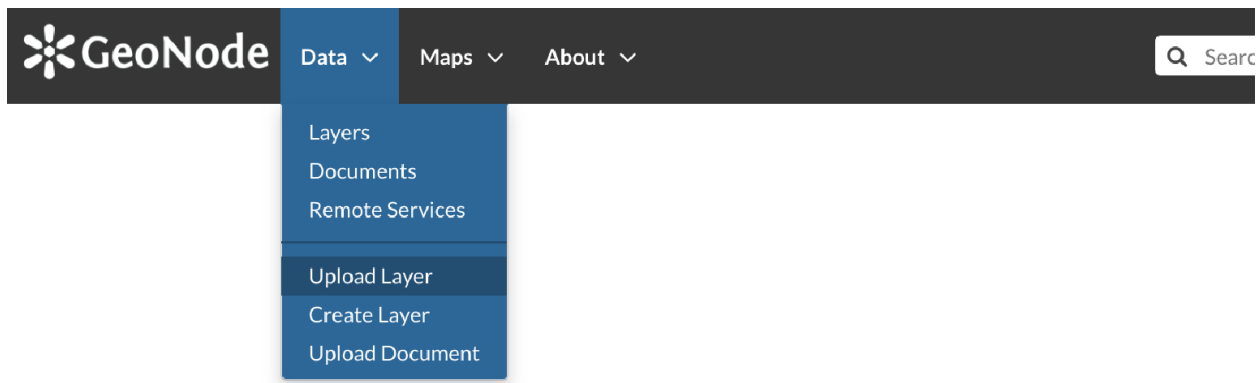


Fig. 81: Ligação para Carregar Camadas

Há também um botão *Upload Layers* na *Página de Camadas*.

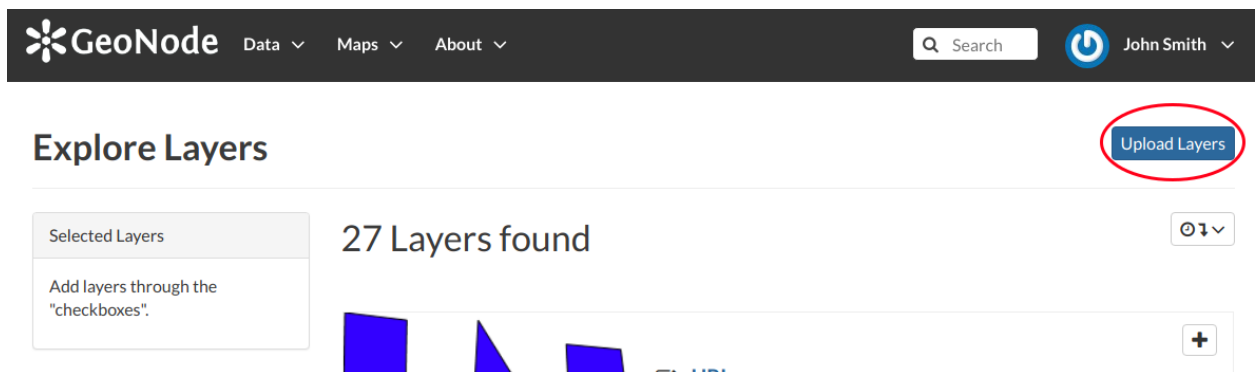


Fig. 82: Botão para Carregamento de Camadas

A página *Carregamento de Camadas* parece-se com a da imagem em baixo.

Através do botão *Choose Files* pode seleccionar ficheiros do seu disco, certifique-se de que são dados espaciais raster

Fig. 83: A página de Carregamento de Camadas

ou vectores válidos. Pode também, alterar as definições de *Permissões* predefinidas (veja [Alterando as Permissões da Camada](#) para obter mais informações sobre como definir permissões).

Selecione o *charset* e clique em *Upload files* para iniciar o processo ou clique em *Clear* para remover todos os arquivos carregados da página.

Fig. 84: Carregamento de Shapefile

Neste exemplo foi escolhido o Shapefile ESRI *roads*, com todos os seus arquivos obrigatórios (*.shp*, *.shx*, *.dbf* e *.prj*). Uma barra de progresso mostra a operação feita durante o carregamento da camada e alerta quando o processo termina. Quando o processo terminar, clique em *Layer Info* para verificar se a camada foi carregada correctamente.

Nota: Existem muitos conjuntos de dados espaciais gratuitos disponíveis na Internet. Neste exemplo foi utilizado um extracto do mapa das estradas do centro da cidade de Berlim do conjunto de dados [BBBike extract](#) [OpenStreetMap](#).

Nos próximos parágrafos aprenderá a criar uma camada do nada, como definir permissões, como explorar as propriedades da camada e como editá-las.

Criação de uma camada do nada

Uma ferramenta interessante que o GeoNode disponibiliza para si é a *Criar Camada*. Permite criar uma nova camada vectorial do nada. O *Formulário de Criação de Camada* é alcançável através do link *Create Layer* mostrado na imagem em baixo.

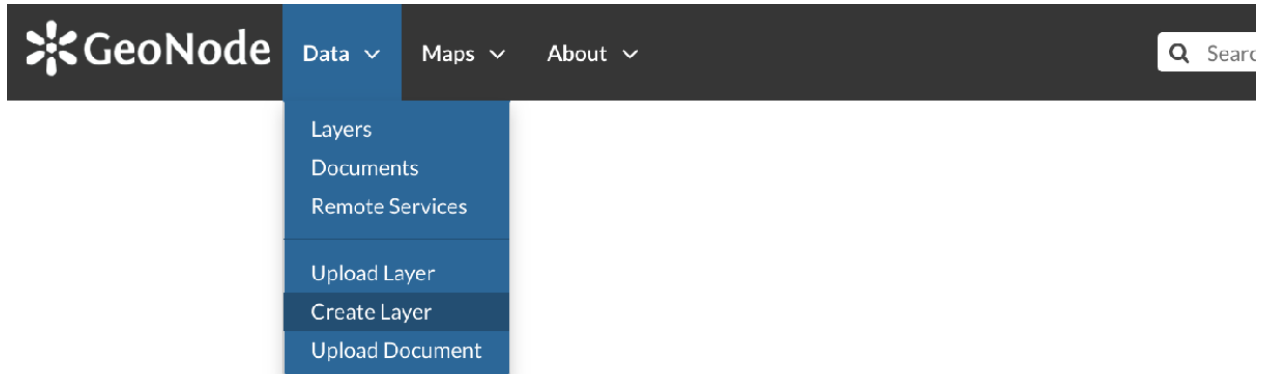


Fig. 85: Ligação criar camada

Para criar a nova Camada tem de preencher os campos necessários:

- *Nome*
- *Título*
- *Tipo de geometria*

Geometry type

Points	▼
Points	
Lines	
Polygons	

Fig. 86: Tipos de geometria

Normalmente, as características das camadas devem ter alguns *Atributos* que enriquecem a quantidade de informação associada a cada uma delas. Através do botão *Add Attribute* pode adicionar novos atributos.

Fig. 87: Criação de Camada Nova do zero

Neste momento pode também alterar as configurações padrão de *Permissões*, consulte permissões-de-camada para saber como.

Uma vez preenchido o formulário, clique em *Create*. Será redireccionado para a *Página de Camadas* (veja [Informação da Camada](#)). Agora a sua camada foi criada, mas ainda está vazia, nenhum recurso foi adicionado ainda. Veja a secção [Edição de Camada](#) para aprender como adicionar novos recursos.

Usando Serviços Remotos

No GeoNode pode adicionar novas camadas não apenas carregando-as do seu disco, mas também usando *Serviços Remotos*. Nesta secção irá aprender como adicionar um novo serviço e como carregar recursos no GeoNode através deles.

Vamos experimentar!

Clique na ligação *Remote Services* do menu *Data* na barra de navegação.

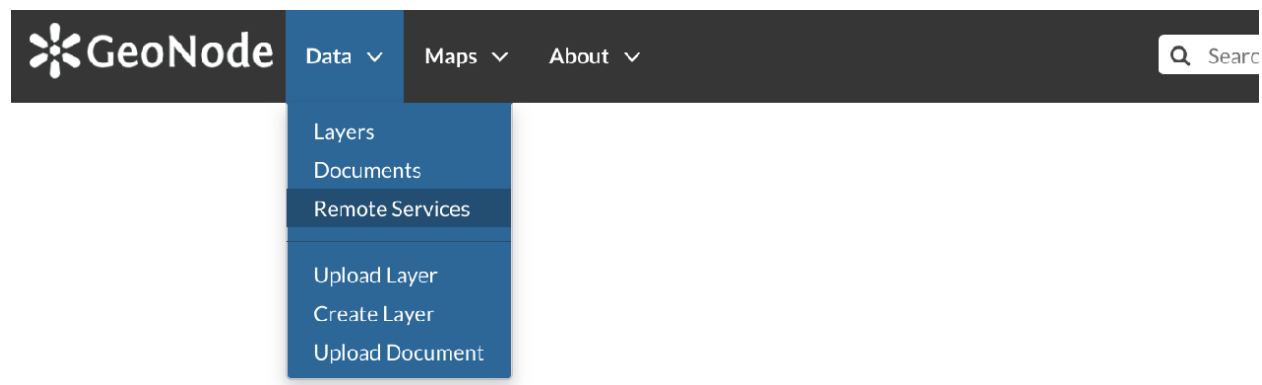


Fig. 88: *Ligação de Serviços Remotos*

A página que se abre irá conter a lista dos serviços disponíveis.

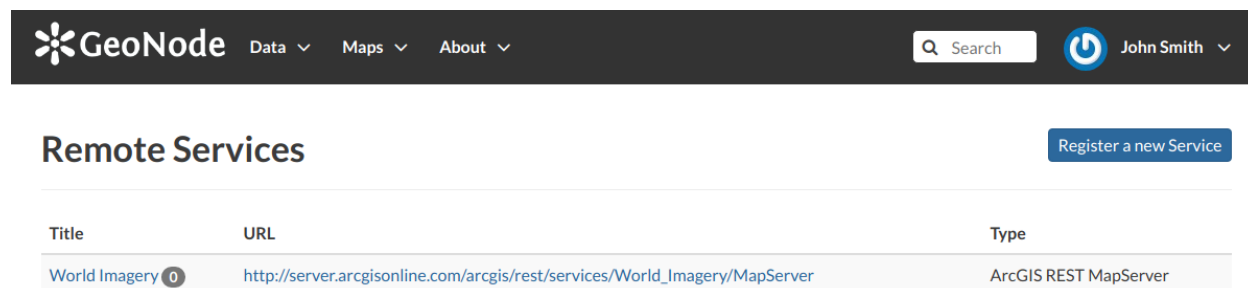


Fig. 89: *Serviços Remotos*

Para configurar um novo serviço:

- clique em *Register a new Service*
- digite o *URL do Serviço*
- seleccione o *Tipo de Serviço*

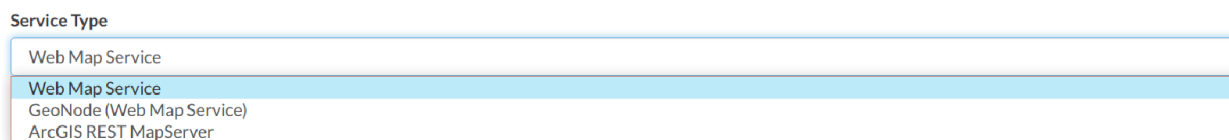


Fig. 90: *Tipos de Serviço*

- clique em *Create*

O GeoNode oferece suporte a três **tipos de serviços remotos**:

- *Serviço Web de Mapas*

Serviço Web Genérico de Mapas (WMS) baseado em um protocolo padrão para servir imagens de mapas georreferenciados pela Internet. Essas imagens são normalmente produzidas por um servidor de mapas (como [GeoServer](#)) a partir de dados fornecidos por um ou mais bancos de dados geoespaciais distribuídos. As operações comuns realizadas por um serviço WMS são: *GetCapabilities* (para recuperar metadados sobre o serviço, incluindo operações e parâmetros suportados e uma lista das camadas disponíveis) e *GetMap* (para recuperar uma imagem de mapa para uma área e conteúdo especificados)

Nota: Muitos serviços WMS estão disponíveis na internet, neste exemplo usamos o <https://demo.geo-solutions.it/geoserver/wms>.

- *Serviço Web de Mapas do GeoNode*

Geralmente, um WMS não é invocado diretamente; aplicativos cliente, como GIS-Desktop ou WEB-GIS, são usados para fornecer ao utilizador controles interactivos. Um WMS GeoNode executa automaticamente algumas operações e permite que obtenha recursos imediatamente.

Nota: Um exemplo de WMS GeoNode está disponível em <http://dev.geonode.geo-solutions.it/geoserver/wms>.

- *ArcGIS REST MapServer*

Este serviço de mapas fornece informações básicas sobre o mapa, incluindo as camadas que ele contém, se o mapa está armazenado em cache ou não, sua referência espacial, extensões iniciais e completas, se o serviço tem permissão para exportar blocos e contagem máxima de exportação de blocos etc. Um conjunto de operações que fazem a gestão do estado e do conteúdo do serviço é permitido: Editar serviço, Actualizar, Actualizar blocos. A URL deve seguir este padrão: <https://<servicecatalog-url>/serviços/<serviceName>/MapServer>.

Nota: Experimente o seguinte serviço para entender melhor como ele funciona: <https://sampleserver6.arcgisonline.com/arcgis/rest/services/USA/MapServer>.

Depois de configurar o serviço, você pode carregar os recursos de seu interesse por meio da página *Importar Recursos*, para a qual será redirecionado automaticamente. Veja a animação em baixo para ver todo o processo.

Fig. 91: *Um novo Serviço Remoto*

Na página onde estão listados os serviços, é possível clicar no *Título* de um serviço. Ele abre a página de *Detalhes do Serviço*.

Cada serviço tem seus próprios metadados, como *Tipo de Serviço*, *URL*, um *Resumo*, algumas *Palavras-chave* e o utilizador de *Contacto*. Pode editar esses metadados através do formulário disponível no botão *Edit Service Metadata* da página *Detalhes do Serviço* (veja a imagem abaixo).

The screenshot shows the GeoNode web interface. At the top is a dark navigation bar with the GeoNode logo, 'Data', 'Maps', and 'About' menus, a search bar, and a user profile for 'John Smith'. The main content area is titled 'GeoServer Web Map Service'. It lists the service type as 'Web Map Service', the URL as 'https://demo.geo-solutions.it/geoserver/wms', an abstract describing it as a compliant WMS implementation, keywords 'GEOSERVER, WFS, WMS', and contact 'johnsmith'. To the right is a 'Manage' sidebar with buttons for 'Edit Service Metadata', 'Import Service Resources', and 'Remove Service'. Below the service details is a 'Service Resources' section with a table listing two resources: 'sfдем' and 'Regioni Italiane', both with the description 'No abstract provided'.

Title	Description
sfдем	No abstract provided
Regioni Italiane	No abstract provided

Fig. 92: Metadados de Serviço Remoto

Alterando as Permissões da Camada

Ao criar ou carregar uma nova Camada, tem de definir quem pode ver, transferir, editar e gerir essa Camada. Por padrão, apenas os proprietários podem editar e gerir camadas, qualquer um pode vê-las e descarregá-las.

Para modificar as definições de *Permissões* da Camada tem de clicar no botão *Change the Layer Permissions* na página Camada.

Através do Painel de Definições de *Permissões* pode adicionar ou remover permissões para utilizadores e grupos. A imagem abaixo mostra um exemplo.

Você pode definir os seguintes tipos de permissões:

- *Visualizar* permite ver a camada;
- *Baixar* permite baixar a camada;
- *Alterar Metadados* permite alterar os metadados da camada;
- *Editar Dados* permite alterar atributos e propriedades dos recursos das camadas;
- *Editar Estilo* permite alterar o estilo da camada;
- *Gerir* permite actualizar, excluir, alterar permissões, publicar e cancelar a publicação da camada.

Aviso: Ao atribuir permissões a um grupo, todos os membros do grupo terão essas permissões. Tenha cuidado no caso de editar permissões.

Clique em *Apply Changes* para guardar essas configurações.


[Data](#)
[Maps](#)
[About](#)

 John Smith

Italian Towers



[Download Layer](#)
[Metadata Detail](#)
[Editing Tools](#)
[View Layer](#)
[Download Metadata](#)

Legend

Default Point

- Red Square

Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

[Create a Map](#)

Add the layer to an existing map

Regioni Italiane Map

Click the button below to add the layer to the selected map.

[Add to Map](#)

Styles

The following styles are associated with this layer. Choose a style to view it in the preview map.

- (default style) Default Point

Refresh Attributes and Statistics of this layer

Click the button below to allow GeoNode refreshing the list of available Layer Attributes. If the option 'WPS_ENABLED' has been also set on the backend, it will recalculate their statistics too.

[Refresh Attributes and Statistics](#)

Clear the Server Cache of this layer

Click the button below to wipe the tile-cache of this layer.

[Empty Tiled-Layer Cache](#)

Permissions

Click the button below to change the permissions of this layer.

[Change Layer Permissions](#)

About

Owner, Point of Contact, Metadata Author

 johnsmith
John Smith Foundation

Info [Attributes](#) [Share](#) [Ratings](#) [Comments](#) [Favorite](#)

Title Italian Towers
License Not Specified
Abstract No abstract provided
Publication Date June 7, 2019, 4:03 a.m.
Type Vector Data
Keywords features , italian_towers
Category Environment
Regions Global
Owner johnsmith

[More info](#)

[Layer WMS GetCapabilities document](#)

Fig. 93: Alterar Permissões de Camada

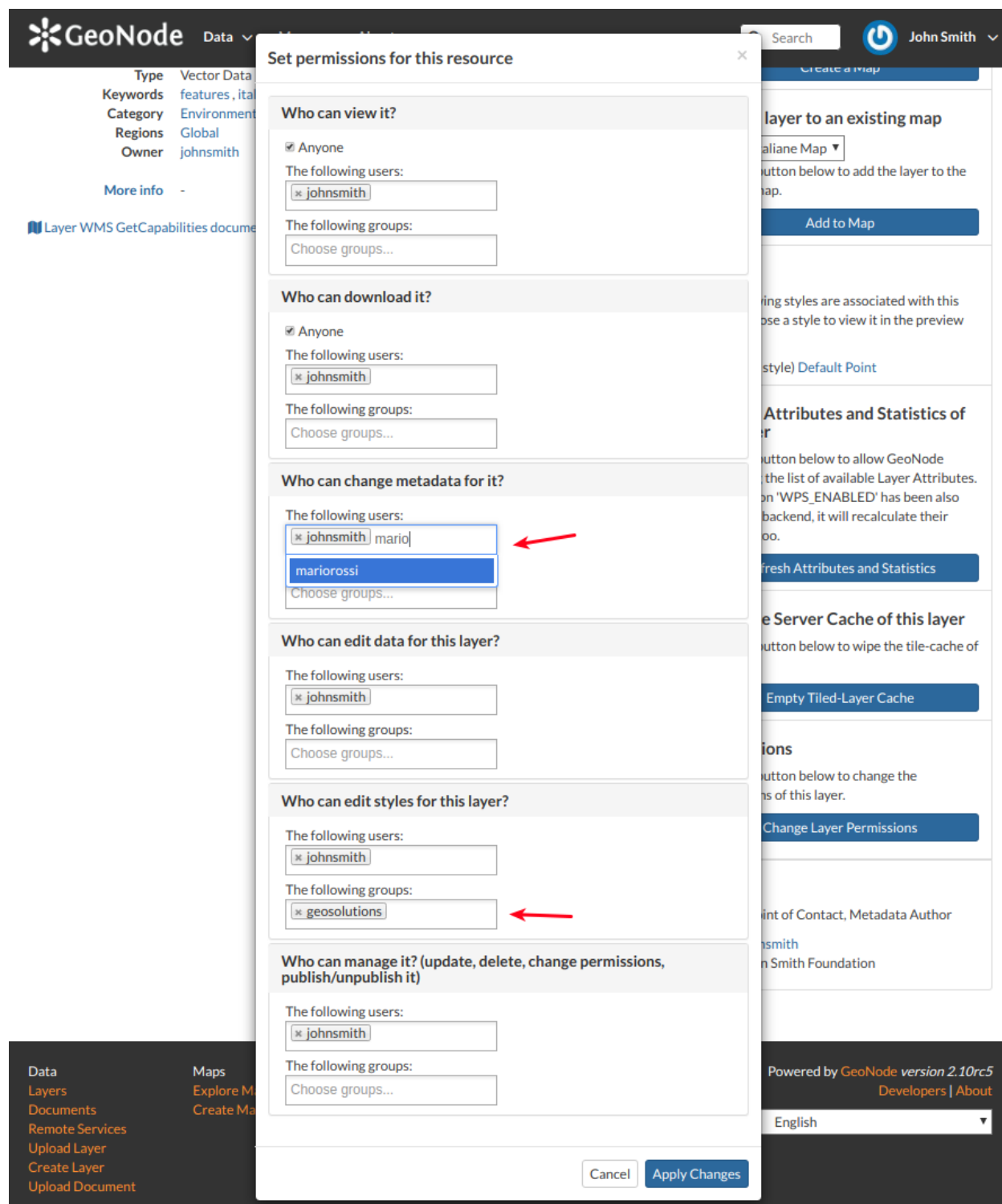


Fig. 94: Configurações de Permissões de Camada para utilizadores e grupos

Informação da Camada

Nesta secção, aprenderá mais sobre camadas. Na secção pesquisando-camadas explicamos como encontrar camadas, agora queremos ir mais a fundo mostrando como explorar informações detalhadas sobre isto.

Na página da lista de camadas, clique na camada do seu interesse. A *Página de Camadas* será aberta.

Conforme mostrado na imagem acima, a *Página de Camadas* é dividida em três secções principais:

1. a secção *Pré-visualização da Camada*, sob o título
2. a secção *Guias*, sob a pré-visualização da camada
3. a secção *Ferramentas*, no lado direito da página

Pré-visualização da Camada

A *Pré-visualização da Camada* mostra a camada num mapa com funcionalidades muito básicas:

- o *Comutador de Mapa Básico* que permite alterar o mapa básico;
- a ferramenta *Zoom in/out* para ampliar e encolher a visualização;
- a ferramenta *Zoom na extensão máxima* para que o zoom se ajuste ao tamanho da camada;
- a ferramenta *Consulta de Objectos* para recuperar informações sobre os objectos do mapa clicando no mapa;
- a ferramenta *Imprimir* para imprimir a pré-visualização.

O visualizador de mapa GeoNode é baseado em [MapStore](#), consulte a [Documentação MapStore](#) para saber mais.

Secções de Guias

A *Página da Camada* mostra algumas secções das guias contendo diferentes informações sobre a camada:

- A guia *Informações* está activa por defeito. Esta secção da guia mostra alguns metadados da camada, como título, resumo, data de publicação, etc. Os metadados também indicam o proprietário da camada, a quais categorias de tópico a camada pertence e quais regiões são afectadas.
- A guia *Atributos* mostra a estrutura de dados por trás da camada. Todos os atributos são listados e para cada um deles algumas estatísticas (por exemplo, o intervalo de valores) são estimados (se possível).
- A guia *Partilhar* fornece ligações para a camada poder ser partilhada pelas redes sociais ou email.
- Pode *Classificar* a camada através do *Sistema de Classificação*.
- Na secção da guia *Comentários*, pode postar o seu comentário. Clique em *Add Comment*, insira o seu comentário e clique em *Submit Comment* para postá-lo.

O seu comentário será adicionado depois do último comentário existente. Se desejar removê-lo, clique no botão vermelho *Delete*.

- Se desejar esta camada nos seus *Favoritos* (veja [Actualização do Perfil](#)), abra a guia *Favoritos* e clique em *Add to Favorites*.


Data ▾
Maps ▾
About ▾

 John Smith ▾

Italian Towers



+
-
...

© OpenStreetMap contributors

Info Attributes Share Ratings Comments Favorite

Title Italian Towers
License Not Specified 
Abstract No abstract provided
Publication Date June 7, 2019, 4:03 a.m.
Type Vector Data
Keywords features , italian_towers
Category Environment 
Regions Global
Owner johnsmith
[More info](#) -

 Layer WMS GetCapabilities document

Download Layer

Metadata Detail

Editing Tools

View Layer

Download Metadata

Legend

Default Point

- Red Square

Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

Create a Map

Add the layer to an existing map

Regioni Italiane Map ▾

Click the button below to add the layer to the selected map.

Add to Map

Styles

The following styles are associated with this layer. Choose a style to view it in the preview map.

☒ (default style) Default Point

Refresh Attributes and Statistics of this layer

Click the button below to allow GeoNode refreshing the list of available Layer Attributes. If the option 'WPS_ENABLED' has been also set on the backend, it will recalculate their statistics too.

Refresh Attributes and Statistics

Clear the Server Cache of this layer

Click the button below to wipe the tile-cache of this layer.

Empty Tiled-Layer Cache

Permissions

Click the button below to change the permissions of this layer.

Change Layer Permissions

About

Owner, Point of Contact, Metadata Author

 johnsmith
John Smith Foundation

Fig. 96: Pré-visualização da Camada

[Info](#)
[Attributes](#)
[Share](#)
[Ratings](#)
[Comments](#)
[Favorite](#)

Title	Regioni Italiane
License	Not Specified i
Abstract	No abstract provided
Publication Date	June 7, 2019, 4:49 a.m.
Type	Data
Keywords	features , Reg2015_WGS84_g
Category	Environment i
Regions	Global , Africa , Central Africa , North Africa , Algeria , Tunisia , Europe , Albania , Austria , Bosnia and Herzegovina , Croatia , France , Greece , Holy See (Vatican City) , Hungary , Italy , Liechtenstein , Malta , Monaco , Montenegro , San Marino , Serbia , Slovenia , Switzerland , Pacific
Owner	johnsmith
<u>More info</u>	-
Language	English
Supplemental Information	No information provided

Fig. 97: Guia Informações da Camada

 [Data](#) [Maps](#) [About](#)

 John Smith

Italian Towers



[Info](#) [Attributes](#) [Share](#) [Ratings](#) [Comments](#) [Favorite](#)

Attribute Name	Label	Description	Range	Average	Median	Standard Deviation
fid			NA			
city			NA			
name			NA			
height			NA			

[Download Layer](#)
[Metadata Detail](#)
[Editing Tools](#)
[View Layer](#)
[Download Metadata](#)

Legend
Default Point
■ Red Square

Maps using this layer
This layer is not currently used in any maps.

Create a map using this layer
Click the button below to generate a new map based on this layer.
[Create a Map](#)

Add the layer to an existing map

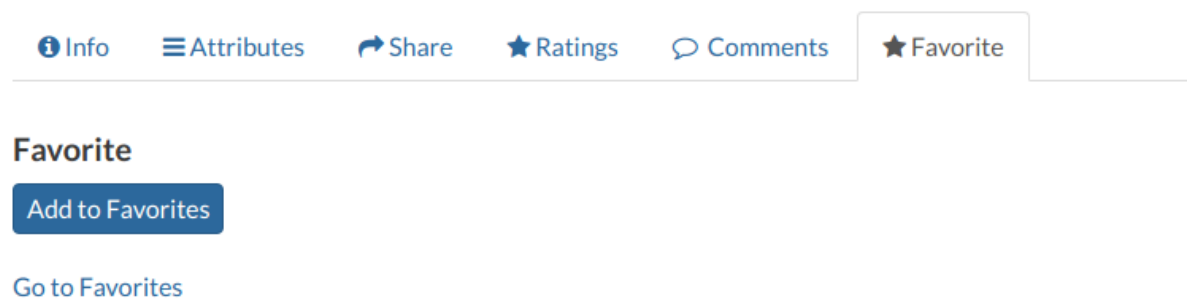
Fig. 98: Guia de Atributos da Camada

[Info](#) [Attributes](#) [Share](#) [Ratings](#) [Comments](#) [Favorite](#)

Share This Layer

- [Email](#)
- [Facebook](#)
- [Twitter](#)
- [Google +](#)

Fig. 99: Partilha de Camadas

Fig. 100: *Classifique a Camada*Fig. 101: *Camada de Comentários*Fig. 102: *Sua Camada Favorita*

Ferramentas de Camada

No lado direito da *Página de Camada*, existem alguns botões e informações que poderão ajudá-lo a fazer a gestão da camada. Este parágrafo cobrirá apenas as ferramentas que mostram informações de camadas. As *Ferramentas de Edição* serão exploradas na secção *Edição de Camada*.

- através do botão *Download Layer* pode descarregar a camada com algumas opções, veja *Descarregando Camadas*;
- o botão *Metadata Detail* para ver os metadados da camada, consulte *Metadados de Camadas* para ler mais;
- o botão *Editing Tools* permite aceder a muitas ferramentas de edição. Essas funcionalidades serão explicadas na secção *Edição de Camada*;
- o botão *View Layer* abre a camada num mapa, veja o *Informação do Mapa* para mais detalhes;
- o botão *Download Metadata* permite que você descarregue os metadados da camada em vários formatos;
- a *Legenda* mostra a que símbolos e estilos no mapa se referem;
- na secção *Mapa usando esta camada* todos os mapas que usam a camada são listados;
- em *Criar um mapa usando esta camada*, o botão *Create a Map* permite criar um mapa do nada usando a camada;
- a secção *Adicionar a camada a um mapa existente* mostra um menu suspenso no qual todos os mapas que o utilizador pode visualizar são listados. O botão *Add to Map* permite que adicionar a camada ao mapa que seleccionou no menu anterior;
- a secção *Estilos* mostra todos os estilos associados à camada. Clique na caixa de selecção correspondente a um dos estilos listados para aplicá-lo na visualização;
- na secção *Actualizar atributos e estatísticas desta camada* o *Refresh Attributes and Statistics* permite que o GeoNode actualize a lista de atributos de camada disponíveis. Se a opção “WPS_ENABLED” também for definida no backend, ele irá recalcular as estatísticas também;
- na secção *Limpar a cache do servidor desta camada* *Empty Tiled-Layer Cache* permite limpar a cache de tiles desta camada;
- a secção *Sobre* mostra a camada *Proprietário*, o utilizador de *Contacto* e o *Autor de Metadados*.

Descarregando Camadas

No topo da *Página da Camada* está o botão *Download Layer* (veja *Informação da Camada*). Ele fornece acesso à capacidade de guardar dados geoespaciais armazenados no GeoNode.

Verá uma lista de opções dos formatos de exportação suportados. Pode escolher os formatos de *Imagens* PNG, PDF, JPEG se quiser salvar uma imagem «semelhante a uma captura de ecrã» da camada.

Pode também, baixar os dados da camada, os formatos de exportação suportados serão listados na guia *Dados*. Clique no formato desejado para accionar o download.

Conforme mostrado na imagem em cima, o GeoNode permite que se descarregue um subconjunto de dados. Clique em *Do you want to filter it?* Para filtrar os dados da camada antes de descarregar.


[Data](#)
[Maps](#)
[About](#)

[Test Menu](#)

[johnsmith](#)



[Download Layer](#)
[Metadata Detail](#)
[Editing Tools](#)
[View Layer](#)
[Download Metadata](#)

Legend

Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

[Create a Map](#)

Styles

The following styles are associated with this layer. Choose a style to view it in the preview map.

- ☐ A orange line style
- ☒ (default style) Red road

Refresh Attributes and Statistics of this layer

Click the button below to allow GeoNode refreshing the list of available Layer Attributes

[Info](#)
[Attributes](#)
[Share](#)
[Ratings](#)
[Comments](#)
[Favorite](#)

Title roads

License Not Specified

Abstract No abstract provided

Publication Date June 7, 2019, 2:15 p.m.

Type Vector Data

Keywords features, ne_10m_roads

Category Transportation

Regions Global

Owner johnsmith

More info -

Fig. 103: Altere o estilo da camada na pré-visualização

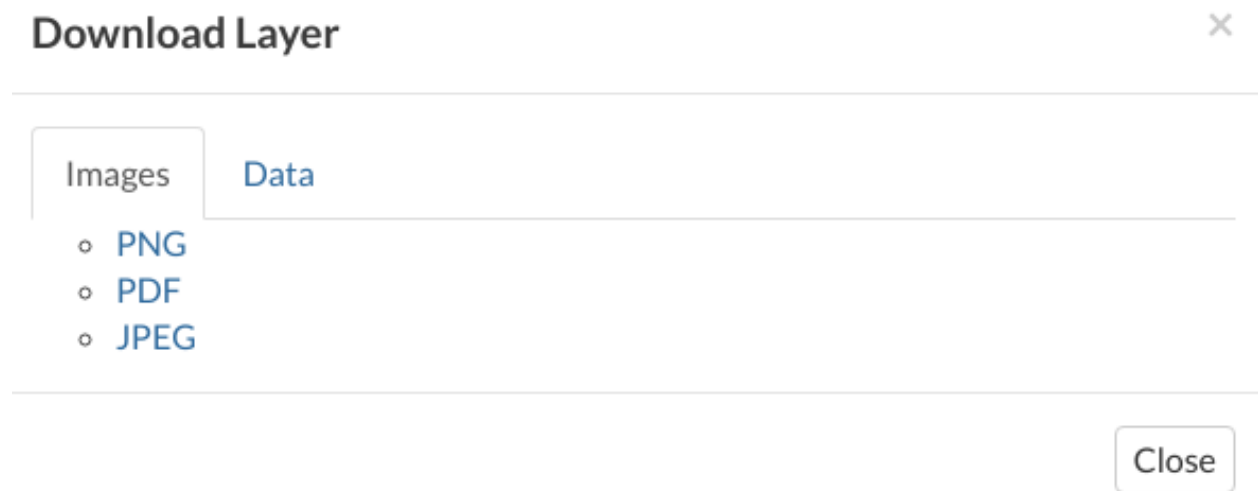


Fig. 104: Descarregando Camadas como Imagens

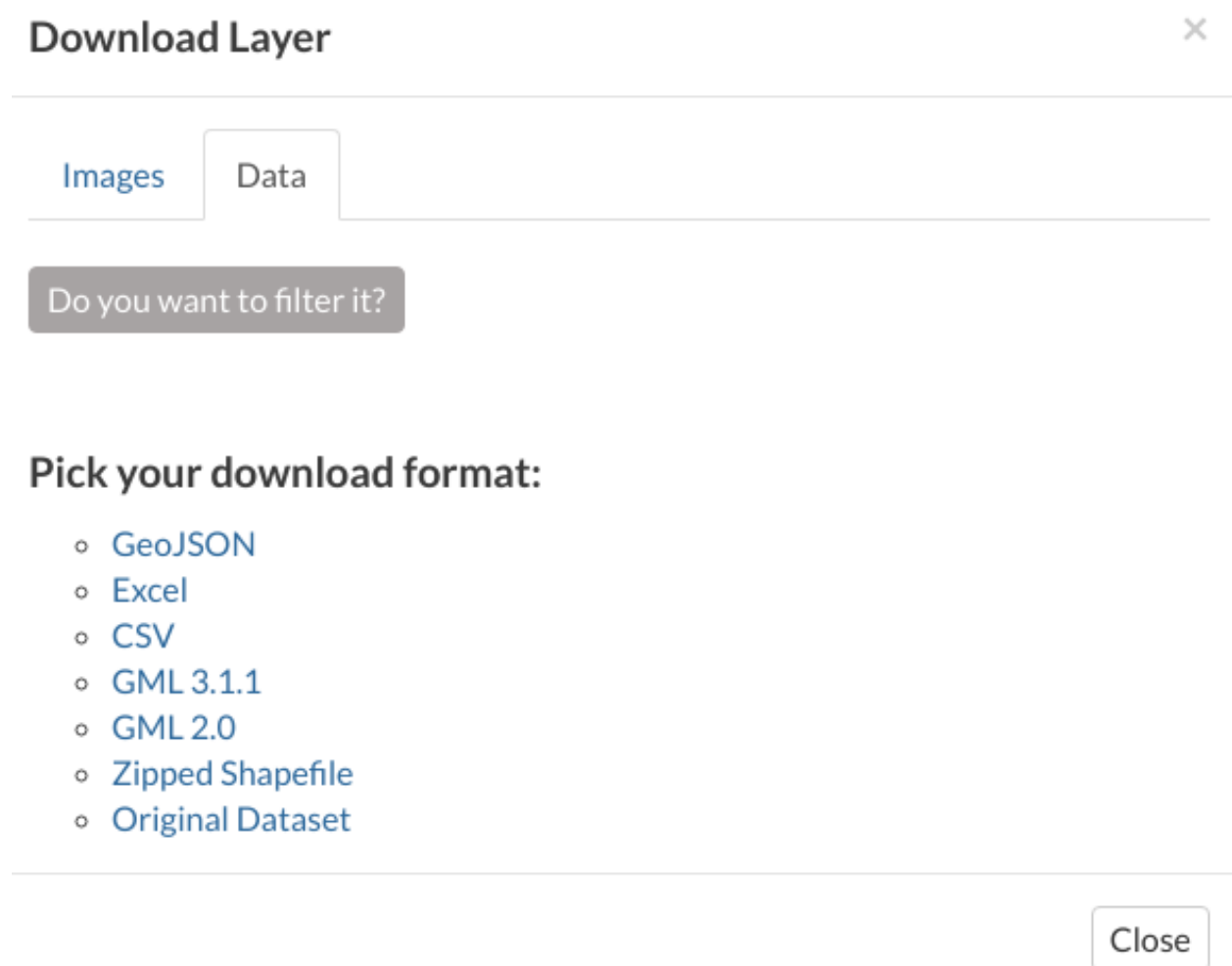


Fig. 105: Descarregando os Dados da Camada

Fig. 106: Descarregando os Dados da Camada

Edição de Camada

O botão *Editing Tools* da *Página Camada* (veja [Informação da Camada](#)) abre um painel como o mostrado na imagem em baixo.

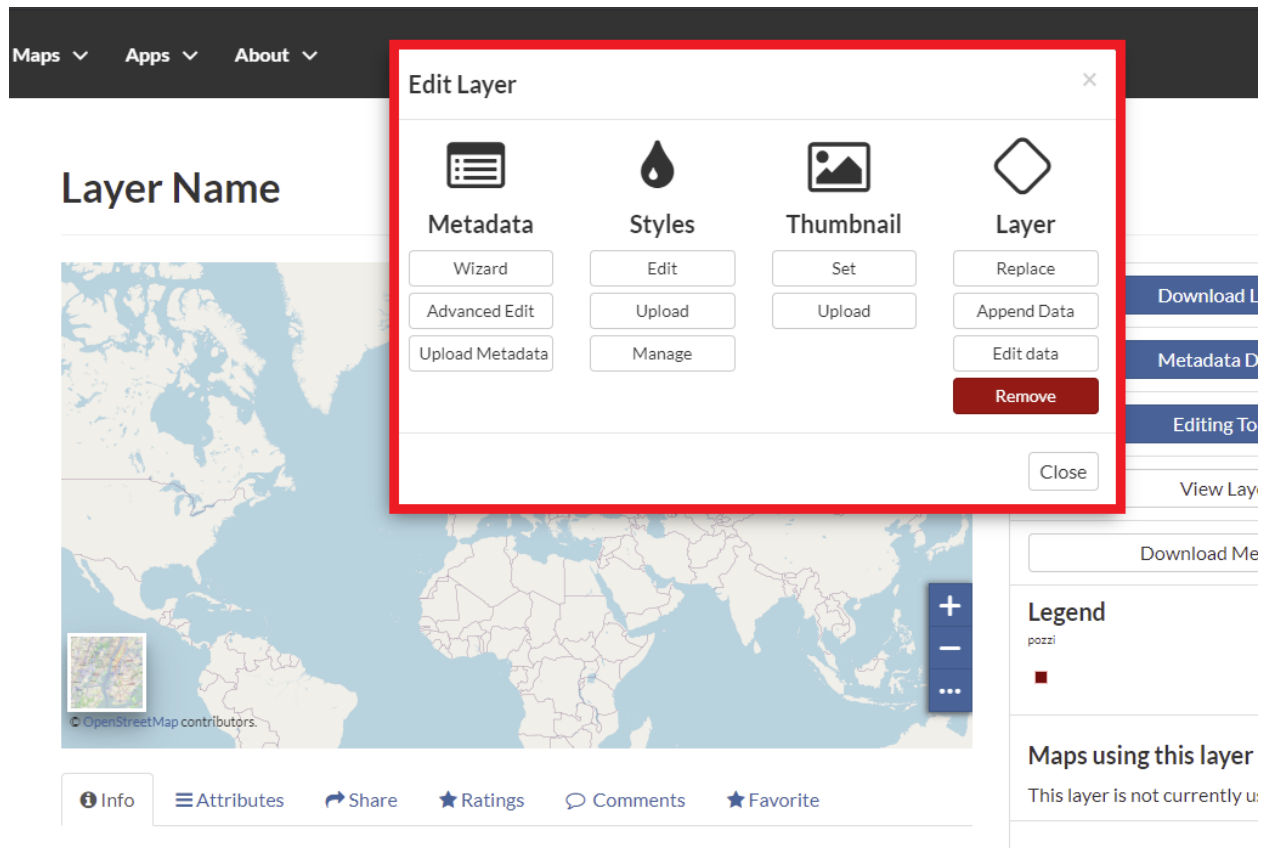


Fig. 107: O painel de Edição de Camadas

Neste painel, pode ver muitas opções agrupadas por quatro categorias:

1. *Metadados*
2. *Estilos*
3. *Miniatura*
4. *Camada*

Nesta secção, aprenderá como editar uma *Camada*, como substituir e editar os seus dados. Veja [Metadados de Camadas](#) para aprender como explorar a camada *Metadados*, como carregá-los e editá-los. Os *Estilos* serão abordados em uma secção dedicada, consulte [Modelação de Camada](#).

Configurando a Miniatura da Camada

A Miniatura da camada que será exibida na página de lista de *Camadas* pode ser alterada arrastando e ampliando a visualização da camada para seleccionar qual parte será exibida e, em seguida, clicando no botão *Set* do *Painel de Edição de Camadas*.

Uma mensagem confirmará que a miniatura foi alterada correctamente.

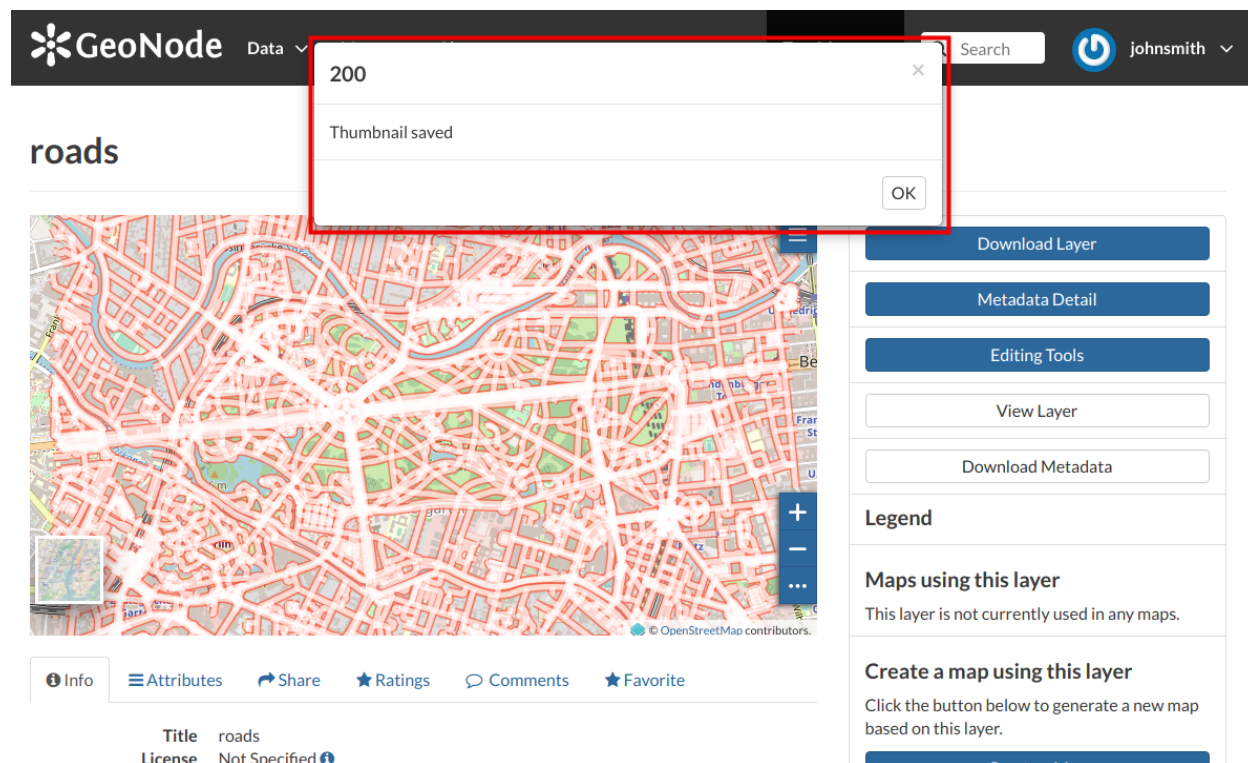


Fig. 108: O painel de Edição de Camadas

É também possível carregar manualmente uma miniatura usando o botão *Upload* do painel *Edição de Camadas*. Usando a página «Carregar Miniatura», é possível habilitar a miniatura gerada automaticamente ou carregar uma imagem para ser usada no lugar dela.

Substituindo a Camada

No painel *Edição de Camada*, clique em: `guilabel:Replace` para alterar o conjunto de dados de origem da camada. Será direccionado para a página *Substituir Camada* na qual o botão *Choose Files* permite que selecione arquivos do seu disco.

Assim que o *Charset* for seleccionado, o processo de upload pode ser accionado clicando em *Replace Layer*. Se nenhum erro ocorrer, irá ver uma mensagem como a da imagem em baixo.

Substituímos o conjunto de dados *estradas* pelo conjunto de *ferrovias*. Pode ver as diferenças na *Pré-visualização da Camada*.

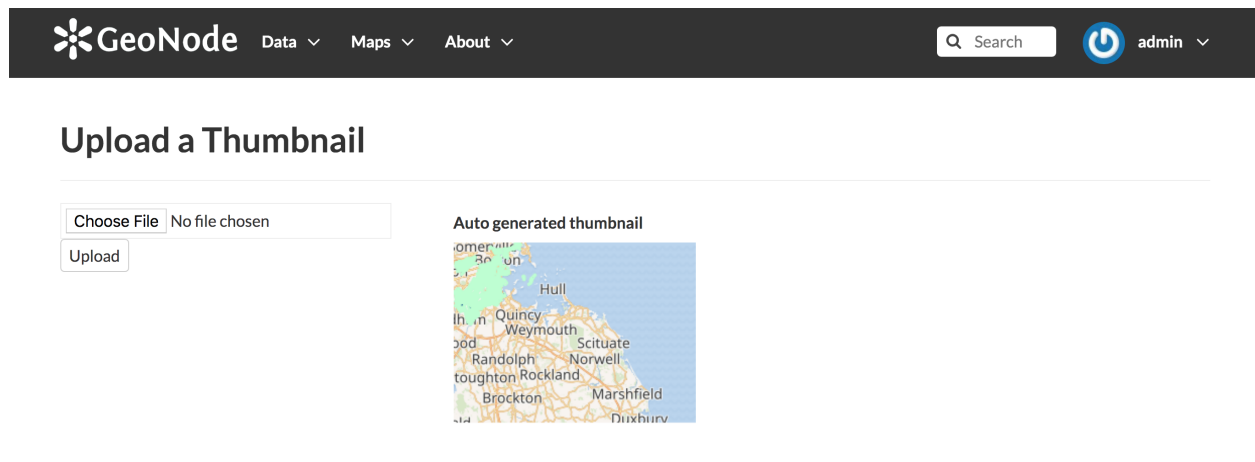


Fig. 109: O painel Carregar Miniatura

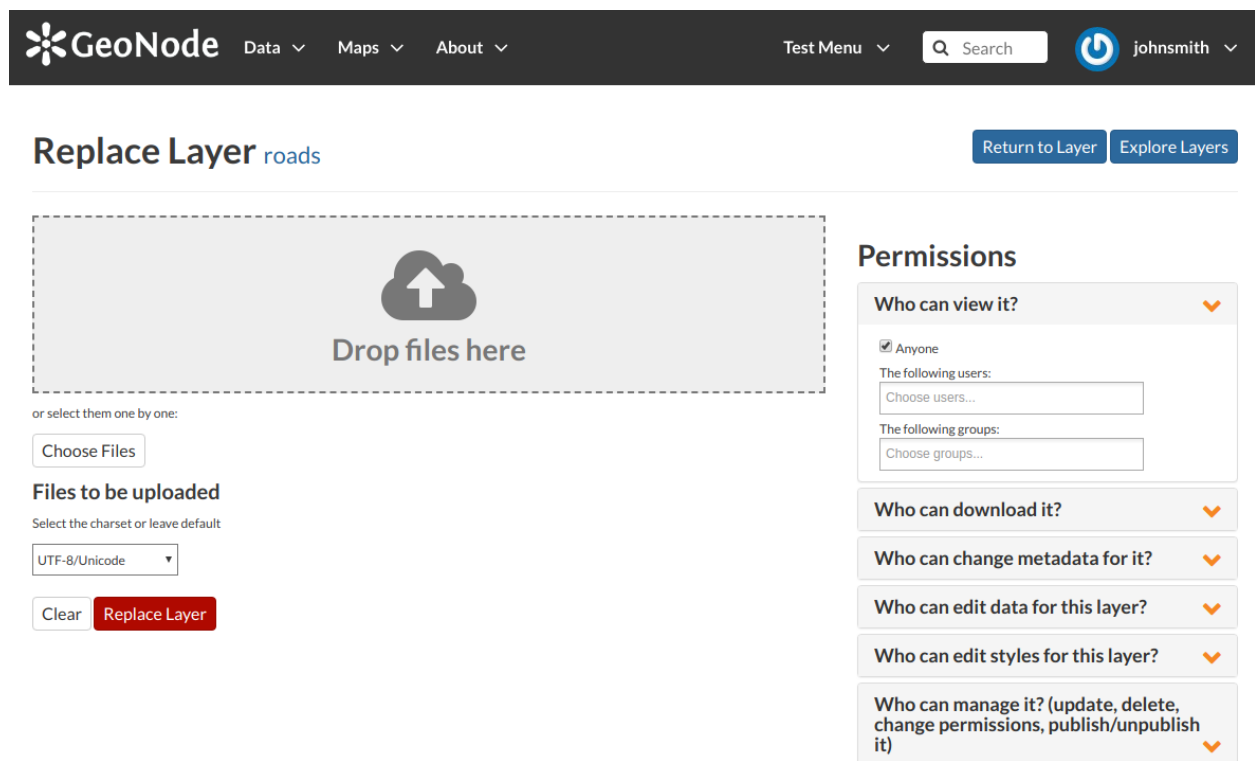


Fig. 110: Substituir uma Camada

 Data ▾ Maps ▾ About ▾

Test Menu ▾  johnsmith ▾

Replace Layer [roads](#)

[Return to Layer](#) [Explore Layers](#)



Drop files here

or select them one by one:

Choose Files

Files to be uploaded

railways

ESRI Shapefile

- [railways.shx](#) Remove
- [railways.shp](#) Remove
- [railways.prj](#) Remove
- [railways.dbf](#) Remove

Your layer was successfully updated

[Layer Info](#) [Edit Metadata](#) [Upload Metadata](#) [Upload SLD](#) [Manage Styles](#)

Select the charset or leave default

UTF-8/Unicode ▾

Clear [Replace Layer](#)

Permissions

Who can view it? ▾

☒ Anyone
The following users:

The following groups:

Who can download it? ▾

Who can change metadata for it? ▾

Who can edit data for this layer? ▾

Who can edit styles for this layer? ▾

Who can manage it? (update, delete, change permissions, publish/unpublish it) ▾

Fig. 111: Sucesso na Substituição da Camada


[Data](#) [Maps](#) [About](#)

[Test Menu](#)

johnsmith


[Download Layer](#)
[Metadata Detail](#)
[Editing Tools](#)
[View Layer](#)
[Download Metadata](#)

Legend

Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

[Create a Map](#)

Styles

The following styles are associated with this layer. Choose a style to view it in the preview map.

- ☐ A orange line style
- ☒ (default style) Red road

Info [Attributes](#) [Share](#) [Ratings](#) [Comments](#) [Favorite](#)

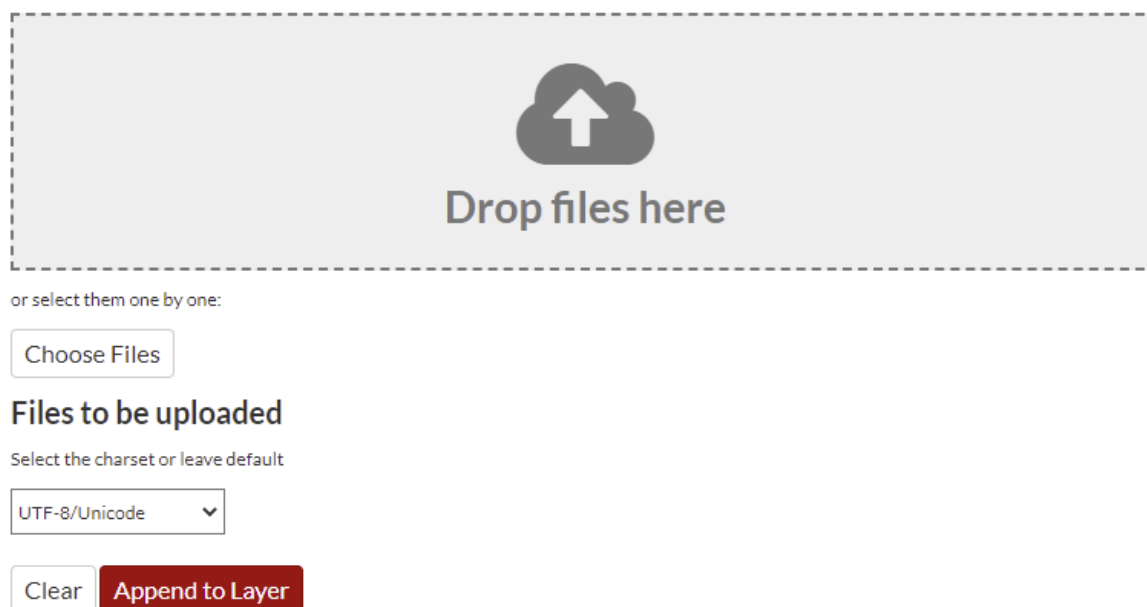
Title	roads
License	Not Specified
Abstract	No abstract provided
Publication Date	June 7, 2019, 2:15 p.m.
Type	Vector Data
Keywords	features, ne_10m_roads
Category	Transportation
Regions	Global
Owner	johnsmith
More info	-

Fig. 112: Resultado da Substituição da Camada

Anexar Dados à Camada

No painel *Edição de Camada*, clique em *Append* para anexar dados ao conjunto de dados de origem da camada. Será direcionado para a página *Anexar Camada* na qual o botão *Choose Files* permite que selecione arquivos do seu disco.

Append to Layer Layer Name



Drop files here

or select them one by one:

Choose Files

Files to be uploaded

Select the charset or leave default

UTF-8/Unicode ▼

Clear Append to Layer

Fig. 113: Anexar a uma Camada

Uma vez selecionado o *Charset*, o processo de upload pode ser accionado clicando em *Append to Layer*. Se nenhum erro ocorrer, irá ver uma mensagem como a da imagem em baixo.

Anexamos o dataset *nome_da_camada* ao existente. Pode ver as diferenças na *Pré-visualização da Camada*.

Editando os Dados da Camada

O botão *Edit data* do painel *Edição de Camadas* abre a *Camada* dentro de um *Mapa*.

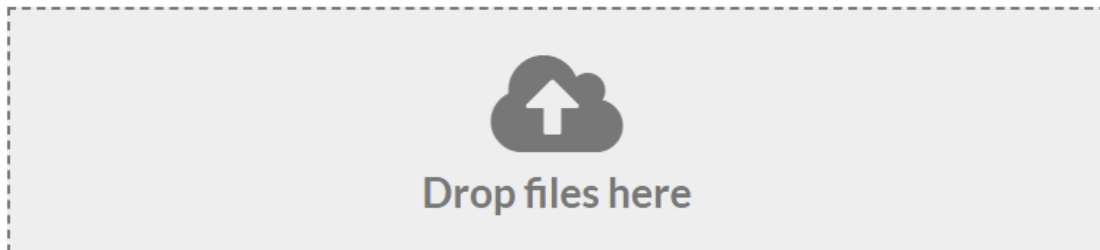
O painel *Tabela de Atributos* da *Camada* irá aparecer automaticamente na parte inferior do *Mapa*. Neste painel, todos os recursos são listados. Para cada recurso, pode aumentar o zoom clicando no ícone da lupa correspondente  no início da linha, pode também observar quais valores o recurso assume para cada atributo.

Clique no botão *Modo de Edição*  botão para iniciar uma sessão de edição.

Agora pode:

- *Adicionar novos Recursos*

Append to Layer [Layer Name](#)



or select them one by one:

Choose Files

Files to be uploaded

ESRI Shapefile

- [.dbf Remove](#)
- [prj Remove](#)
- [shp Remove](#)
- [shx Remove](#)

Your layer was successfully updated

Layer Info

Edit Metadata

Upload Metadata

Manage Styles

Fig. 114: *Sucesso na Substituição da Camada*

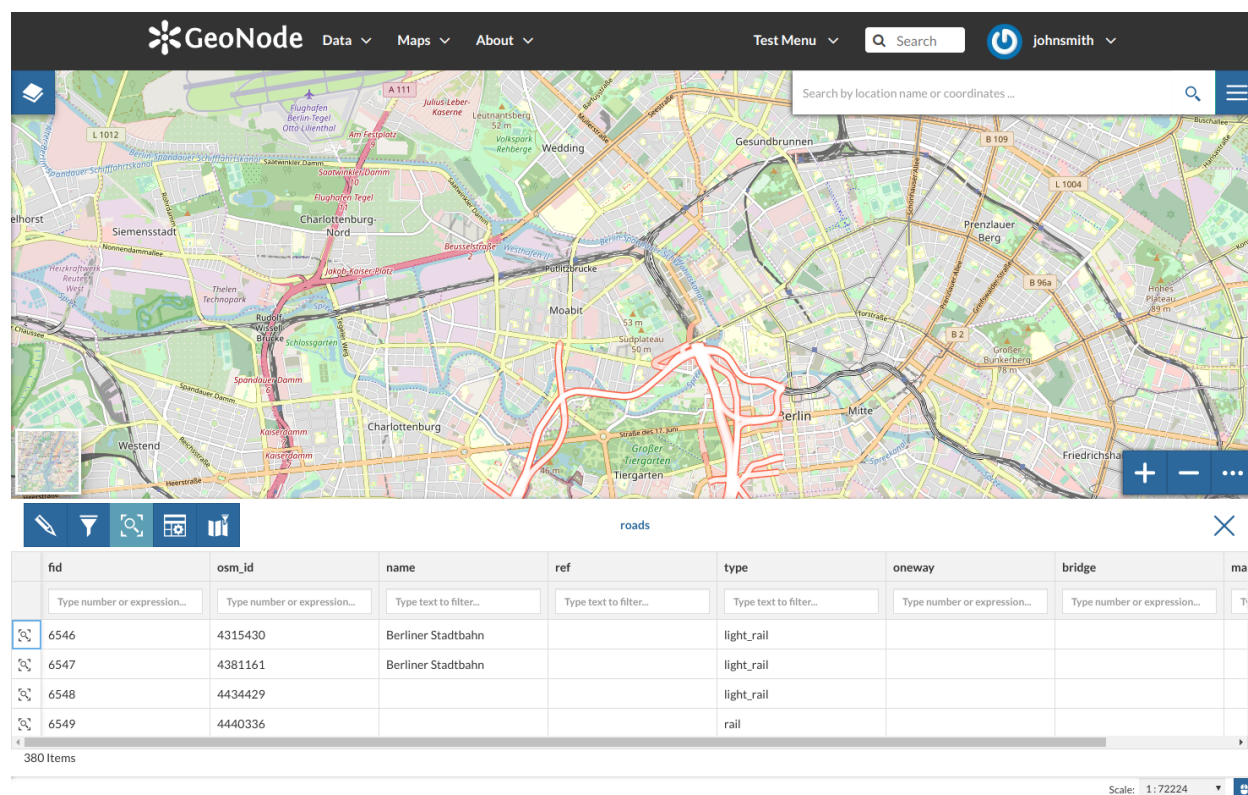


Fig. 115: Editando os Dados da Camada

Através do botão *Adicionar Novo Recurso*  é possível configurar um novo recurso para a sua camada. Preencha os campos de atributos e clique em  para salvar a sua alteração. O seu novo recurso ainda não tem geometria, clique em  para desenhar a sua forma directamente no *Mapa* e clique em  para guardá-lo.

Fig. 116: Adicione um Novo Recurso à Camada

Nota: Quando o novo recurso tem uma forma com vários vértices, deve clicar duas vezes no último vértice para finalizar o desenho.

- *Excluir Recursos*

Se deseja excluir um elemento, deve seleccioná-lo na *Tabela de Atributos* e clicar em .

Fig. 117: Excluir um Recurso

- *Alterar o Formato do Recurso*

Pode editar a forma de uma geometria existente arrastando os seus vértices com o rato. Um círculo azul permite que saiba qual o vértice que está a mover.

Fig. 118: Edição de Forma de Recurso - Alterar a forma existente

Os recursos podem ter *formas com várias partes*. Pode adicionar partes à forma ao editá-la.

Fig. 119: Edição de Forma de Recurso - Adicionar peças à forma existente

- *Altere os Atributos do Recurso*

Quando está no *Modo de Edição*, também pode editar os valores dos atributos, alterando-os directamente nos campos de texto correspondentes.

Fig. 120: Edição de Atributos do Recurso

Assim que terminar, pode encerrar a *Sessão de Edição* clicando no botão .

Por padrão, o visualizador de mapa GeoNode é baseado em [MapStore](#), consulte a [Documentação MapStore](#) para mais informações.

Metadados de Camadas

No GeoNode é dada especial importância aos *Metadados* e seus formatos padrão. Pode explorar os *Metadados* de uma *Camada* clicando no botão *Metadata Detail* na *Página da Camada*.

A página *Metadados da Camada* será exibida.

Nesta página, pode ver todo o conjunto de metadados disponíveis da camada. Os metadados são agrupados para mostrar os seguintes tipos de informações:

- *Identificação* para identificar de forma única a camada (Título, Resumo, Data de Publicação, etc.);
- *Proprietário*, o utilizador a quem pertence a camada;
- *Informações*, a Imagem de Identificação, Extensão Espacial, Sistema de Projecção e assim por diante;
- *Recursos*, Idioma, Informações suplementares e outras Informações;
- *Pontos de Contacto*, o utilizador disponível para entrar em contacto;
- *Referências*, vários links para informações de recursos e dados;
- *Autor dos Metadados*, informações sobre o autor dos metadados.

Descarregando Metadados

O botão *Download Metadata* da *Página da Camada* permite que descarregar os metadados da camada em vários formatos.

Os formatos para descarregamento disponíveis estão agrupados em três categorias:

- *Metadados completos*
- *Metadados Padrão - formato XML*
- *Informação de Atributo*

Clique no nome do formato da sua preferência para iniciar o descarregamento.


Data ▾
Maps ▾
About ▾
Test Menu ▾


johnsmith ▾

roads



Download Layer

Metadata Detail

Editing Tools

View Layer

Download Metadata

Legend

roads0

 dark orange line

Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

Create a Map

Info

Attributes

Share

Ratings

Comments

Favorite

Title roads

License Not Specified ⓘ

Abstract No abstract provided

Publication Date June 10, 2019, 12:58 p.m.

Fig. 121: O botão Detalhe da Camada de Metadados


[Data](#)
[Maps](#)
[About](#)
[Test Menu](#)

 [johnsmith](#)

Metadata : roads

[Return to Layer](#)

Identification

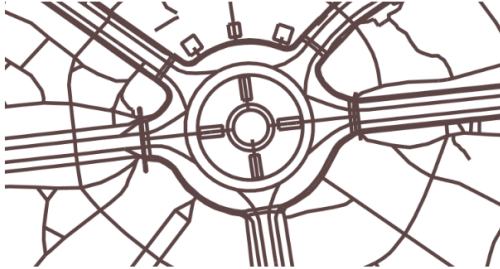
Title	roads
Abstract	No abstract provided
License	Not Specified
Publication Date	June 10, 2019, 12:58 p.m.
Type	Vector Data
Keywords	features ne_10m_roads
Category	Transportation
Regions	Global
Approved	Yes
Published	Yes
Featured	No

Owner

Name	johnsmith
email	johnsmith@mail.com
Position	None
Organization	None
Location	None
Voice	None
Fax	None

Information

Identification Image



Spatial Extent	---
Projection System	EPSG:4326
Extension x0	13.326002500000000
Extension x1	13.386977980000000
Extension y0	52.503006700000000
Extension y1	52.525999300000000

Features

Language	English
Supplemental Information	No information provided

Contact Points

Name	johnsmith
email	johnsmith@mail.com
Position	None
Organization	None
Location	None
Voice	None
Fax	None

References

Link Online	/layers/geonode:roads0
Metadata Page	/layers/geonode:roads0/metadata_detail

Thumbnail	roads.png
Remete Thumbnail	roads.png
Legend	roads.png
GeoJSON	roads.json
Excel	roads.excel
CSV	roads.csv
GML 3.1.1	roads.gml
GML 2.0	roads.gml
Zippped Shapefile	roads.zip
PNG	roads.png
PDF	roads.pdf
JPEG	roads.jpg
Original Dataset	roads.zip

OGC WFS: geonode Service	Geoservice OGC:WFS
OGC WMS: geonode Service	Geoservice OGC:WMS

Metadata Author

Name	johnsmith
email	johnsmith@mail.com
Position	None
Organization	None
Location	None
Voice	None
Fax	None

Fig. 122: Os Detalhes de Metadados da Camada

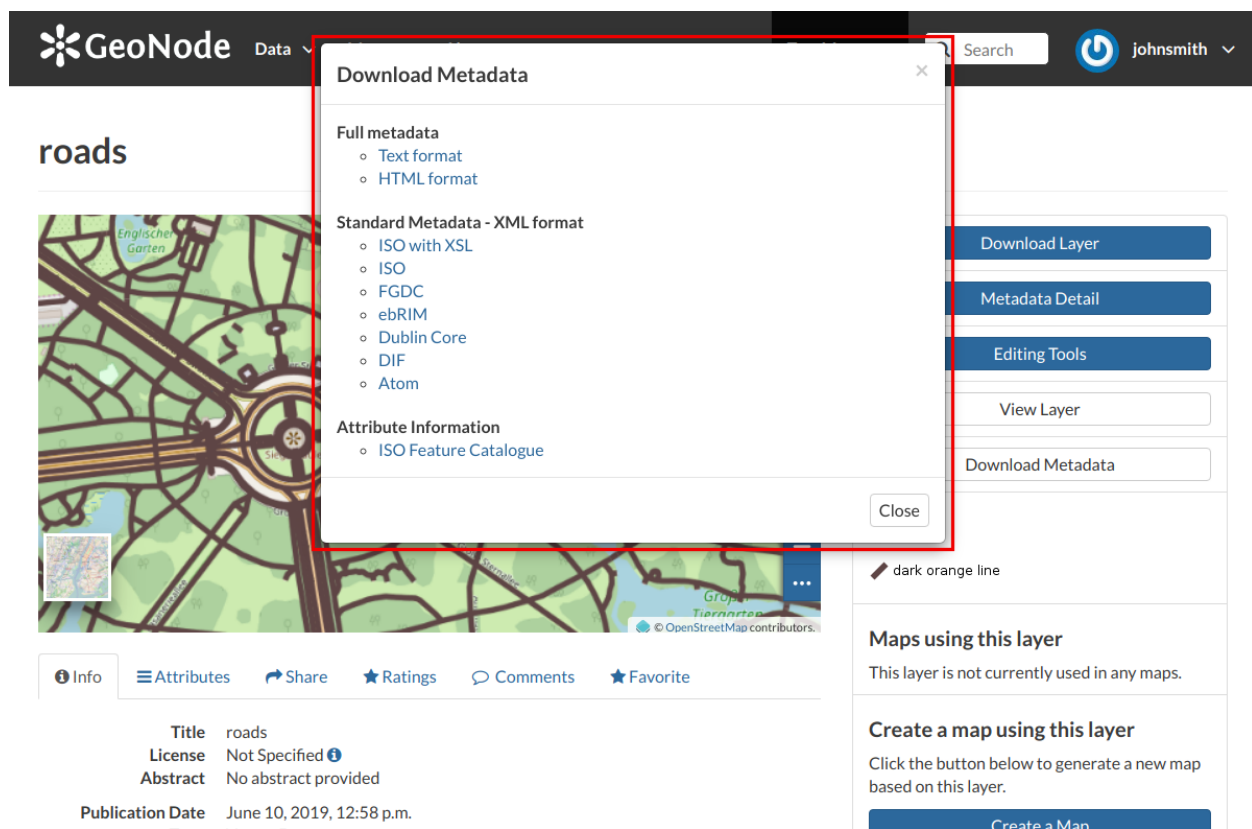


Fig. 123: Como Descarregar Metadados

Assistente de Metadados

Os metadados contêm todas as informações relacionadas à camada. Eles fornecem informações essenciais para sua identificação e compreensão. Os metadados também tornam a camada mais visível por meio da pesquisa por outros utilizadores.

Os *Metadados* de uma camada podem ser alterados através de um *Assistente* que envolve quatro etapas, uma para cada tipo de metadados considerado:

- **Metadados Básicos**

As duas primeiras etapas são obrigatórias (nenhuma camada será publicada se as informações necessárias não forem fornecidas), enquanto as duas últimas são opcionais.

Metadata for roads

Completeness: 50% (Check Schema mandatory fields)

Buttons: Edit, Preview, Settings

Progress Bar: Mandatory (red), Mandatory (red), Optional (blue), Optional (blue)

Steps:

- 1 Basic Metadata**
 - Thumbnail: Map of roads with 'Edit' button.
 - Title: roads
 - Abstract: Editor with 'No abstract provided' and '3 WORDS POWERED BY TINY'.
- 2 Location and Licenses**
- 3 Optional Metadata**
 - Free-text Keywords: features, roads
 - Date type: Publication, Date: 2020-11-10 17:1
 - Category: ---
 - Group: ---
- 4 Dataset Attributes**

Buttons: Return to Layer, Update, Next >>

Fig. 124: *Metadados de Camada Básica*

Na primeira etapa, o sistema pede que você insira os seguintes metadados:

- A *Miniatura* da camada (clique *Edit* para alterá-la);
- O *Título* da camada, que deve ser claro e compreensível;

- Um *Resumo*; breve resumo narrativo do conteúdo da camada

Nota: O painel *Resumo* permite inserir código HTML por meio de um editor de texto *wysiwyg*

- As *Datas de Criação/Publicação/Revisão* que definem o período de tempo coberto pela camada;
- As *Palavras-Chave*, que devem ser escolhidas dentro da lista disponível. O colaborador busca as palavras-chave disponíveis clicando na barra de busca, ou no logotipo da pasta que representa, ou digitando as primeiras letras da palavra desejada;
- A *Categoria* à qual a camada pertence;
- O *Grupo* ao qual a camada está vinculada.

• Localização e Licenças

Metadata for roads

Completeness: 50%
Check Schema mandatory fields

Edit Preview Settings

Mandatory Mandatory Optional

1 Basic Metadata 2 Location and Licenses 3 Optional Metadata 4 Dataset Attributes

Language English
License Not Specified
DOI a DOI will be added by Admin before publicatio
Attribution authority or function assigned, as to a ruler, legi

Regions Global

Data quality statement

Restrictions

Other constraints

Return to Layer << Back Update Next >>

Fig. 125: Localização e Metadados de Licenças para Camadas

A lista a seguir mostra quais tipos de metadados que deve inserir (veja também a imagem em baixo):

- O *Idioma* da camada;
- A *Licença* do conjunto de dados;
- O *DOI* do conjunto de dados; se disponível, representa o [Digital Object Identifier](#) do recurso
- A *Atribuição* do conjunto de dados; autoridade ou função atribuída, como governante, assembleia legislativa, delegado ou semelhante
- As *Regiões*, que informam sobre a extensão espacial coberta pela camada. As extensões propostas cobrem as seguintes escalas: global, continental, regional, nacional;
- A *Declaração de Qualidade de Dados* (explicação geral do conhecimento do produtor de dados sobre a linhagem de um conjunto de dados);
- Potenciais *Restrições* na partilha de camadas.

Nota: Os painéis *Declaração de Qualidade de Dados* e *Restrições* permitem inserir código HTML por meio de um editor de texto *wysiwyg*

• Metadados Opcionais

As informações complementares são:

- A *Edição* para indicar a referência ou a origem da camada;
- O *Propósito* da camada e os seus objetivos;
- Qualquer *Informação Suplementar* que possa fornecer uma melhor compreensão da camada carregada;
- A *Frequência de Manutenção* da camada;
- Os utilizadores que são *Responsáveis* pela camada, seu *Proprietário* e o *Autor* de seus metadados;
- O *Tipo de representação espacial* utilizada.

Nota: Os painéis *Objectivo* e *Informações suplementares* permitem inserir código HTML por meio de um editor de texto *wysiwyg*

• Atributos do Conjunto de Dados

Nesta etapa pode enriquecer os atributos do conjunto de dados com informações úteis como as seguintes:

- O *Rótulo* exibido
- Uma *Descrição* detalhada
- A *Ordem de Exibição*
- O *Tipo de Exibição*; o valor padrão é *Rótulo*, o que significa que o valor do atributo será renderizado como um texto simples. Existe a possibilidade de instruir o GeoNode a tratar os valores como diferentes tipos de media. Por exemplo, se os valores do atributo seleccionados irão conter urls de imagem, seleccionando *IMAGE Tipo de Exibição* permitirá que o GeoNode renderize a imagem directamente ao consultar a camada dos mapas. O mesmo para os tipos *MIME VIDEO*, *AUDIO* ou *IFRAME*.
- O sinalizador *Visível*; permite instruir o GeoNode a ocultar ou não um atributo dos resultados de *Obter Tipo de Recurso*

É possível definir um template HTML completamente customizado para o resultado *Obter Tipo de Recurso*. Isso é possível activando o sinalizador *Usar um modelo personalizado* conforme mostrado na figura em baixo.

Metadata for roads

Completeness
 ✖ Check Schema mandatory fields
 92%

Edit Preview Settings

Mandatory
Mandatory
Optional

1

Basic Metadata

Other, Optional, Metadata

Edition ⓘ

version of the cited resource

Purpose

File Edit View Insert Format Tools Table

Help

↶ ↷ **B** *I* U ~~S~~ ...

P 0 WORDS POWERED BY TINY

Supplemental information

File Edit View Insert Format Tools Table

Help

↶ ↷ **B** *I* U ~~S~~ ...

2

Location and Licenses

temporal extent start

Maintenance frequency ⓘ

Spatial representation type ⓘ

3

Optional Metadata

temporal extent end

4

Dataset Attributes

Responsible Parties

Point of Contact

admin × ▾

Responsible and Permissions

Owner

admin × ▾

Metadata Author

admin × ▾

Fig. 126: Metadados de Camada Opcionais

Metadata for roads

Completeness
 ✖ Check Schema mandatory fields
 92 %

Edit Preview Settings

Mandatory
Mandatory
Optional

1

Basic Metadata

2

Location and Licenses

3

Optional Metadata

4

Dataset Attributes

Use a custom template? Off

	Attribute	Label	Description	Display Order	Display Type	Visible
⋮	fid			1	Label	☒
⋮	the_geom			2	Label	☐
⋮	cat			3	Label	☒
⋮	label			4	Label	☒

Label
 URL
Image
 Video (mp4)
 Video (ogg)
 Video (webm)
 Video (3gp)
 Video (flv)
 Video (YouTube/VIMEO - embedded)
 Audio
 IFRAME

Fig. 127: Metadados de Atributos do Conjunto de Dados para Camadas

Metadata for roads

Completeness
✖ Check Schema mandatory fields
92 %

Edit Preview Settings

Mandatory

Mandatory

Optional

1

2

3

4

Basic Metadata

Location and Licenses

Optional Metadata

Dataset Attributes

Use a custom template? ☒

File Edit View Insert Format Tools Table Help

↶ ↷

B *I* U ~~S~~

Helvetica

14px

Paragraph

≡ ≡ ≡ ≡

≡ ≡

≡ ≡ ≡ ≡

≡ ≡ ≡ ≡

the custom GetFeatureInfo template HTML contents.

P

0 WORDS POWERED BY TINY

Fig. 128: *Use um modelo personalizado*

Usando uma keyword `{properties.<attribute_name>}`, pode instruir o GeoNode a renderizar o valor real do atributo no mapa.

Como uma instância, o exemplo em baixo

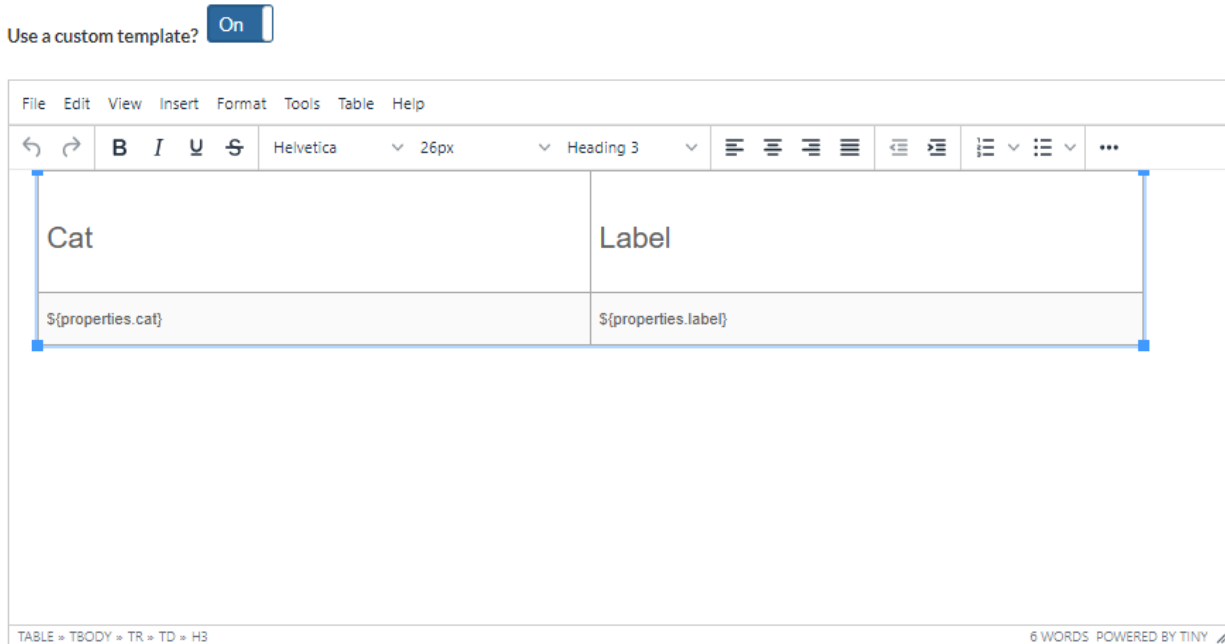


Fig. 129: Use um modelo personalizado: HTML

Irá renderizar uma Tabela HTML junto com os valores mostrados aqui abaixo

Use *next* >> ou << *back* para navegar por essas etapas. Assim que terminar, clique em *Update*.

Alguns metadados são obrigatórios, se algum desses metadados faltarem, a barra *Completeness* mostra uma mensagem em vermelho como a da imagem em baixo.

Edição Avançada de Metadados

No painel *Edição de Camadas* o *Advanced Edit* também está disponível.

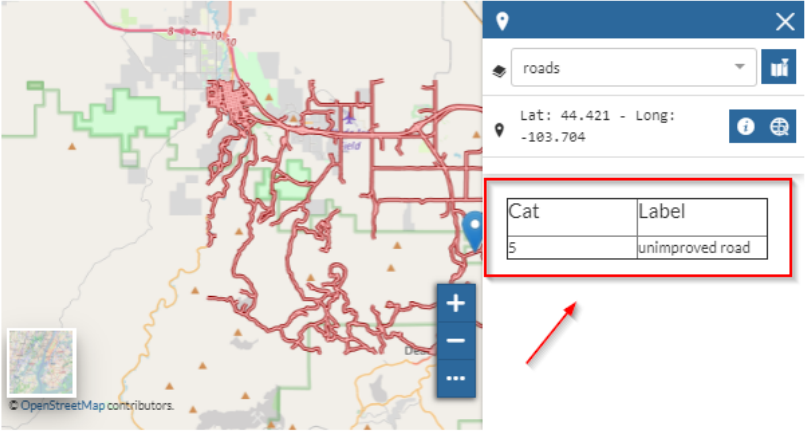
Clique nele para exibir a *Página de Edição Avançada de Metadados*. Esta página permite editar todos os metadados da camada descritos no parágrafo anterior. Assim que terminar de editá-los, clique em *Update* para guardar as suas alterações.

Carregando Metadados

Os utilizadores também podem fazer upload de um documento XML de metadados (em formato ISO, FGDC ou Dublin Core) para preencher os principais elementos de metadados do GeoNode automaticamente. A imagem em baixo mostra a aparência da página.

Clique em *Choose Files* para seleccionar o documento do seu disco, em seguida clique em *Upload files* para iniciar o processo de upload.

roads



Attribute Name	Label	Description
fid		
cat		
label		

Download Layer

Metadata Detail

Editing Tools

View Layer

Download Metadata

Legend

roads

Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

Create a Map

Fig. 130: Use um modelo personalizado: Obtenha Resultados de Informações sobre o Recurso

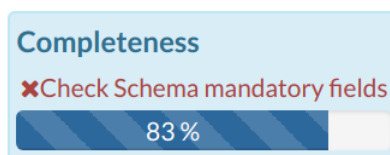


Fig. 131: Barra de Progresso de Completude

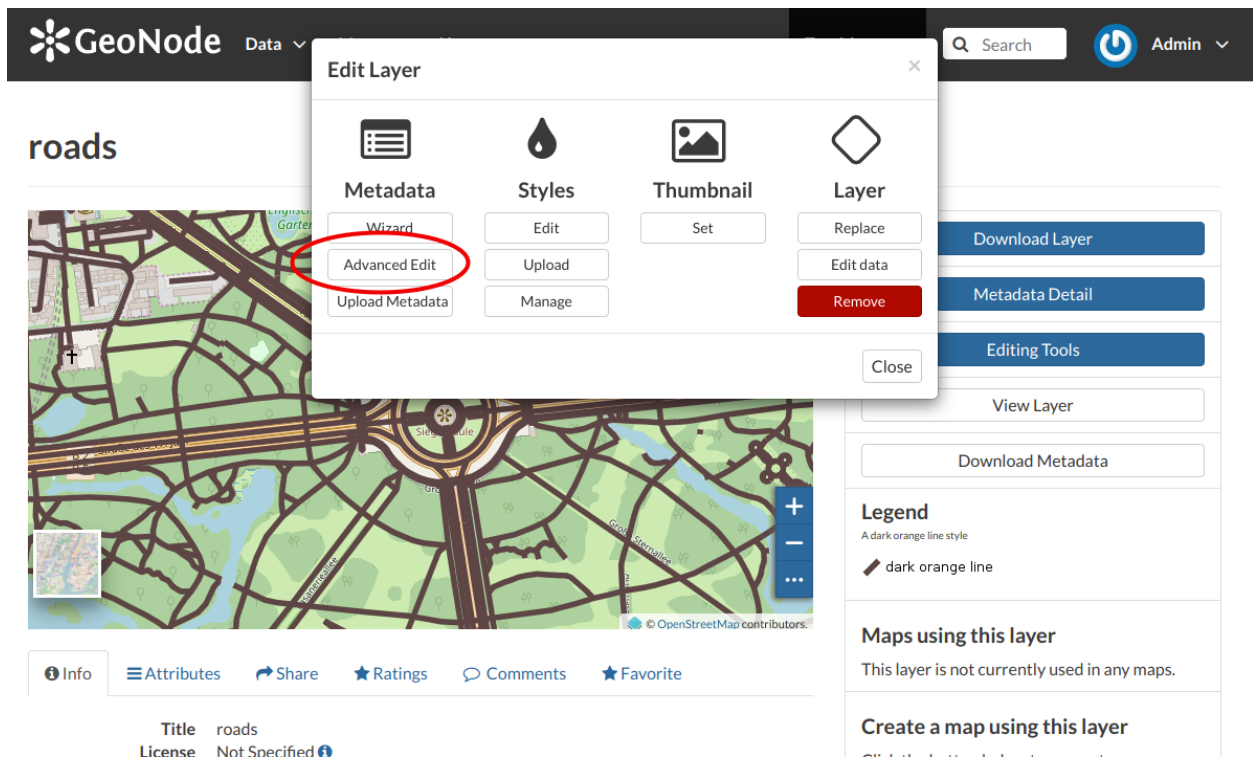


Fig. 132: O botão de Edição Avançada

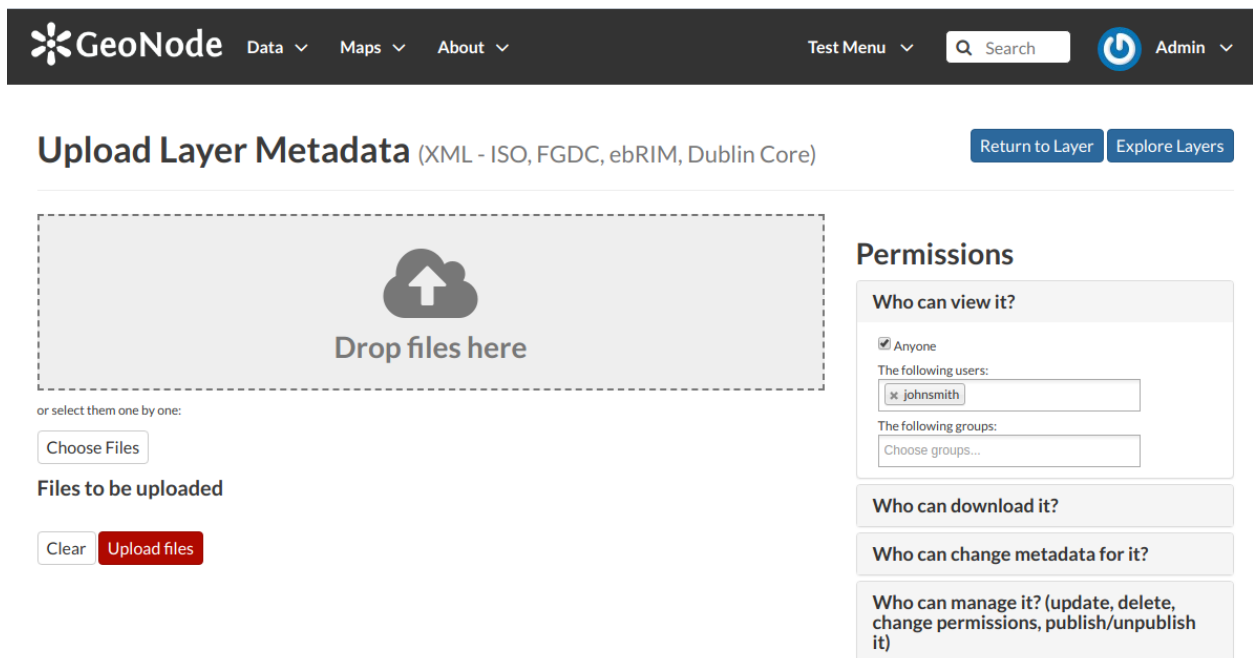


Fig. 133: A página de Edição Avançada de Metadados

Modelação de Camada

Os mapas são úteis porque permitem obter uma compreensão mais profunda dos seus dados, permitindo-lhe visualizá-lo de muitas maneiras diferentes. Assim, pode contar histórias diferentes dependendo da forma como os dados são apresentados. Para qualquer dado ou camada, deve explorar diferentes opções de estilo e escolher o melhor estilo.

No GeoNode cada camada tem um *Estilo Predefinido* que é determinado pela natureza dos dados que está a mapear. Ao carregar uma nova camada (veja *Carregamento de Camadas*) será associado um novo estilo predefinido.

Fig. 134: *Estilo Predefinido para Camadas*

Referindo-se ao exemplo acima, as linhas laranja escuras não são muito boas para representar as vias navegáveis, pelo que teríamos de mudar este estilo. Nos parágrafos seguintes, você irá aprender a criar um novo estilo a partir de modelos, como editar um estilo, como carregar estilos a partir de arquivos e como geri-los.

Criação de novos Estilos

Para criar um novo estilo, abra a *Página Camada* (veja *Informação da Camada*) e clique em *Editing Tools*. Em seguida, clique no botão *Edit* na secção *Estilos* do painel *Edição de Camadas* (veja a imagem em baixo).

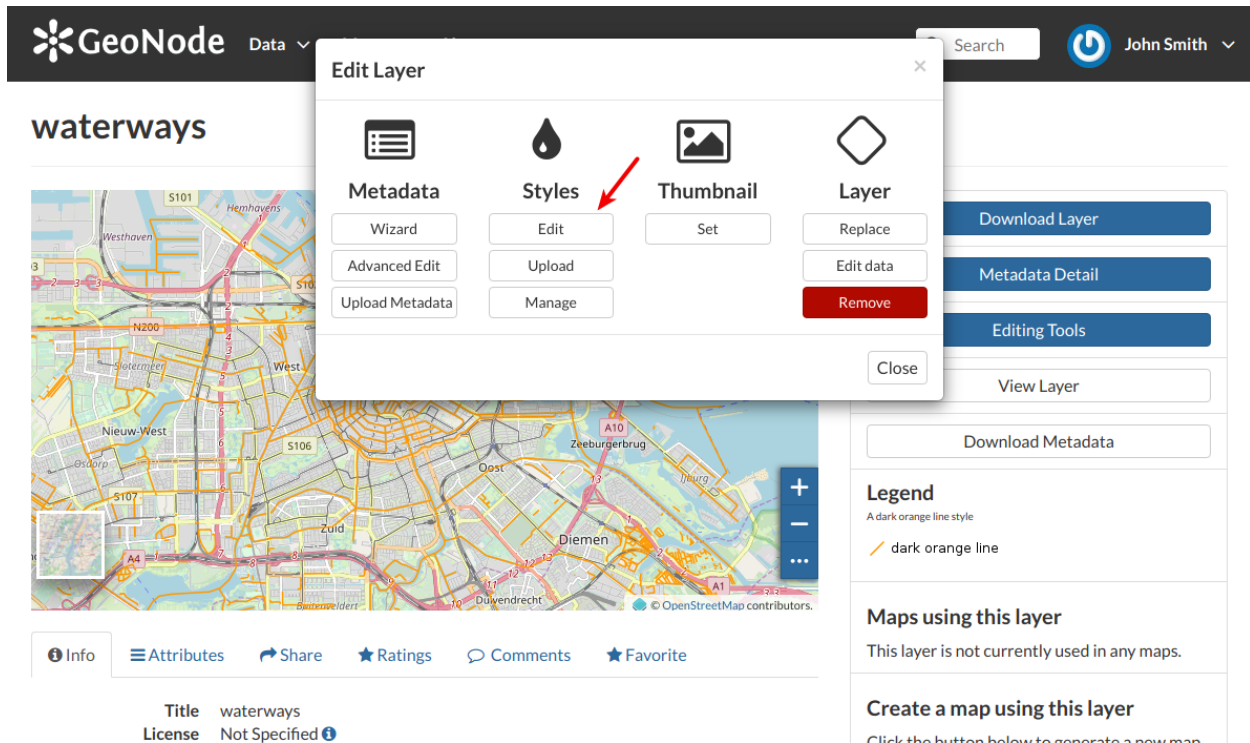


Fig. 135: Botão Editar Estilos

A *Camada* irá abrir num novo *Mapa*. O *Painel de Estilos* irá mostrar-lhe todos os estilos disponíveis para a camada e algumas ferramentas úteis.

Agora siga os passos em baixo:

1. Clique no botão . O *Painel de Modelos de Estilo* será aberto.
2. Escolha um *Modelo de Estilo* na lista (os estilos *CSS* e *SLD* estão disponíveis).
3. Clique no botão  para adicionar o *Modelo de Estilo* à lista de estilos.
4. Insira um *Título* e um *Resumo* (opcional), a seguir clique em *Save*.

O estilo que criou agora está adicionado à *Lista de Estilos*.

Poderá também, ver esse novo estilo na *Página de Camadas*.

Agora pode mudar o estilo clicando na caixa de selecção correspondente.

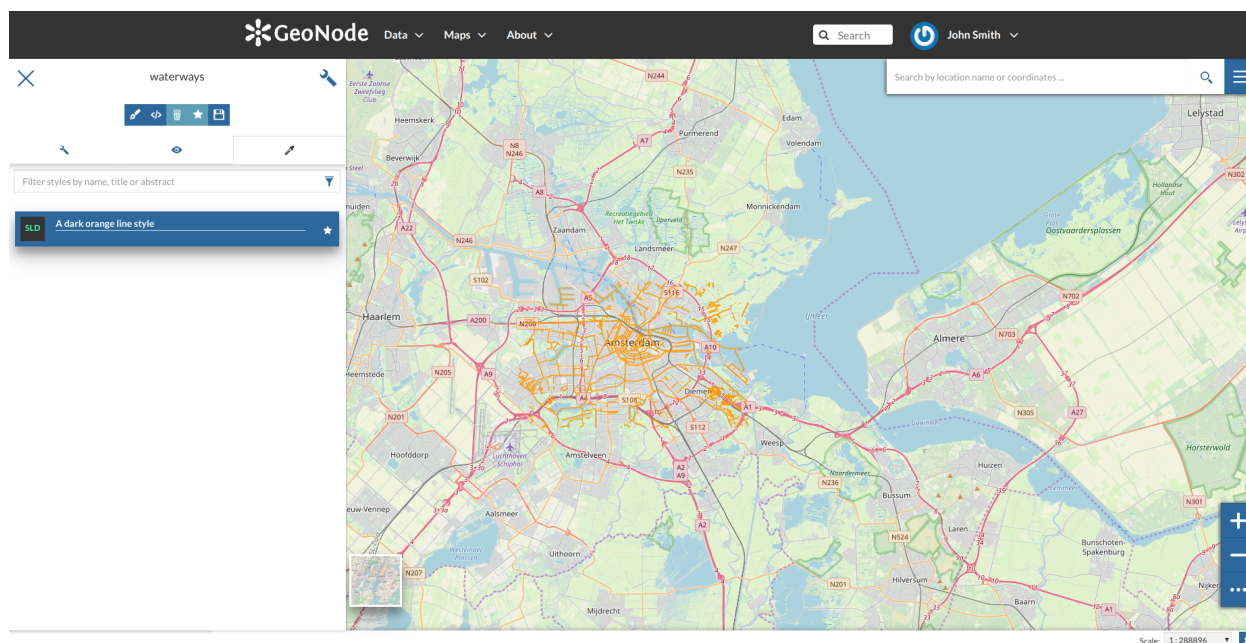


Fig. 136: O Painel de Estilos no Mapa

Seria bom alterar o estilo para diminuir a opacidade da cor de preenchimento, bem como para reduzir a largura das linhas. O [MapStore](#) incorporado disponibiliza um poderoso *Editor de Estilos* para realizar essas tarefas. No próximo parágrafo, explicaremos como.

Edição do Estilo de Camada

As etapas a seguir mostram como editar estilos:

1. De dentro do mapa, abra a *TOC (Tabela de Conteúdos)* clicando no botão



2. Clique em



3. Abra a guia *Estilo*



Aviso: A edição de estilos é permitida apenas para os utilizadores que possuem a permissão necessária. Veja [Alterando as Permissões da Camada](#) para ler mais)

4. Selecciono o *Estilo* e clique em



5. Edite o estilo. O *Editor de Estilos* ajuda a escrever estilos válidos por meio do *Validador de Sintaxe* que mostra um pop-up em caso de erros (veja a imagem em baixo).

6. Clique em



Veja o seguinte gif para recapitular todo o processo.

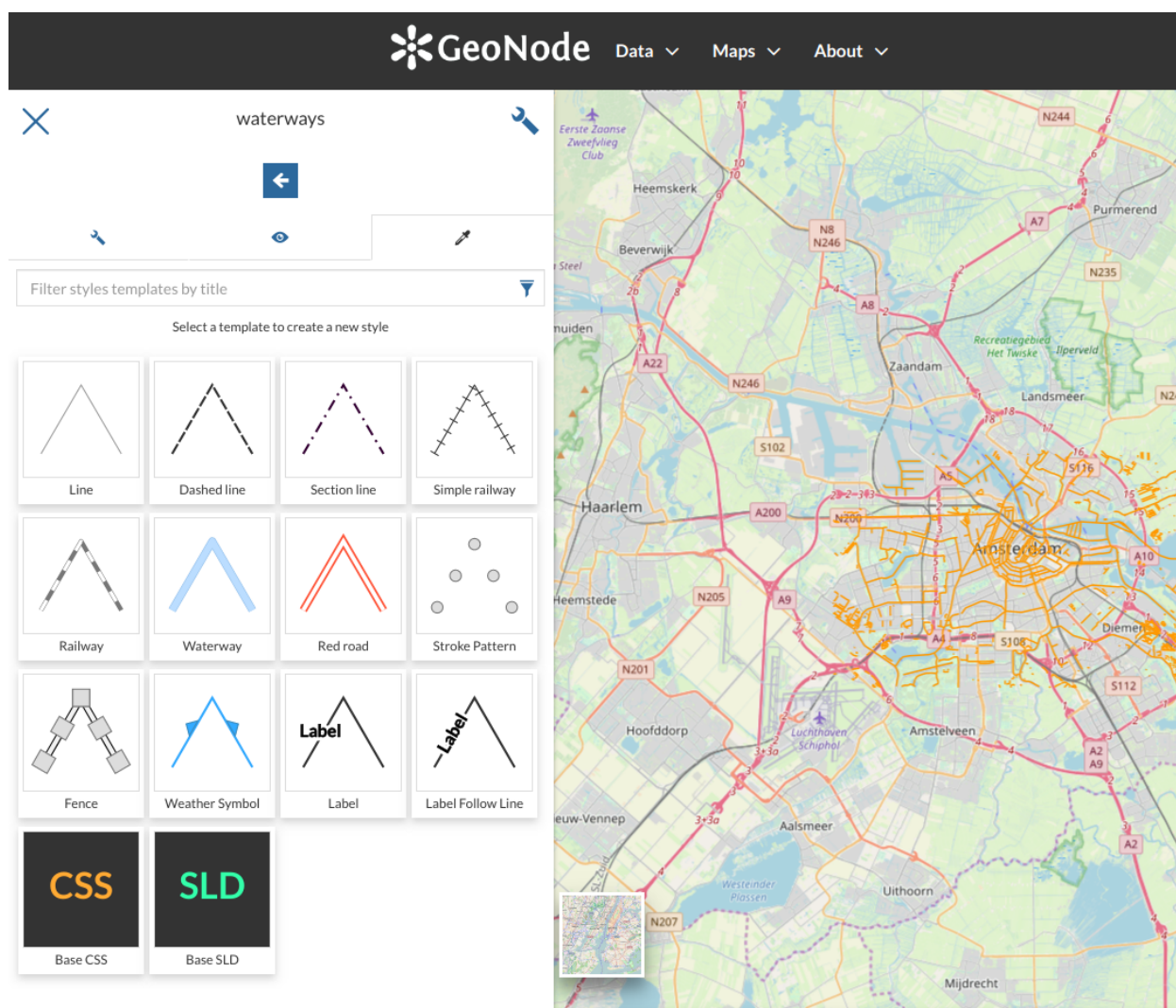


Fig. 137: Criar novos Estilos

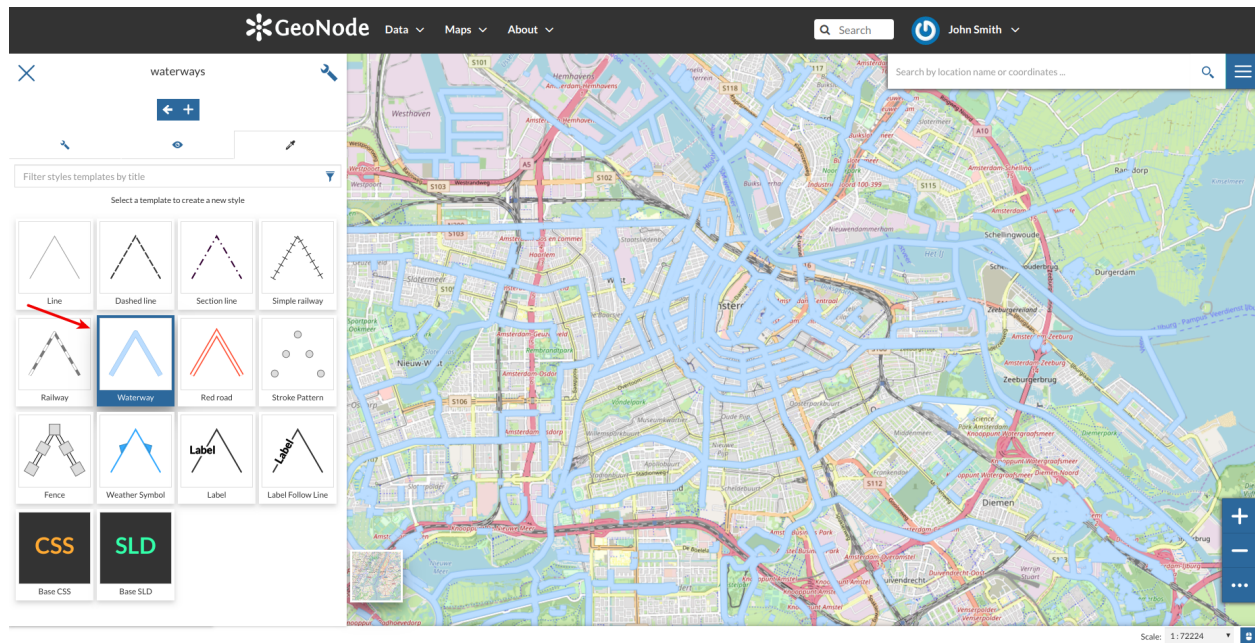


Fig. 138: *Modelos de Estilo*

Create new style

Title

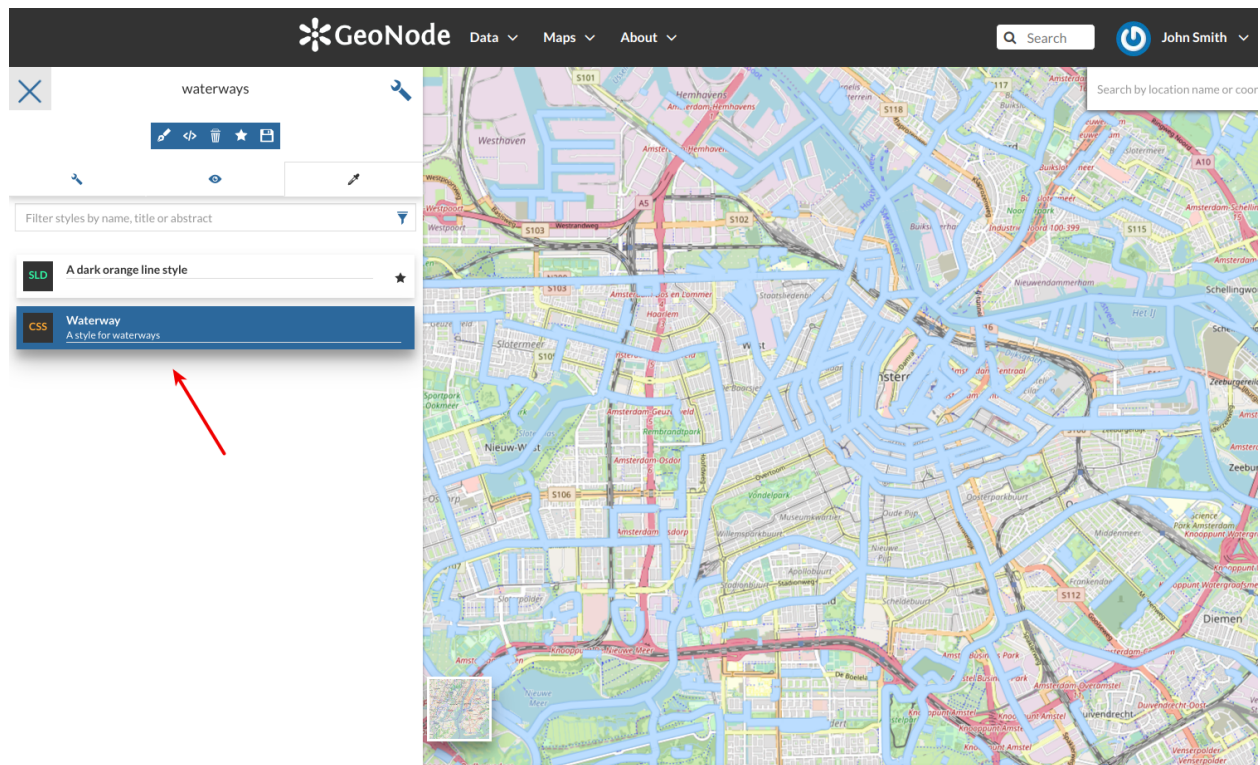
Waterway

Abstract

A style for waterways

Save

Fig. 139: *Título e Resumo para novos Estilos*

Fig. 140: *Novos Estilos para a lista*

Pode, também, decidir tornar o seu novo estilo o *Estilo Padrão* dessa camada. Clique em  para fazer isso.

Clique em  para excluir o estilo.

Carregamento de Estilos

No GeoNode, é também possível carregar um estilo existente a partir de um arquivo.

Aviso: Actualmente, apenas estilos no formato **SLD (Style Layer Descriptor 1.0, 1.1)** podem ser carregados no GeoNode.

Da *Página de Camadas* clique em *Ferramentas de Edição* para abrir o painel *Ferramentas de Edição* e siga os passos abaixo:

1. Clique no botão *Upload* da secção *Estilos*
2. Clique em *Choose Files* e seleccione o seu estilo no disco
3. Clique em *Upload files*

Assim que o processo for concluído, o novo *Estilo* ficará visível na *Página de Camadas*.

waterways

Download Layer

Metadata Detail

Editing Tools

View Layer

Download Metadata

Legend
A dark orange line style

Maps using this layer
This layer is not currently used in any maps.

Create a map using this layer
Click the button below to generate a new map based on this layer.

Create a Map

Styles
The following styles are associated with this layer. Choose a style to view it in the preview map.

☒ (default style) A dark orange line style

☐ Waterway

Refresh Attributes and Statistics of

Info **Attributes** **Share** **Ratings** **Comments** **Favorite**

Title waterways
License Not Specified **Abstract** No abstract provided
Publication Date June 11, 2019, 4:21 a.m.
Type Vector Data
Keywords features , waterways
Regions Global
Owner johnsmith
More info -

Layer WMS GetCapabilities document

Fig. 141: A Página de Camadas com o novo estilo

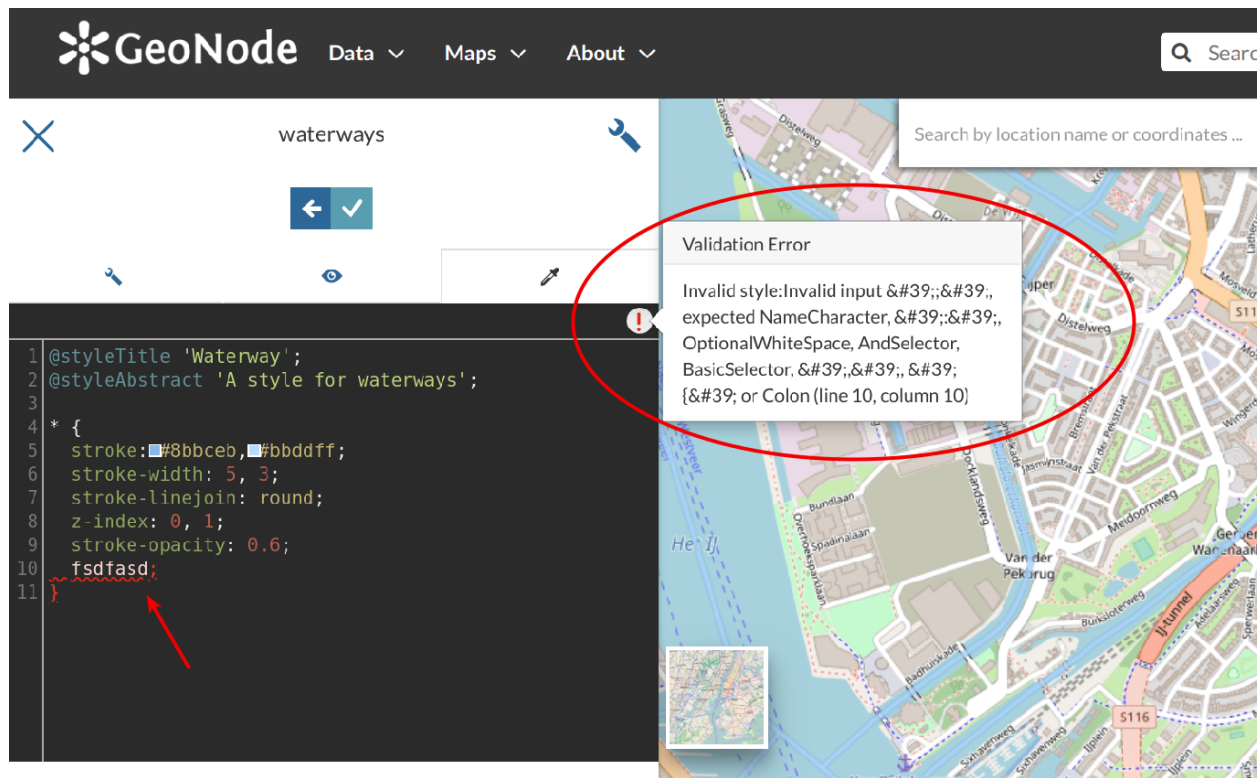


Fig. 142: Validação de Sintaxe do Editor de Estilos

Fig. 143: O Editor de Estilos

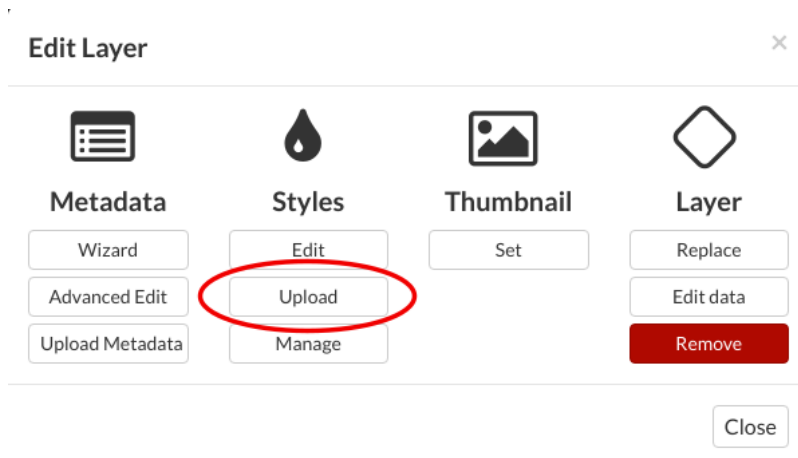


Fig. 144: Botão Carregar Estilos

Fig. 145: Estilos de Upload

Gestão de Estilos

Dada uma camada, pode fazer a gestão de todos os seus estilos na *Página de Gestão de Estilos* acessível a partir do botão *Manage* do painel *Edição de Camadas*.

Nesta página, pode:

- Veja o *Nome da Camada*
- Adicionar/remover estilos para/da lista *Estilos disponíveis*
- Escolha o *Estilo Padrão da Camada* na lista *Estilos disponíveis*

Clique em *Update Available Styles* para salvar as suas alterações.

Gestão Avançada de Camadas com MapStore

O GeoNode disponibiliza ao utilizador métodos avançados de manipulação de camadas como filragem, edição de atributos e exportação em diferentes formatos. Em suma, estes métodos são fornecidos pelo MapStore para onde redireccionamos o utilizador para a documentação deste para mais detalhes.

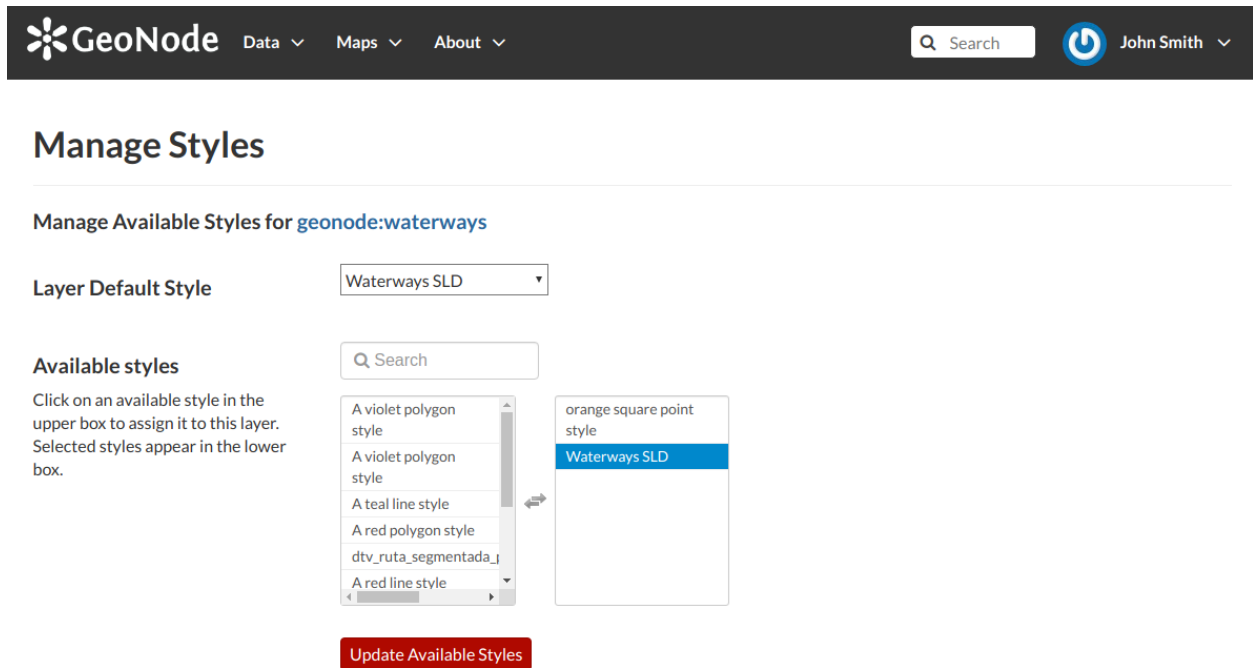


Fig. 146: Gestão de Estilos

Filtragem de Camadas

With GeoNode you can filter a layer via it's attributes, direct map filter by drawing an area of interest over the map canvas and via cross-layer filter, allowing intersection, contained and contains overlay methods. For more detail please check the MapStore documentation [here](#).

Tabela de Atributos

O GeoNode providência ferramentas de manipulação de atributos que permitem a edição e criação de novos atributos com simplicidade. Estas ferramentas fornecidas pelo MapStore também permitem ao utilizador filtrar, pesquisar, fazer zoom a partir de uma perspectiva de tabela de atributos como num normal SIG Desktop. Para mais detalhes por favor veja a documentação do MapStore [aqui](#).

Modelação Avançada

O MapStore permite a edição avançada de estilos de camadas que não foram cobertos na secção anterior do GeoNode. Se pretender aprofundar os seus conhecimentos neste tipo de capacidades, por favor siga esta [ligação para a documentação](#).

1.10.6 Gestão de Mapas

Mapas são conjuntos de camadas exibidas em conjunto num mapa interactivo da web. Mapas podem ser compostos no compositor de mapas e guardados como recursos GeoNode. Os mapas também podem ser associados a metadados, classificações e comentários.

Nesta secção irá aprender como criar e partilhar um novo mapa.

Criação de Mapas

Nesta secção, vamos criar um *Mapa* usando algumas camadas carregadas, combiná-las com algumas outras camadas de serviços remotos da web e, em seguida, partilhar o mapa resultante para visualização pública.

Para criar novos mapas pode utilizar:

- o link *Create Map* do menu *Maps* na barra de navegação

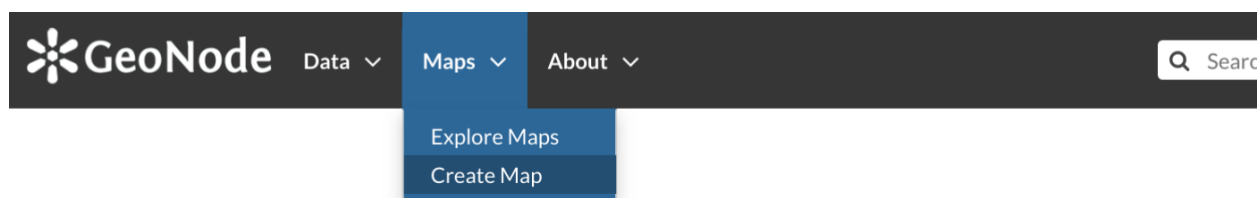


Fig. 147: O link Criar Mapa

- o botão *Create Map* na *Página de Camadas* (cria um mapa usando uma camada específica)
- o botão *Create New Map* na página *Explorar Mapas*

O novo *Mapa* será aberto em um *Visualizador de Mapas* como o da imagem em baixo.

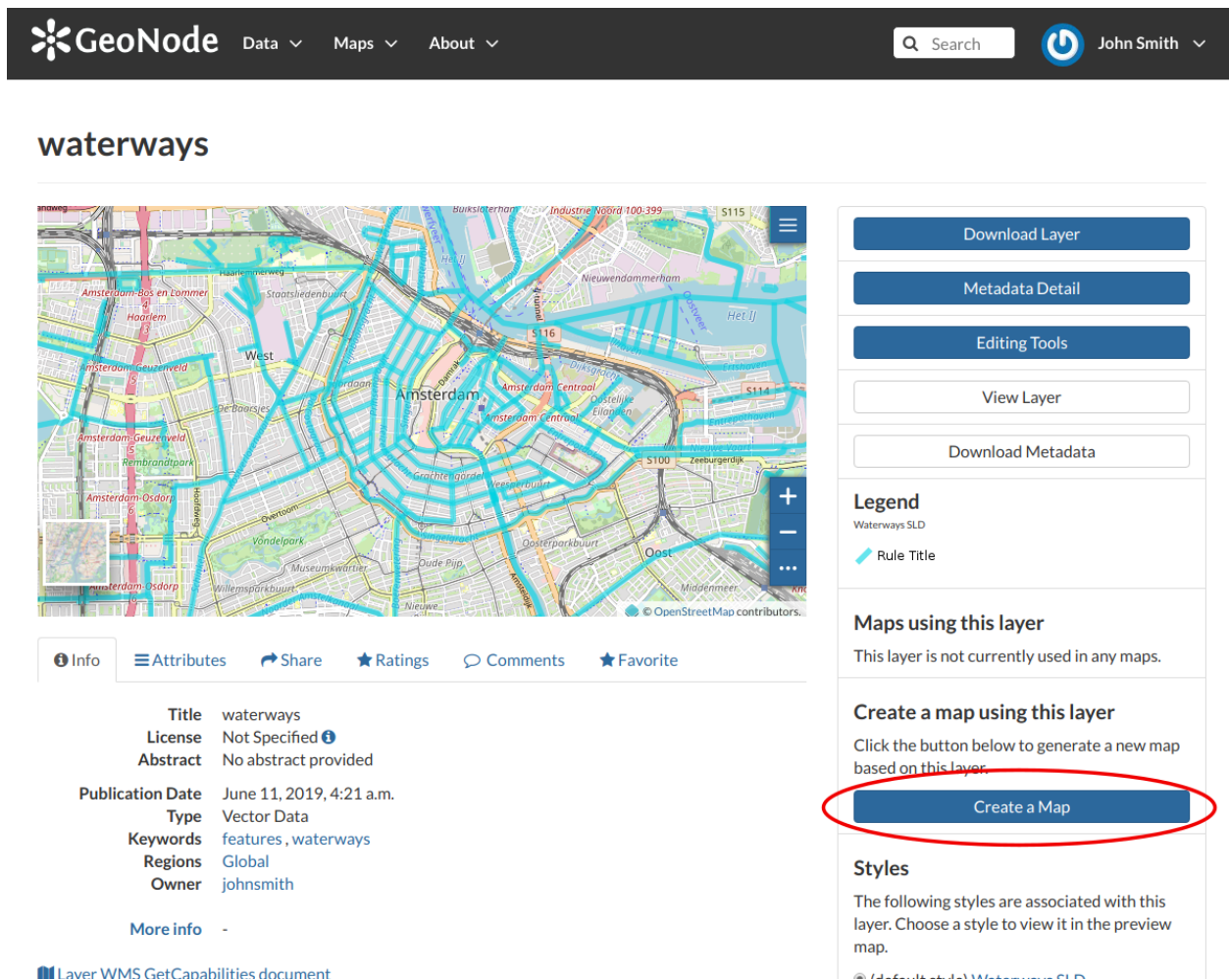
No canto superior esquerdo, o botão  abre o *Índice (TOC)* do *Mapa*. Permite gerir todas as camadas associadas ao mapa e adicionar novas a partir do *Catálogo*.

O componente *TOC* torna possível gerir a sobreposição de camadas no mapa, mudando suas posições relativas na lista (arraste e solte-as para cima ou para baixo na lista).

Também permite ocultar/mostrar camadas ( e ), ampliar as extensões das camadas () e gerir suas propriedades ().

Uma vez que as camadas do mapa foram definidas, é possível salvar o *Mapa* clicando em  e escolhendo *Save as*.

Se seguiu as etapas acima, acabou de criar o seu primeiro *Mapa*. Agora deve vê-lo na página *Explorar Mapas*, consulte [Informação do Mapa](#) para mais detalhes.



waterways

Download Layer

Metadata Detail

Editing Tools

View Layer

Download Metadata

Legend

Waterways SLD

Rule Title

Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

Create a Map

Styles

The following styles are associated with this layer. Choose a style to view it in the preview map.

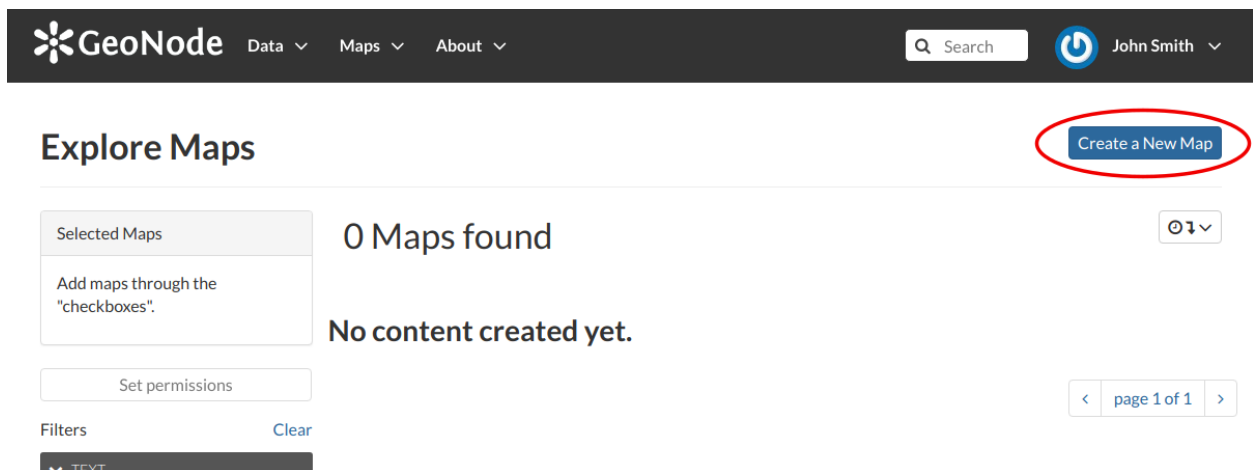
Default style: Waterways SLD

Title	waterways
License	Not Specified
Abstract	No abstract provided
Publication Date	June 11, 2019, 4:21 a.m.
Type	Vector Data
Keywords	features, waterways
Regions	Global
Owner	johnsmith

More info -

Layer WMS GetCapabilities document

Fig. 148: O botão Criar Mapa



Explore Maps

Create a New Map

0 Maps found

No content created yet.

Set permissions

Filters Clear

TEXT

page 1 of 1

Fig. 149: O botão Criar Novo Mapa

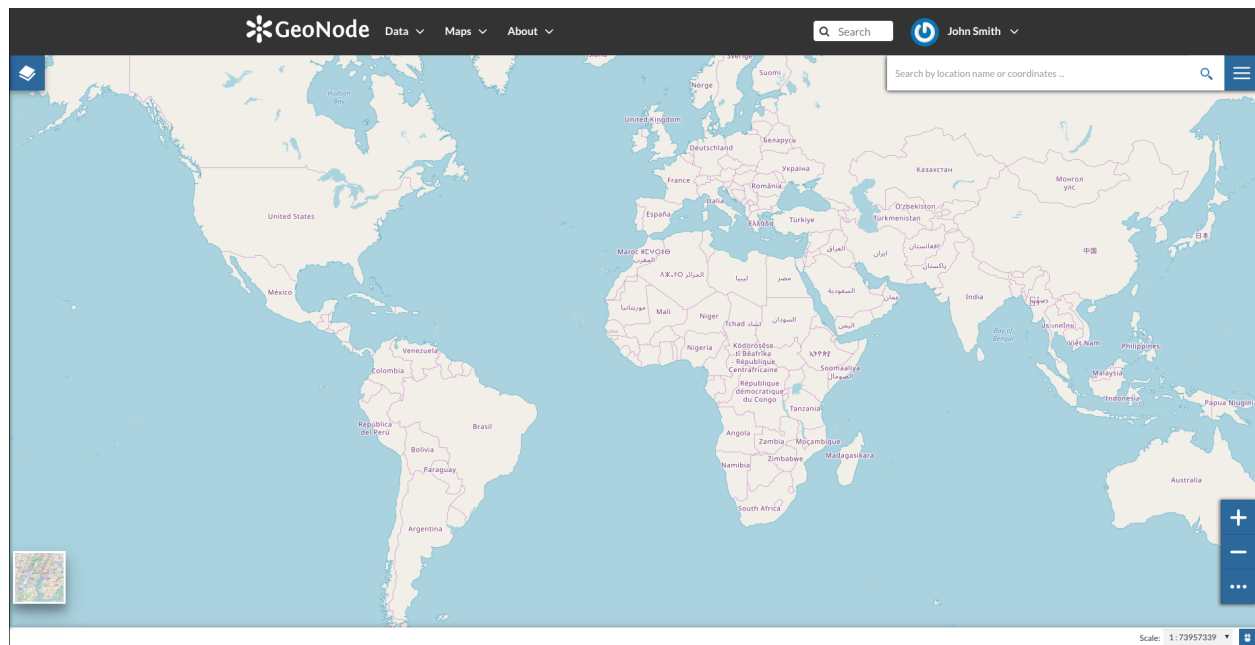


Fig. 150: O Visualizador de Mapas

Fig. 151: Criação de novos Mapas

Vamos verificar com mais detalhe as ferramentas do *Visualizador de Mapas* na secção *Exploração de Mapas*.

Informação do Mapa

Conforme mencionado na secção *Mapas*, no GeoNode pode ver os seus mapas e todos os mapas publicados através da ligação *Explore Maps* da barra de navegação.

Clique no título do *Mapa* do seu interesse para abrir a página de *Informações*, que deve ser semelhante a esta.

A *Página do Mapa* está dividida em três secções principais:

1. a secção *Pré-visualização do Mapa*, sob o título
2. a secção *Guias*, sob a pré-visualização da camada
3. a secção *Ferramentas*, no lado direito da página

Pré-visualização do Mapa

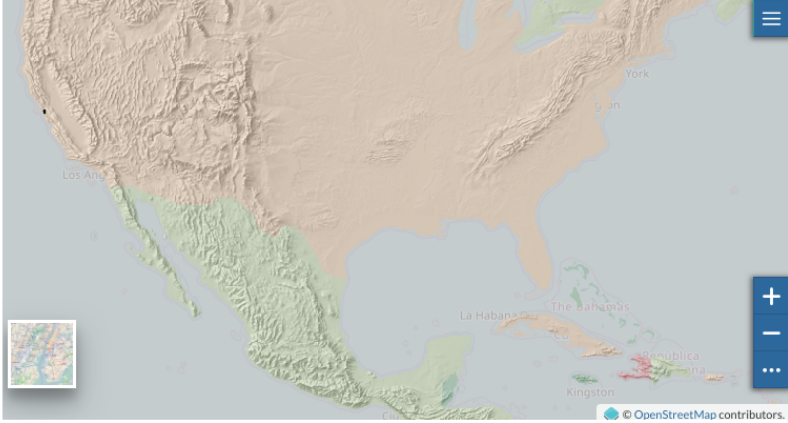
A *Pré-visualização do Mapa* mostra o *Mapa* com funcionalidades muito básicas:

- o *Comutador de Mapa Básico* que permite alterar o mapa básico;
- a ferramenta *Zoom in/out* para ampliar e diminuir a visualização;
- a ferramenta *Zoom na extensão máxima* para que o zoom se ajuste ao tamanho da camada;
- a ferramenta *Consulta de Objectos* para inquirir informações sobre os objectos do mapa clicando no mapa;


[Data](#)
[Maps](#)
[About](#)

 John Smith

My New Map



[Download Map](#)
[Metadata Detail](#)
[Editing Tools](#)
[View Map](#)

Map Layers

This map uses the following layers:

- epi:epi-environmental-performance-index-2010_biodiversity-and-habitat
- US_MSR
- San Francisco Landuse OSM

Permissions

Specify which users can view or modify this map

[Change Permissions of this Map](#)

Copy this map

Duplicate this map and modify it for your own purposes

[Create a New Map](#)

Map WMS

WMS layer group for local map layers:
(on local OWS)

Publish local map layers as WMS layer group

[Publish Map WMS](#)

About

Owner, Point of Contact, Metadata Author

 johnsmith
John Smith Foundation

[Info](#)
[Share](#)
[Ratings](#)
[Comments](#)
[Favorite](#)

Title My New Map

License Not Specified 

Publication Date June 12, 2019, 8:17 a.m.

Regions Global , Africa , Central Africa , West Africa , Pacific , Kiribati

Owner johnsmith

[More info](#) -

 Map layers WMS GetCapabilities document

Fig. 152: A página de Informações do Mapa

- a ferramenta *Imprimir* para imprimir a pré-visualização.

Fig. 153: *Pré-visualização do Mapa*

Veja a [Documentação do MapStore](#) para saber mais.

Secções de Guias

A página *Informações do Mapa* mostra algumas secções das guias contendo diferentes informações sobre o mapa:

- A guia *Info* está activa por padrão. Esta secção da guia mostra alguns metadados como o Título, a Licença, a Data de Publicação, etc. Os metadados também indicam o proprietário do mapa e quais regiões estão envolvidas. O link `Map Layers WMS GetCapabilities document` também é fornecido.

Info	Share	Ratings	Comments	Favorite
Title My New Map				
License Not Specified ⓘ				
Publication Date June 12, 2019, 8:17 a.m.				
Regions Global , Africa , Central Africa , West Africa , Pacific , Kiribati				
Owner johnsmith				
More info -				
Language English				
Supplemental Information No information provided				
Map layers WMS GetCapabilities document				

Fig. 154: *Guia de Informações dos Mapas*

- A guia *Partilhar* fornece as ligações para partilhar o mapa nas redes sociais ou email.

Info	Share	Ratings	Comments	Favorite
-------------	--------------	----------------	-----------------	-----------------

Share This Map

- [Email](#)
- [Facebook](#)
- [Twitter](#)
- [Google +](#)

Fig. 155: *Partilhar Mapa*

- Você pode *Classificar* o mapa através do *Sistema de classificação*.



Fig. 156: Classificação do Mapa

- Na secção da guia *Comentários*, pode postar seu comentário. Clique em *Add Comment*, insira o seu comentário e clique em *Submit Comment* para postá-lo.

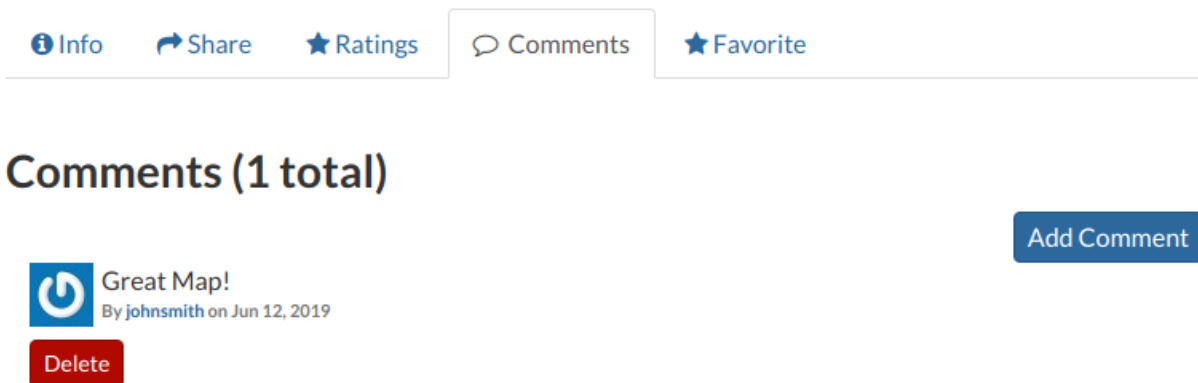


Fig. 157: Comentários do Mapa

O seu comentário será adicionado após o último comentário já existente. Se desejar removê-lo, clique no botão vermelho *Delete*.

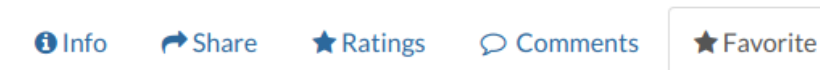
- Se desejar ter este mapa nos seus *Favoritos* (veja *Atualização do Perfil*), abra a aba *Favoritos* e clique em *Add to Favorites*.

Ferramentas de Mapa

No lado direito da página *Informações do Mapa*, existem algumas ferramentas que podem ajudá-lo a gerir seus mapas. Neste parágrafo, irá aprender como encontrar e recuperar informações sobre mapas.

O seguinte é uma lista de acções que pode executar para completar esta tarefa:

- clique no botão *Download Map*, para descarregar o mapa como imagem;
- clique no botão *Metadata Detail* para ver os metadados do mapa, consulte *Metadados de Mapas*;
- clique no botão *Editing Tools* para aceder a muitas ferramentas de edição. Essas funcionalidades serão explicadas na secção *Exploração de Mapas*;



Favorite

Add to Favorites

[Go to Favorites](#)

Fig. 158: Os seus Mapas Favoritos

- clique no botão *View Map* para abrir o mapa, veja a secção *Exploração de Mapas* para mais detalhes;
- veja a secção *Camadas do Mapa* para saber quais camadas são usadas pelo mapa (pode abrir a *Página de Camadas* clicando no seu nome, disponível apenas para camadas locais);
- clique no botão *Create a Map* da secção *Copiar este mapa* para duplicar o mapa;
- clique em *Publish Map WMS* da secção *Mapa WMS* para publicar camadas de mapas locais como grupo de camadas WMS;
- consulte a secção *Sobre* para conhecer o *Proprietário* do mapa, o utilizador de *Contacto* e o *Autor de Metadados*.

Metadados de Mapas

Os *Metadados de Mapas* podem ser explorados clicando no botão *Metadata Detail* na página *Informações do Mapa*.

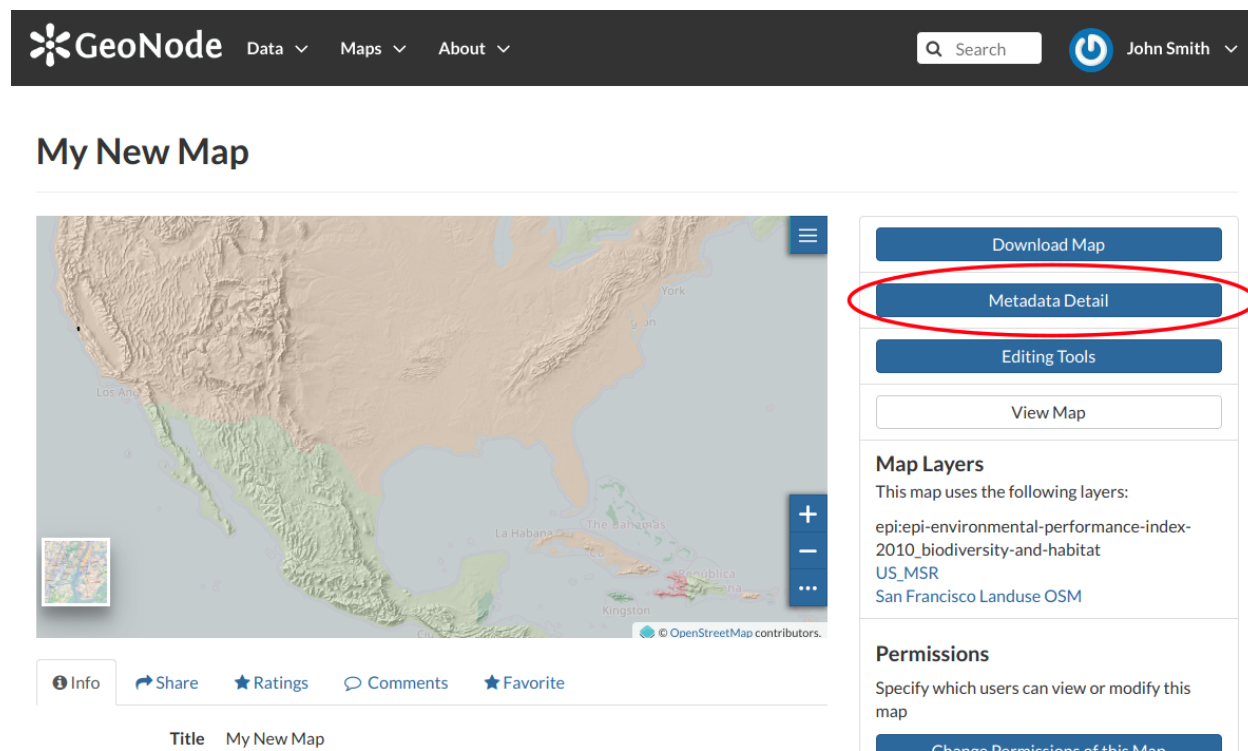


Fig. 159: O botão Detalhe de Metadados do Mapa

A página *Metadados do Mapa* abrirá.

Muita informação é exibida nesta página. Estas informações são agrupadas da seguinte forma:

- *Identificação* para identificar o mapa de forma única (Título, Licença, Data de Publicação e Regiões. Existem também, alguns sinais que indicam o estado do mapa, em particular se está Aprovado e/ou Publicado);
- o *Proprietário* do mapa;
- *Informação*, a Imagem de Identificação, Extensão Espacial, Sistema de Projecção e Extensão;
- *Recursos*, Idioma, Informações Suplementares e outras Informações;
- *Pontos de Contacto*, o utilizador disponível para entrar em contacto;
- *Referências*, ligações para o mapa e seus metadados;
- *Autor dos Metadados*, informações sobre o autor dos metadados.

Assistente de Metadados

Os metadados fornecem informações essenciais para a identificação e compreensão do mapa. Eles também tornam o mapa mais facilmente recuperável por meio das ferramentas de pesquisa.

Esses *Metadados* podem ser preenchidos por meio de um *Assistente* de três etapas em que você deve fornecer todas as informações obrigatórias para concluir o processo. Essas três etapas são descritas a seguir.

• Metadados Básicos

Na primeira etapa, o sistema pede que insira os seguintes metadados (os campos obrigatórios são destacados com contornos vermelhos):

- A *Miniatura* do mapa (clique em *Edit* para alterá-lo);
- O *Título* do mapa, que deve ser claro e compreensível;
- Um *Resumo*; breve resumo narrativo do conteúdo do Mapa

Nota: O painel *Resumo* permite inserir código HTML por meio de um editor de texto *wysiwyg*

- As *Datas de Criação/Publicação/Revisão* que definem o período de tempo coberto pelo mapa;
- As *Palavras-Chave*, que devem ser escolhidas dentro da lista disponível;
- A *Categoria* à qual o mapa pertence;
- O *Grupo* ao qual o mapa está vinculado.

Clique em *Next >>* para ir para a próxima etapa.

• Localização e Licenças

A lista a seguir mostra quais tipos de metadados que deve inserir (veja também a imagem abaixo):

- O *Idioma* da camada;
- A *Licença* do conjunto de dados;
- As *Regiões* cobertas pela extensão das camadas. As extensões propostas cobrem as seguintes escalas: global, continental, regional, nacional;
- A *Declaração de Qualidade de Dados* (explicação geral do conhecimento do produtor de dados sobre a linhagem de um conjunto de dados);


Data ▾
Maps ▾
About ▾

 Search

 John Smith ▾

Metadata : My New Map

Return to Map

Identification

Title	My New Map
License	Not Specified ⓘ
Publication Date	June 12, 2019, 8:17 a.m.
Regions	Global, Africa, Central Africa, West Africa, Pacific, Kiribati
Approved	Yes
Published	Yes
Featured	No

Owner

Name	John Smith (johnsmith)
email	john.smith@mail.com
Position	CEO and Founder
Organization	John Smith Foundation
Location	John Smith Avenue 12345 John Smith City John Smith District ZAF
Voice	123456789
Fax	987654321

Information

Identification Image	
Spatial Extent	---
Projection System	EPSG:3857
Extension x0	-20037397.023299999535084
Extension x1	666726.142827000003308
Extension y0	-5787726.067250000312924
Extension y1	20037397.023299999535084

Features

Language	English
Supplemental Information	No information provided

Contact Points

Name	John Smith (johnsmith)
email	john.smith@mail.com
Position	CEO and Founder
Organization	John Smith Foundation
Location	John Smith Avenue 12345 John Smith City John Smith District ZAF
Voice	123456789
Fax	987654321

References

Link Online	/maps/47
Metadata Page	/maps/47/metadata_detail

Metadata Author

Name	John Smith (johnsmith)
email	john.smith@mail.com
Position	CEO and Founder
Organization	John Smith Foundation
Location	John Smith Avenue 12345 John Smith City John Smith District ZAF
Voice	123456789
Fax	987654321

Metadata for My New Map

Completeness
 ✖ Check Schema mandatory fields
 50%

Edit

Settings

Mandatory

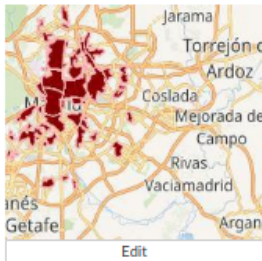
Mandatory

Optional

1

Basic Metadata

Thumbnail



Title

My New Map

Abstract

File Edit View Insert Format Tools

Table Help

↶ ↷ B I U S ...

asdasd

P 1 WORDS POWERED BY TINY

2

Location and Licenses

3

Optional Metadata

Free-text Keywords

Date type

Publication

Date

2020-11-20 12:12

Category

Group

Fig. 161: Metadados de Mapa Básicos

Metadata for My New Map

Completteness

✗ Check Schema mandatory fields

50%

Edit

Settings

Mandatory

Mandatory

Optional

1

2

3

Basic Metadata

Location and Licenses

Optional Metadata

Language

English

License

Not Specified

DOI

* Field declared Mandatory by the Metadata Schema

Attribution

* Field declared Mandatory by the Metadata Schema

Regions

✗ Global

Data quality statement

File Edit View Insert Format Tools

Table Help

↶ ↷ B I U S ...

P 0 WORDS POWERED BY TINY

* Field declared Mandatory by the Metadata Schema

Restrictions

* Field declared Mandatory by the Metadata Schema

Other constraints

File Edit View Insert Format Tools

Table Help

↶ ↷ B I U S ...

P 0 WORDS POWERED BY TINY

<< Back

Update

Next >>

Fig. 162: Localização e Metadados de Licenças para Mapas

- Potenciais *Restrições* na partilha de camadas.

Nenhum outro metadado obrigatório é necessário na próxima etapa, então, uma vez que os campos necessários tenham sido preenchidos, um botão verde *Done* ficará visível na tela. Clique em *Next >>* para ir para a próxima etapa ou *<< Back* para voltar à etapa anterior.

- **Metadados Opcionais**

Metadata for My New Map

Completiness
✖ Check Schema mandatory fields
50%

Edit Settings

Mandatory Mandatory Optional

1 2 3

Basic Metadata

Other, Optional, Metadata

Edition

Purpose

File Edit View Insert Format Tools

Table Help

↶ ↷ B I U ✂ ...

P 0 WORDS POWERED BY TINY

Supplemental information

File Edit View Insert Format Tools

Table Help

↶ ↷ B I U ✂ ...

No information provided

Location and Licenses

temporal extent start

temporal extent end

Maintenance frequency

Spatial representation type

Optional Metadata

Responsible Parties

Point of Contact

admin x

Responsible and Permissions

Owner

admin x

Metadata Author

admin x

Fig. 163: *Metadados de Mapa Opcionais*

As informações complementares são:

- A *Edição* do mapa;
- O *Propósito* do mapa e seus objectivos;

- Qualquer *Informação Suplementar* que possa fornecer uma melhor compreensão do mapa;
- A *Frequência de Manutenção* do mapa;
- O *Tipo de Representação Espacial*, o método utilizado para representar a informação geográfica no conjunto de dados;
- Os utilizadores que são *Responsáveis* pela camada, seu *Proprietário* e o *Autor* de seus metadados;

Se alguns metadados obrigatórios não forem preenchidos, a barra *Completeness* mostra uma mensagem em vermelho como a da imagem em baixo.

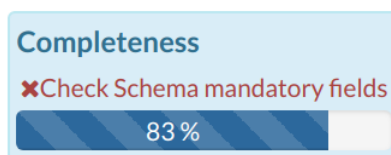


Fig. 164: Barra de Progresso de Completude

Edição Avançada de Metadados

A ferramenta de edição *Advanced Edit* permite alterar os metadados do mapa. Pode encontrar este botão no mapa *Ferramentas de Edição*.

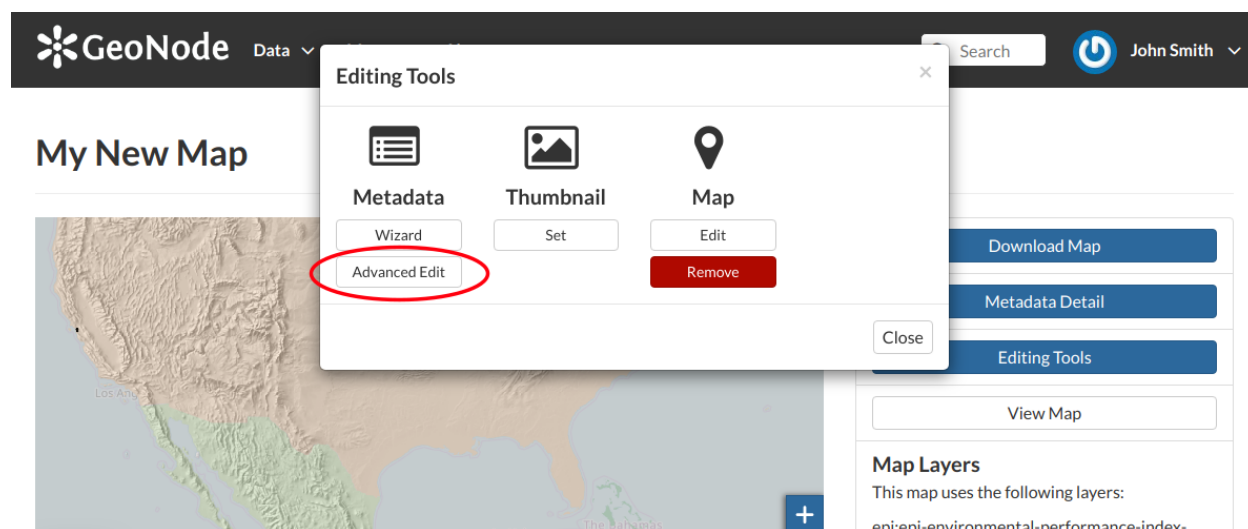


Fig. 165: O botão de Edição Avançada

Clique no botão para exibir a *Página de Edição Avançada de Metadados*. Essa página permite que edite todos os metadados da camada descritos no parágrafo anterior. Assim que terminar de editá-los, clique em *Update* para guardar suas alterações.

Alterando as Permissões do Mapa

Na secção *Informação do Mapa* deste guia, dissemos que pode ver seus mapas e todos os mapas publicados. No GeoNode, o sistema de gestão de permissões é na realidade mais complexo. Os administradores podem escolher quem pode fazer o quê em cada mapa. Os utilizadores podem gerir apenas os mapas que possuem ou os mapas que estão autorizados a gerir.

Por padrão, apenas os proprietários podem editar e gerir mapas, qualquer pessoa pode visualizá-los e baixá-los.

Para modificar as configurações de *Permissões do Mapa*, deve clicar no botão *Change the Layer Permissions* na *Página do Mapa*.

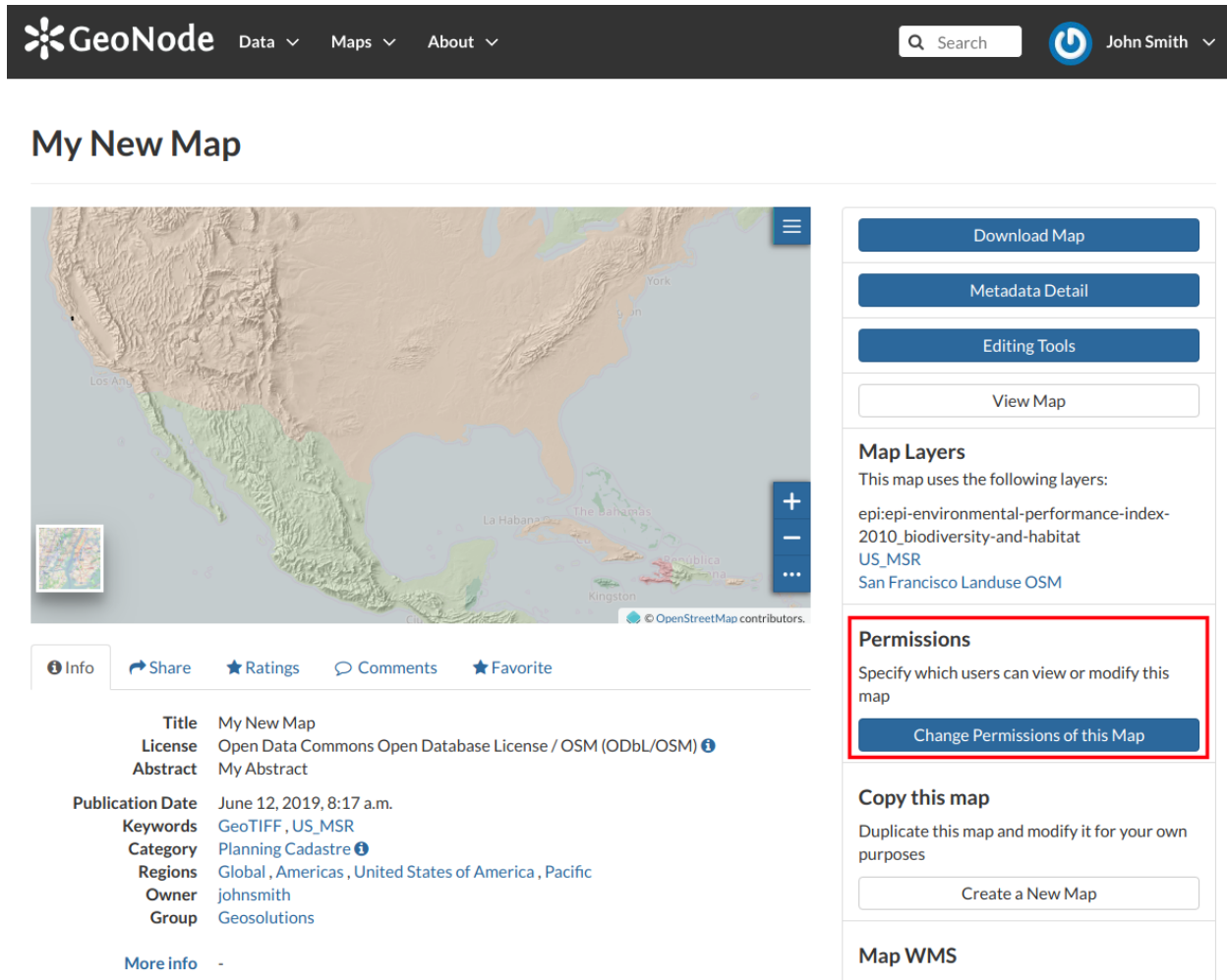


Fig. 166: *Altere as Permissões do Mapa*

Através do *Painel de Configurações de Permissões*, pode adicionar ou remover permissões para utilizadores e grupos. A imagem em baixo mostra um exemplo.

Pode definir os seguintes tipos de permissões:

- *Visualizar* permite ver o mapa;
- *Download* permite descarregar o mapa;
- *Alterar Metadados* permite alterar os metadados do mapa;

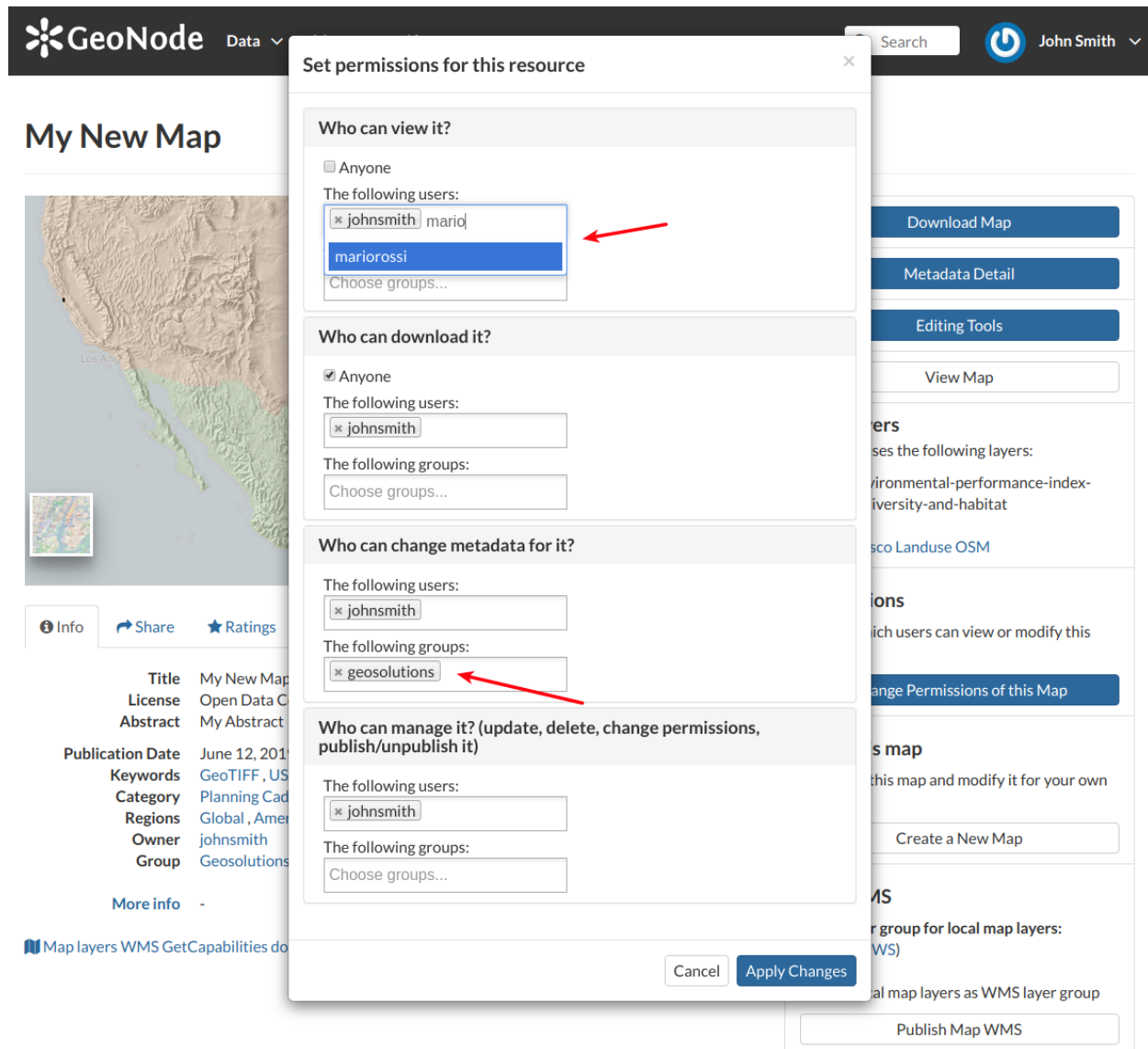


Fig. 167: Configurações de Permissões do Mapa para utilizadores e grupos

- *Gerir* permite actualizar, excluir, alterar permissões, publicar e cancelar a publicação do mapa.

Aviso: Ao atribuir permissões a um grupo, todos os membros do grupo terão essas permissões. Tenha cuidado ao editar permissões.

Clique em *Apply Changes* para guardar essas configurações.

Exploração de Mapas

A partir do link *Explore Maps* da barra de navegação pode aceder a página *Lista de Mapas* (veja [Mapas](#)). Seleccione um mapa de seu interesse e clique nele, a *Página do Mapa* será aberta.

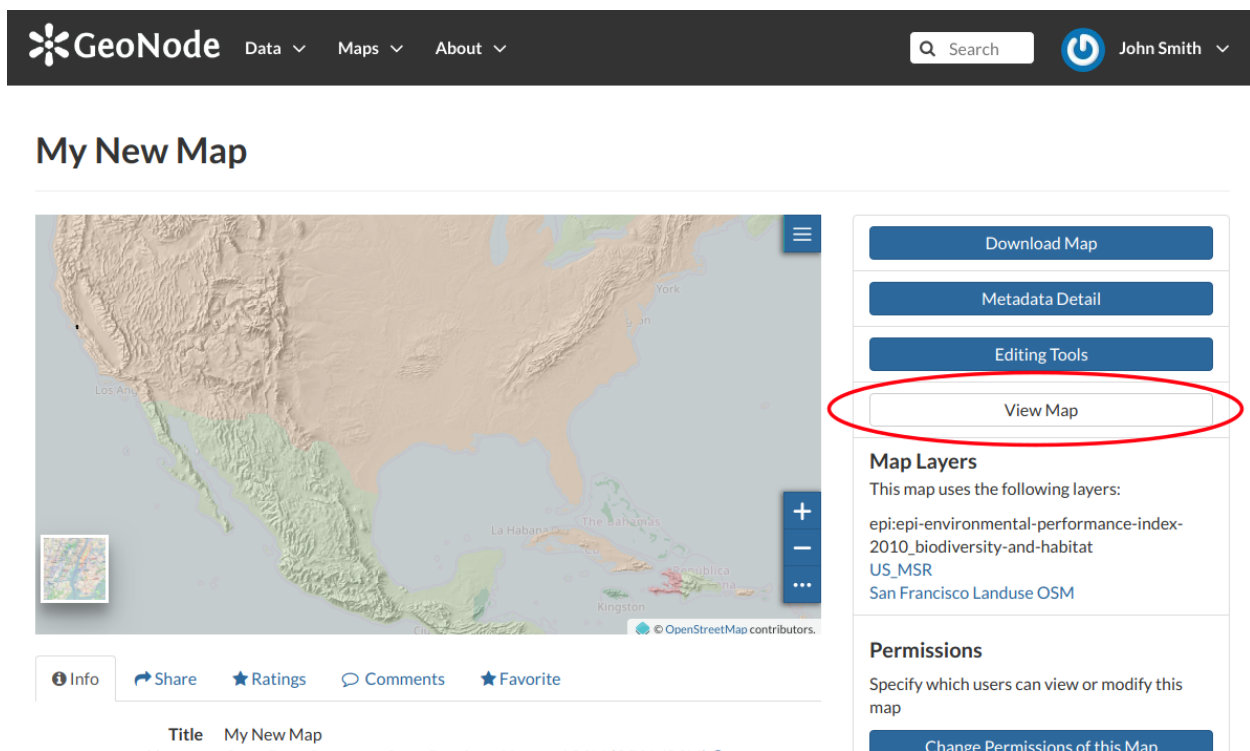


Fig. 168: O botão Exibir Mapa

Clique no botão *View Map* para abrir o *Visualizador de Mapas*.

O Visualizador de Mapas (baseado em [MapStore](#)) fornece as seguintes ferramentas:

- o *Índice (TOC)* para gerir o conteúdo do mapa;
- o *Comutador de mapa Base* para mudar o mapa base (veja os próximos parágrafos);
- a *Barra de Pesquisa* para pesquisar por localização, nome e coordenadas (veja o parágrafo abaixo);
- o *Ferramentas do Menu de Opções* que contém o link para a ferramenta *Imprimir*, para as camadas *Catálogo* e para a ferramenta *Medir*;
- a *Barra lateral* e suas ferramentas, como as ferramentas *Zoom* e *Obter Informações sobre Recursos*;
- as *Ferramentas de Rodapé* para gerir a escala do mapa, rastrear as coordenadas do rato e alterar o CRS (Sistema de Referência de Coordenadas).

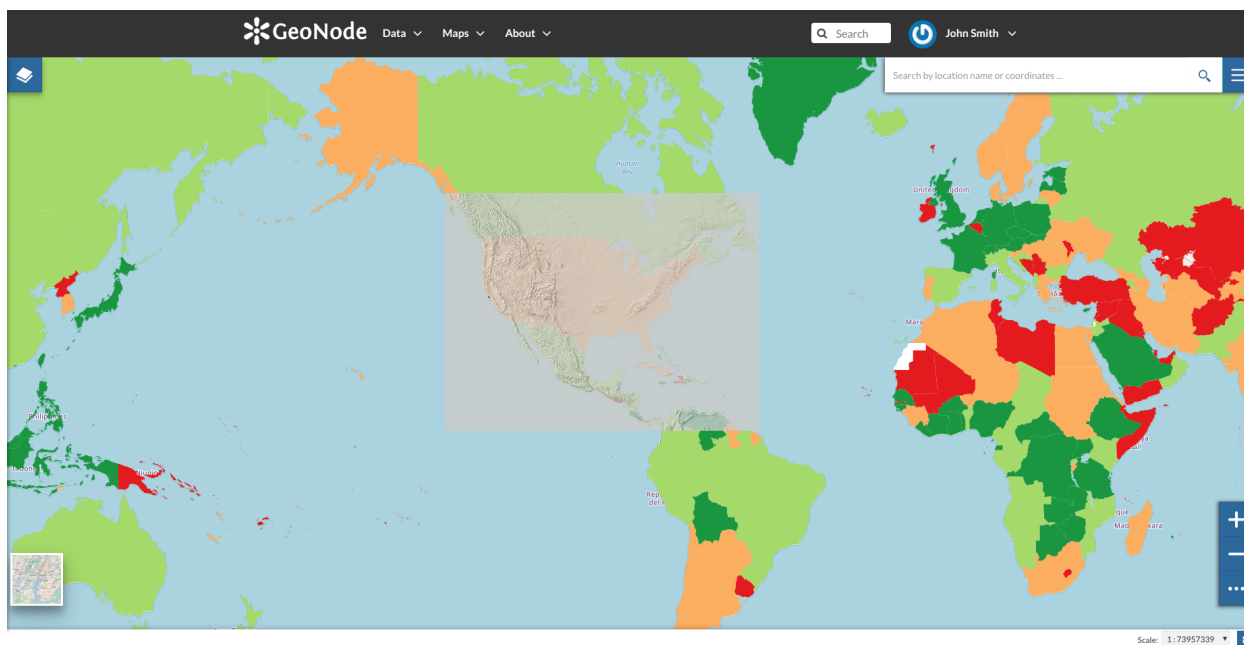


Fig. 169: A Visualização do Mapa

Índice (TOC)

No canto superior esquerdo, clique em  para abrir o *Índice*, abreviadamente *TOC* de agora em diante, do mapa. O *TOC* mostra todas as camadas envolvidas com o *Mapa* e permite gerir as suas propriedades e representações no mapa.

A partir do *TOC* pode:

- gerir a *Sobreposição* das camadas ;
- filtre a lista de camadas digitando texto no campo *Camadas de filtro*;
- adicione novas camadas do *Catálogo* clicando no botão *Add Layer*;
- gerir as propriedades das camadas como *Opacidade* (role o cursor de opacidade), *Visibilidade* (clique em  para tornar a camada invisível, clique em  para mostrá-la no mapa);
- gerir as *Configurações de Camada*, consulte o próximo parágrafo.

Selecione uma *Camada* da lista e clique nela, a *Barra de Ferramentas da Camada* deve aparecer no *TOC*.

A *Barra de Ferramentas* mostra muitos botões:

-  permite ampliar até a extensão da camada;
-  guia-o através da personalização de configurações de camada (consulte o próximo parágrafo);
-  para explorar os recursos da camada e seus atributos (mais informações em tabela-de-atributos);
-  para excluir camadas (clique em *Delete Layer* para confirmar a sua escolha);

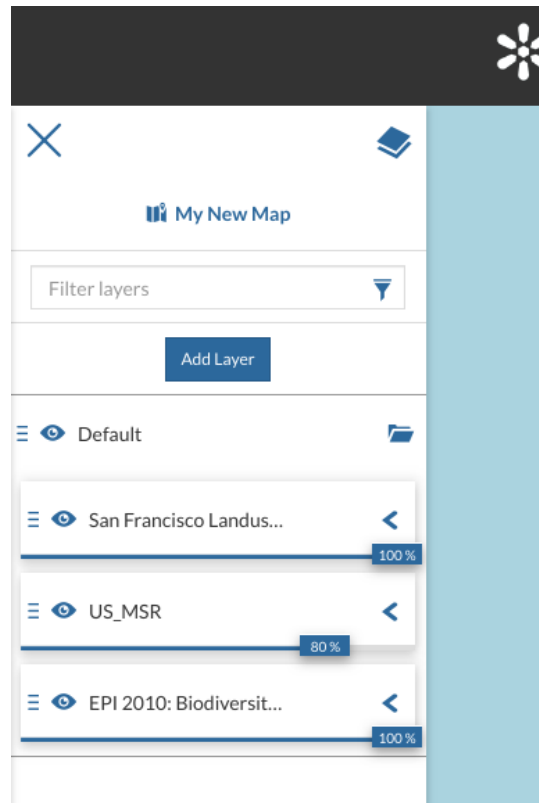


Fig. 170: O Índice (TOC)

Fig. 171: Rolagem da Camada de Opacidade

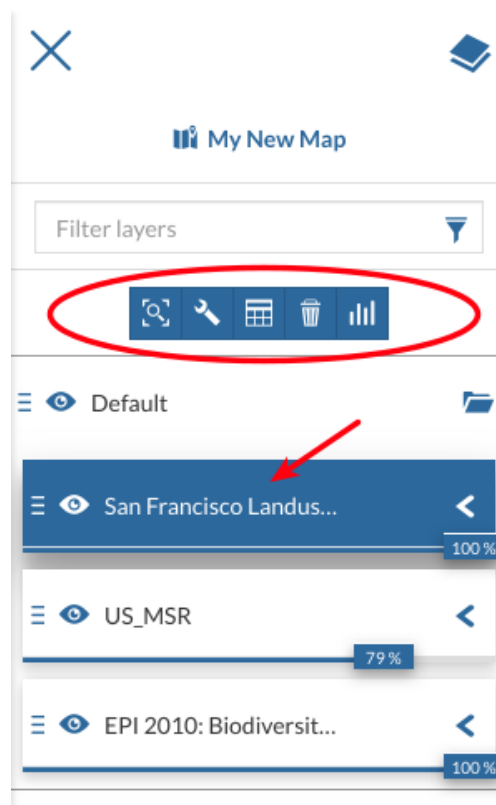


Fig. 172: A Barra de Ferramentas da Camada

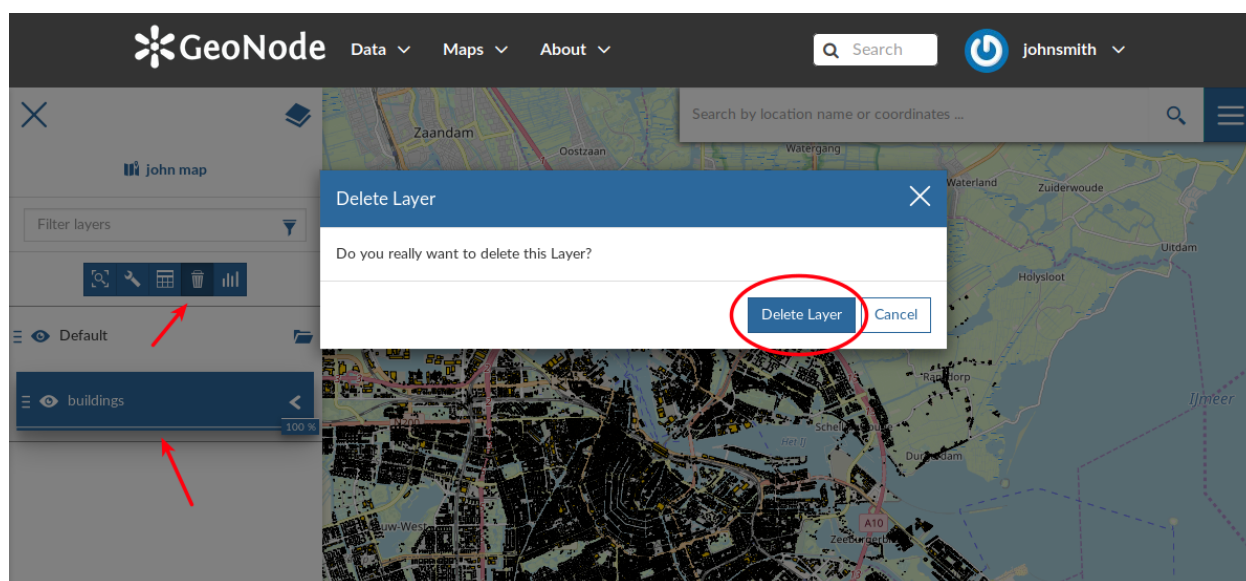
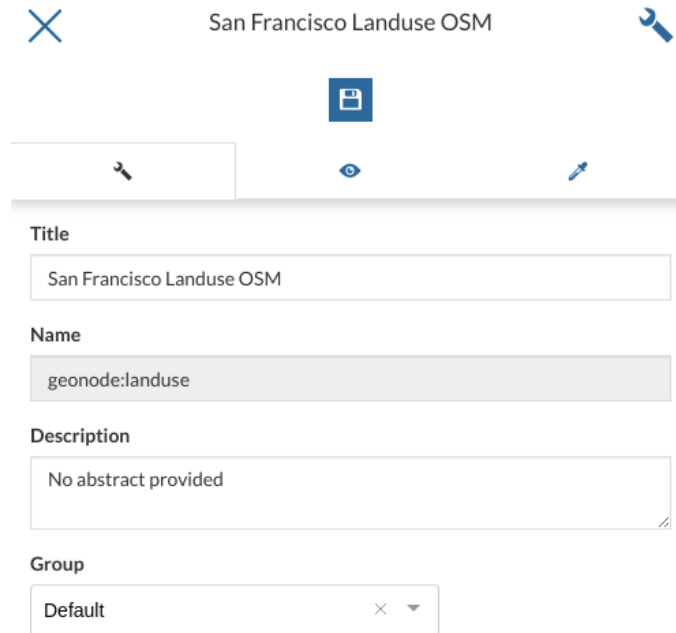


Fig. 173: Excluindo Camadas

-  para criar *Widgets* (veja *Criação de Widgets*).

Fazendo a Gestão de Configurações de Camada

O painel *Configurações de Camada* é semelhante ao painel abaixo.



The screenshot shows the 'San Francisco Landuse OSM' layer configuration panel. At the top, there is a title bar with a close button (X), the layer name 'San Francisco Landuse OSM', and a settings icon (wrench). Below the title bar is a toolbar with icons for settings, visibility, and editing. The main configuration area contains the following fields:

- Title:** A text input field containing 'San Francisco Landuse OSM'.
- Name:** A text input field containing 'geonode:landuse'.
- Description:** A text area containing 'No abstract provided'.
- Group:** A dropdown menu showing 'Default'.

Fig. 174: O Painel de Configurações de Camada

As *Configurações de Camada* são divididas em três grupos:

1. Configurações *Gerais*
2. Configurações do *Visualizador*
3. Configurações de *Estilo*

Na guia **Geral** do *Painel de Configurações*, pode personalizar a camada *Título*, inserir uma *Descrição* e alterar/criar o *Grupo de Camadas*.

Clique na guia **Display** para ver quais são as propriedades de aparência da camada que pode configurar.

O campo *Formato* permite que altere o formato de saída das solicitações WMS.

Pode definir um valor numérico de *Opacidade* usando o campo de entrada correspondente.

Pode, também, definir a camada como *Transparente*, decidir *Usar opções de cache* e usar *Bloco Único*.

A terceira guia é **Estilo**. Ao clicar nela, um *Editor de Estilos* avançado permite criar novos estilos e modificar ou excluir um existente. Veja a secção *Modelação de Camada* para ler mais.

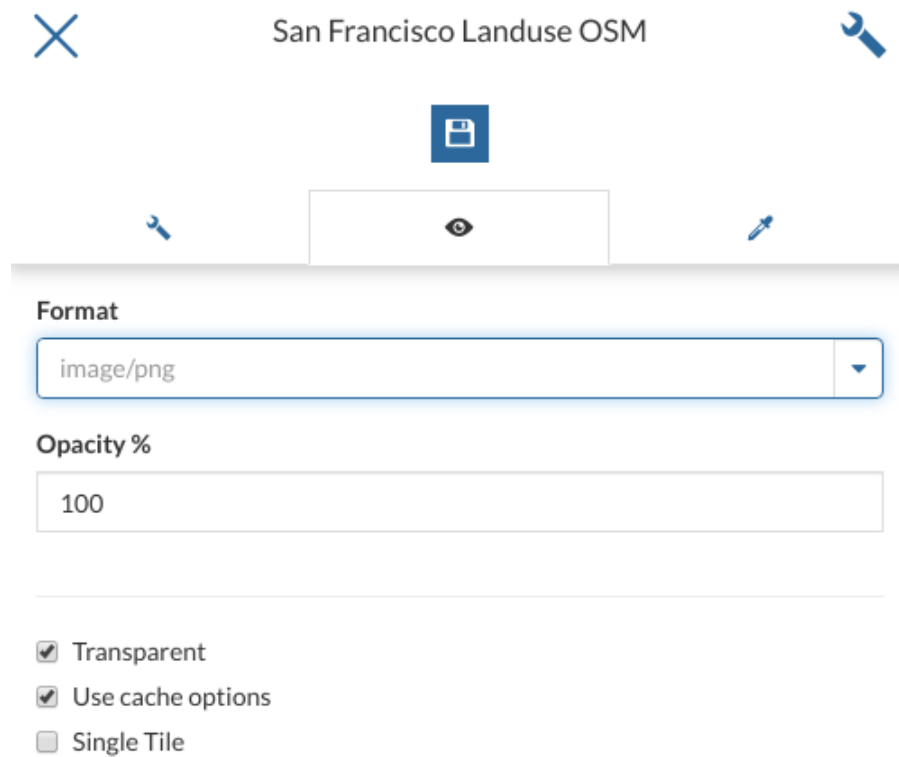


Fig. 175: O Painel de Configurações de Exibição da Camada

Tabela de Atributos

Ao clicar no botão  do *Índice (TOC)*, o painel *Tabela de Atributos* abre na parte inferior da página *Mapa*.

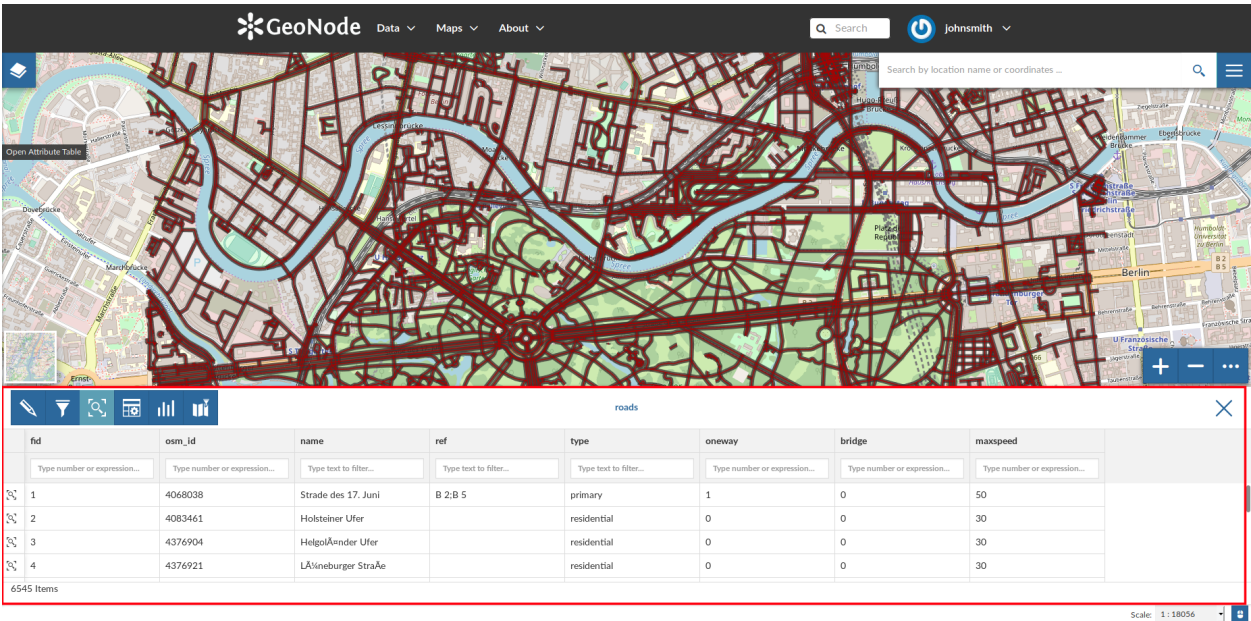


Fig. 176: O Painel da Tabela de Atributos

Neste painel pode navegar pelos recursos da camada, ampliar suas geometrias clicando no ícone  ícone e explorar seus atributos.

As *Tabelas de Atributos* possuem uma linha para cada característica pertencente à camada e uma coluna para cada atributo que descreve a característica.
Cada coluna tem um *Filtro* de campo de entrada por meio do qual pode filtrar os recursos com base em algum valor ou expressão (dependendo do tipo de dados do campo).

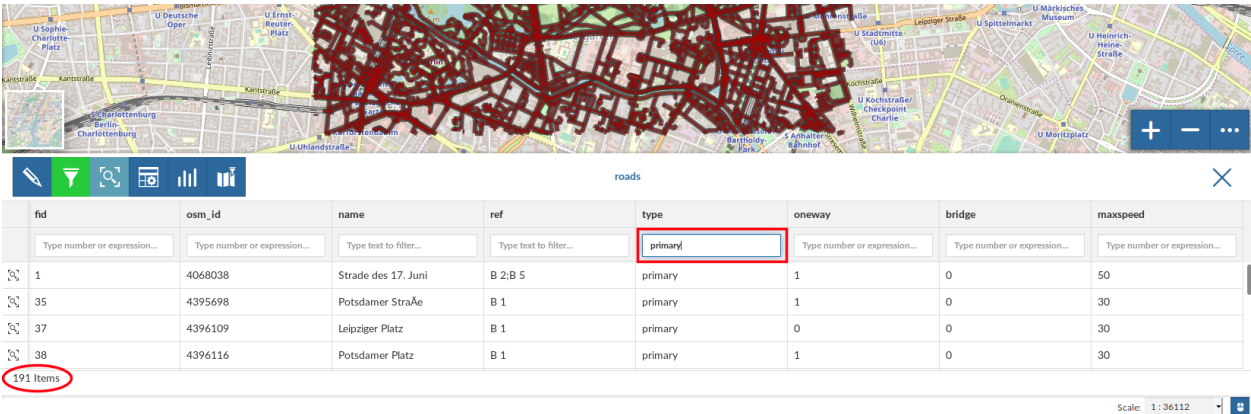


Fig. 177: Recursos de Filtragem por Atributo

O painel *Tabela de Atributos* contém uma *Barra de Ferramentas* que disponibiliza algumas funcionalidades úteis.

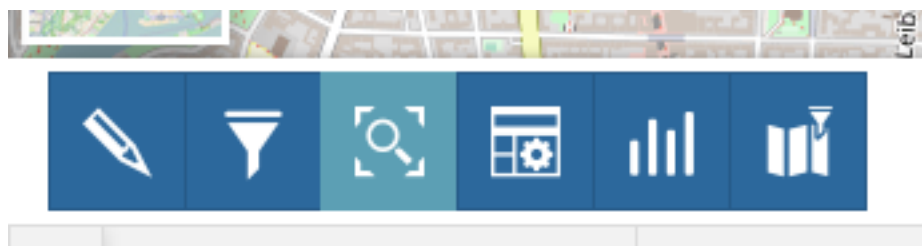


Fig. 178: A Barra de Ferramentas da Tabela de Atributos

Essas funcionalidades são:

- *Modo de Edição*
Clicando em  pode iniciar uma sessão de edição. Permite adicionar novos recursos, excluir ou modificar os existentes, editar geometrias. Veja a seção *layer-data-editing* para mais informações.
- *Busca Avançada*
Clique em , irá surgir um novo painel. Esse painel permite filtrar recursos de muitas maneiras diferentes. Esta funcionalidade será explicada em detalhe na secção *Busca Avançada*.
- *Zoom na extensão da página*
Clique em  para ampliar até a extensão da página.
- *Ocultar/mostrar colunas*
Ao clicar em  outro painel aparece dentro da *Tabela de Atributos*. Através desse painel pode escolher quais colunas deseja ver, veja a imagem em baixo.

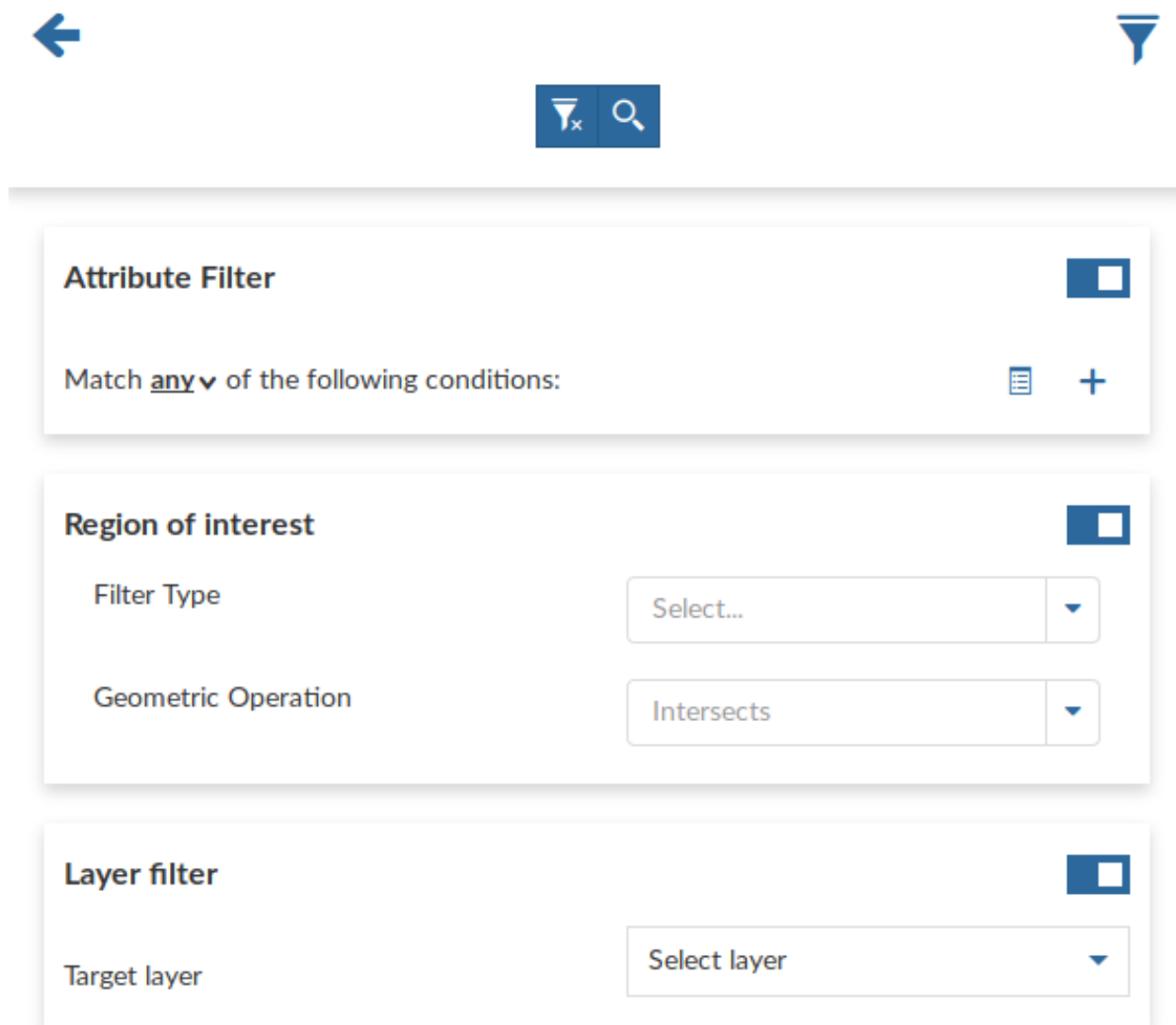
Fig. 179: Ocultar/Mostrar Colunas da Tabela de Atributos

- *Crie um gráfico*
Através do botão  pode abrir o painel *Widgets de gráfico* onde muitas funcionalidades para descrever e visualizar os dados da camada estão disponíveis (veja *Criação de Widgets*).
- *Sincronizar mapa com filtro*
Clique no ícone  para sincronizar o mapa com o filtro.

Busca Avançada

Como mencionado antes, o GeoNode permite uma filtragem espacial e baseada em atributos. Ao clicar em  na camada *Tabela de Atributos*, abre o painel *Pesquisa Avançada*, mostrando três funcionalidades de filtragem diferentes:

- Na secção **Filtro de Atributos**, pode compor uma série de condições sobre os atributos da camada. Clique em  para inserir uma nova condição em branco. Seleccione o atributo de seu interesse, seleccione um operador



The image shows the 'Advanced Search' (Busca Avançada) interface in GeoNode. At the top, there is a navigation bar with a back arrow on the left, a central search icon (funnel with a magnifying glass), and a filter icon (funnel) on the right. Below this, there are three main filter sections, each with a toggle switch on the right:

- Attribute Filter**: Toggled on. It contains the text 'Match any of the following conditions:' and a '+ ' icon to add more conditions.
- Region of interest**: Toggled on. It contains two dropdown menus:
 - Filter Type**: Set to 'Select...'.
 - Geometric Operation**: Set to 'Intersects'.
- Layer filter**: Toggled on. It contains one dropdown menu:
 - Target layer**: Set to 'Select layer'.

Fig. 180: *Busca Avançada*

e digite um valor de comparação. Você pode agrupar condições através do botão *Adicionar Grupo* . Clique em  para realizar a pesquisa.

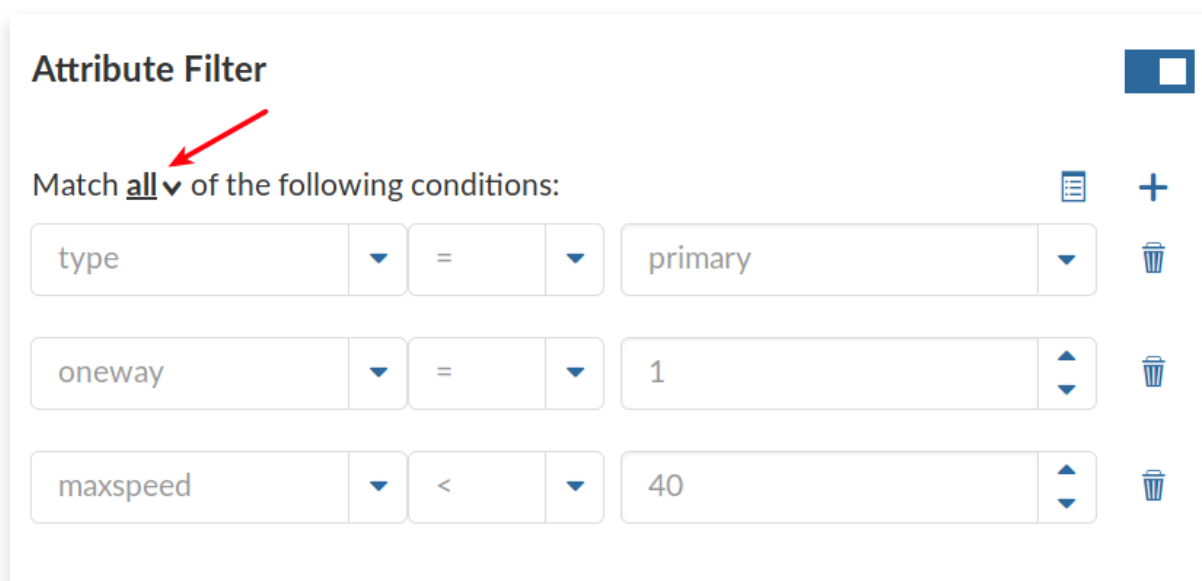


Fig. 181: *Filtragem por Atributos*

Pode também decidir se *Todas* as condições devem ser correspondidas ou se apenas *Qualquer* ou *Nenhuma* delas (veja a seta vermelha na imagem em cima).

- A filtragem de **Região de interesse** permite filtrar recursos que têm alguma relação com uma região espacial desenhada no mapa.

Selecione o *Tipo de Filtro* (Círculo, Janela de Visualização, Polígono ou Rectângulo), desenhe a região espacial de interesse no mapa, selecione uma *Operação Geométrica* (Cruzamento, Caixa Delimitadora, Contém ou Está contido) e clique em .

- Por meio do **Filtro de Camada**, pode seleccionar apenas os recursos que atendem a algumas condições em outras camadas do mapa. Pode também adicionar condições aos atributos dessas camadas.

Pode ler mais sobre a *Tabela de Atributos* e a *Pesquisa Avançada* no [MapStore2 Documentation](#).

Criação de Widgets

Widgets são elementos gráficos que descrevem os dados das camadas. Eles podem ser de diferentes tipos, como *Gráficos*, *Textos*, *Tabelas* e *Contadores*. Por meio do  do *Índice (TOC)* pode abrir o painel *Widgets*.

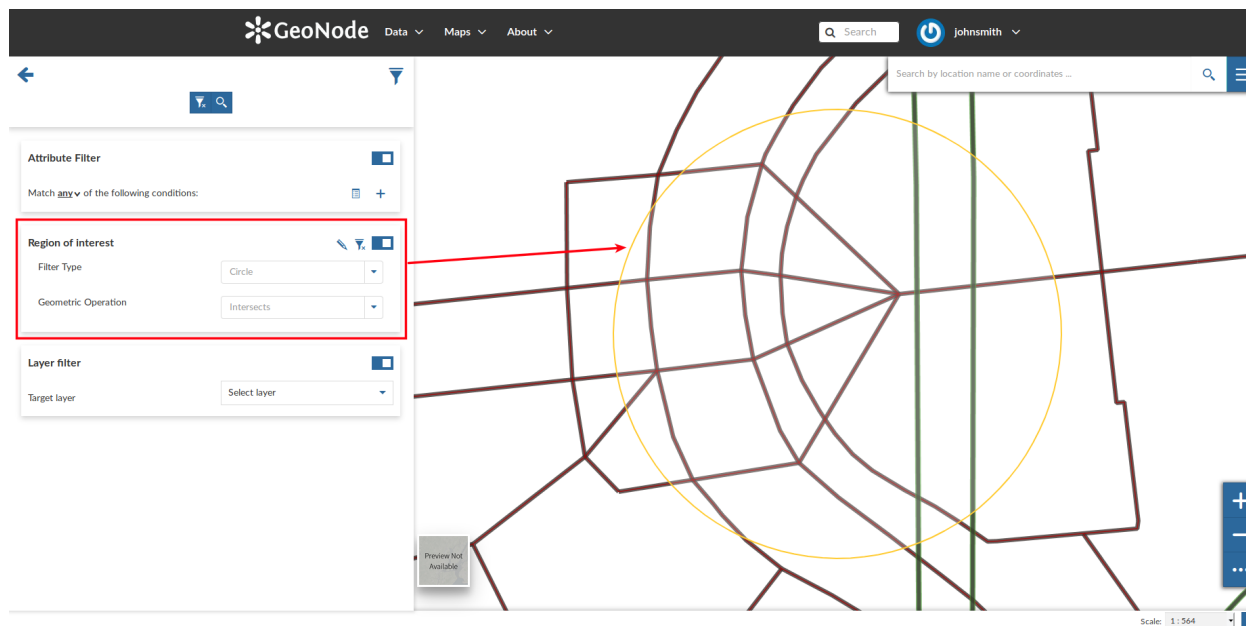


Fig. 182: Filtragem por Região de Interesse

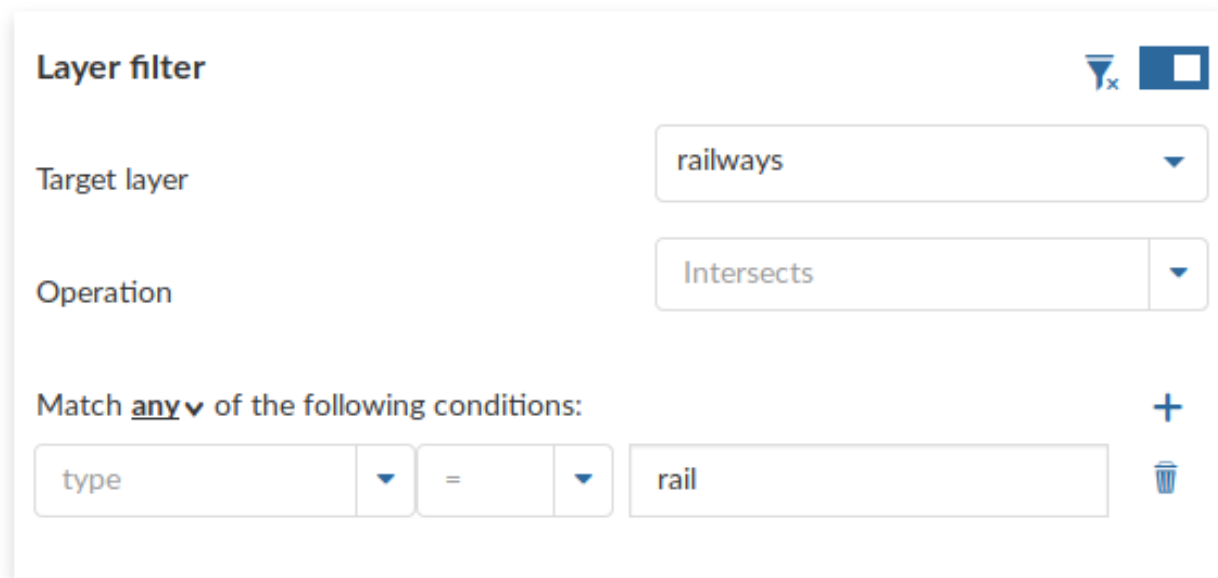


Fig. 183: Filtragem de Camada



Widget



Select the widget type



Chart

add a chart



Text

add a text area



Table

add a table



Counter

add a counter

Fig. 184: Criação de Widgets

Widgets de Gráfico

Chart Widgets são representações gráficas dos dados da camada. Podem ser do tipo *Gráfico de Barras*, *Gráfico Circular* ou *Gráfico de Linhas* conforme mostrado na imagem em baixo.

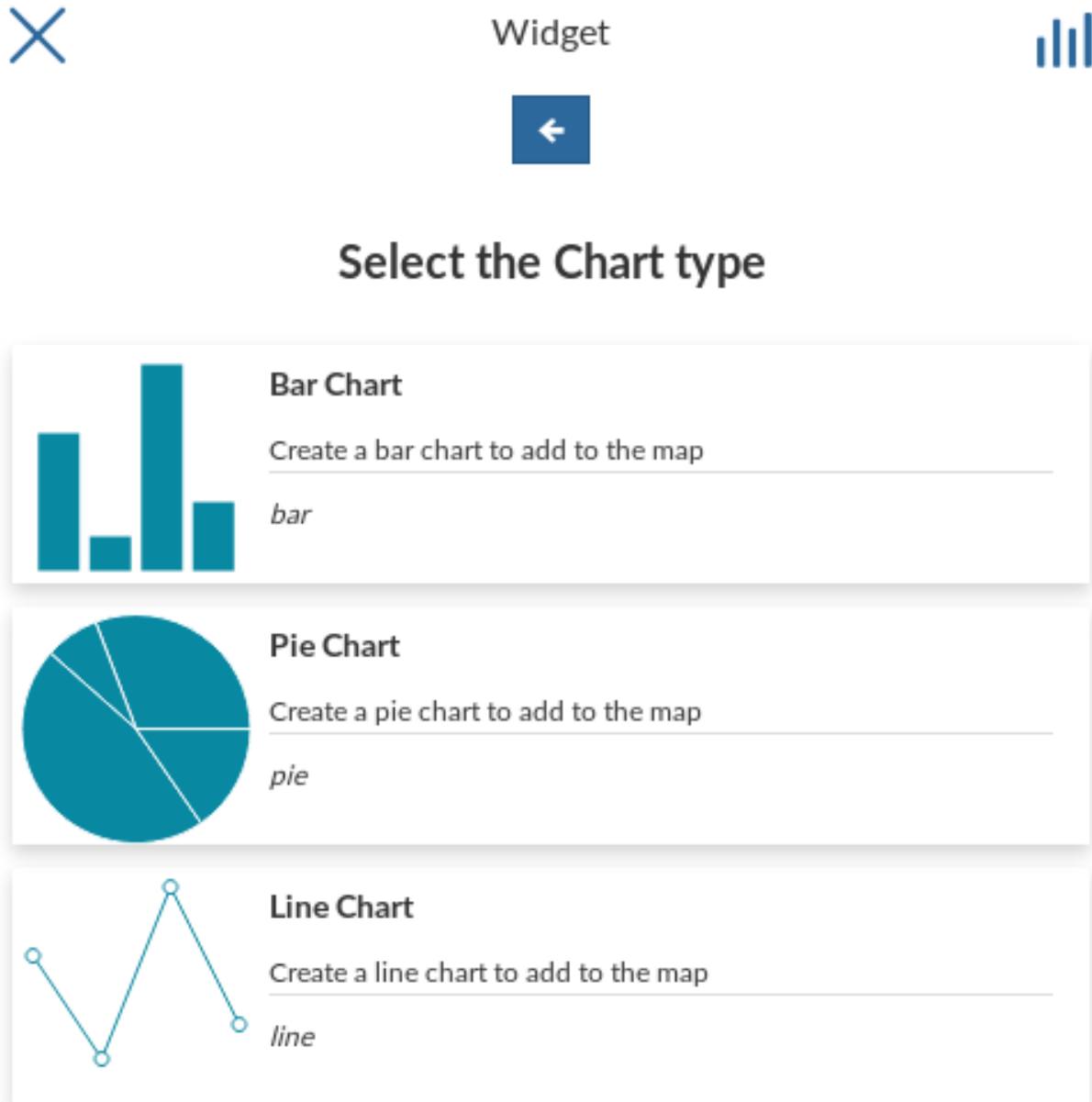


Fig. 185: *Widgets de Gráfico*

Vamos criar um novo **Gráfico de Barras**.

Clique em *Gráfico de Barras* e seleccione o *Atributo X*, o *Atributo Y*, a *Operação* e a *Cor* de sua preferência. Pode também, exibir a *Legenda*, *Ocultar o eixo Y*, *Ocultar a tabela* e decidir qual *Rótulo* será exibido na legenda.

Agora pode filtrar os dados a serem considerados para o gráfico clicando em . Não precisamos de nenhum filtro, então clique em  para configurar outras opções de widget. Insira um *Título* e uma *Descrição* e clique em *Guardar* .

Fig. 186: Criação de Gráficos de Widgets

O ícone verde  significa que o gráfico está ligado ao ecrã de visualização.

Expandindo o menu de opções do widget, você pode *Mostrar os dados renderizados*, *Editar* o widget ou *Excluir*, *Guardar* os dados como um arquivo CSV ou *Exportar* a imagem do gráfico.

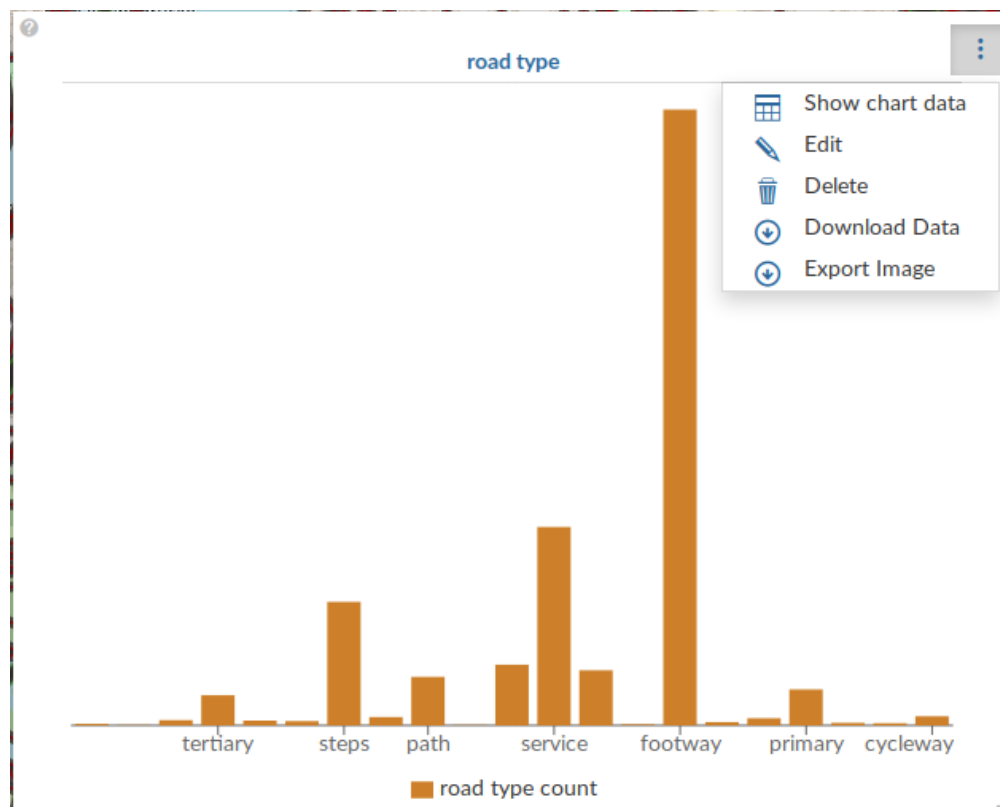


Fig. 187: Opções de Widgets de Gráfico

Widgets de Texto

Se seleccionou *Texto* no painel *Widgets*, poderá criar *Widgets de texto*. Adicione um *Título* e o texto descritivo desejado e clique em .

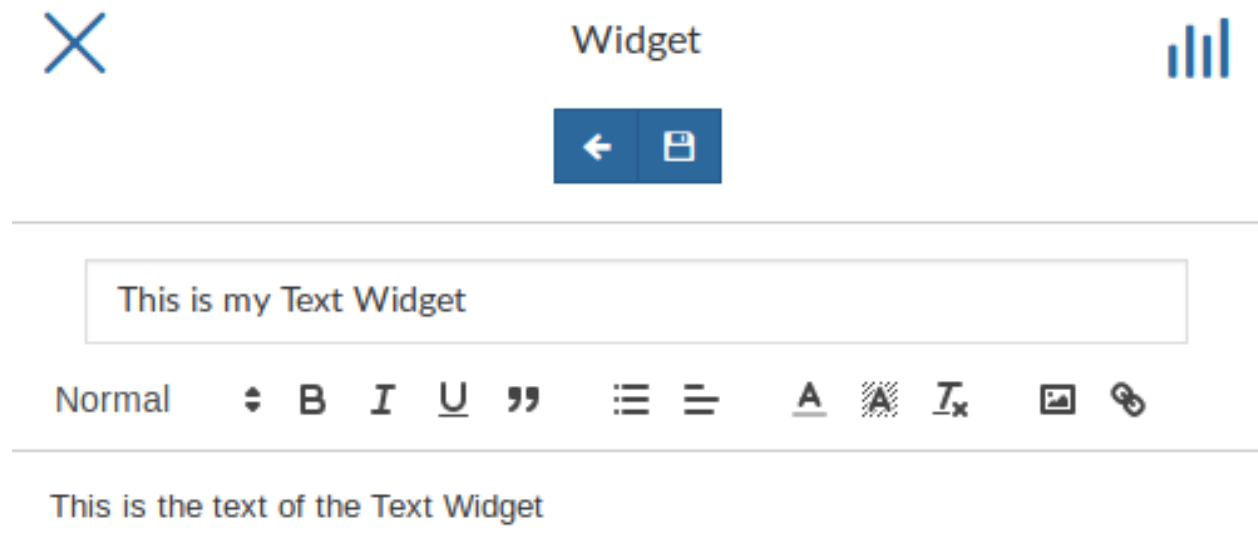


Fig. 188: Criação de *Widgets de Texto*

O widget resultante se parece com o seguinte.

Widgets de Tabela

Através dos *Widgets de Tabela* pode adicionar a tabela-de-atributos da camada ao mapa. Pode decidir mostrar um subconjunto dos recursos, por meio de filtros, e pode seleccionar uma ou mais colunas/atributos.

Portanto, escolha em quais atributos está interessado e clique em .

Insira *Título* e *Descrição* (opcional) e clique em . O exemplo em baixo mostra o *Widget de Tabela* no mapa.

Widgets de Contador

Counter Widgets são representações numéricas de alguns atributos. Por exemplo, pode representar o limite de velocidade média em uma rede rodoviária.

Clique em , insira *Título* e *Descrição* e clique em .

O visualizador de mapas GeoNode é baseado em [MapStore](#), veja [MapStore Documentation](#) para mais informações.

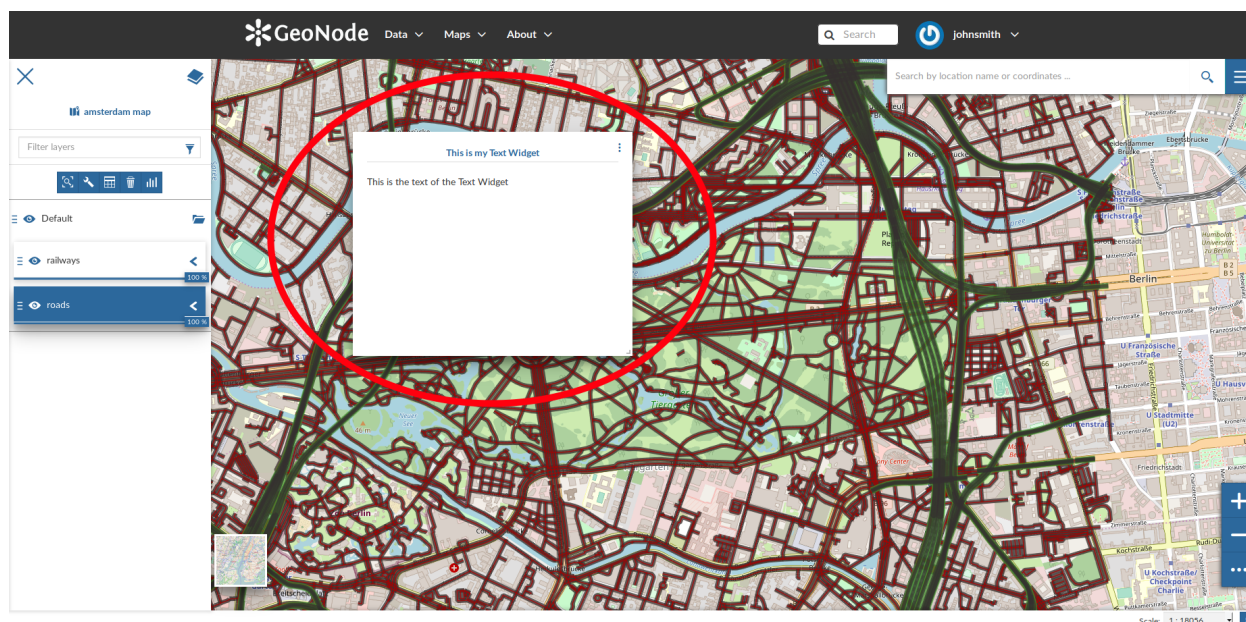


Fig. 189: O meu Widget de Texto

Linha do tempo

O GeoNode pode gerir camadas com uma *dimensão de tempo*. Essas camadas vectoriais podem variar seus dados ao longo do tempo, portanto, é útil representar essa variação no mapa.

O visualizador de mapa baseado em [MapStore](#) usado no Geonode, disponibiliza a ferramenta **Linha de Tempo** que permite observar a evolução das camadas ao longo do tempo, para inspeccionar a configuração da camada num dado instante de tempo específico e para visualizar as diferentes configurações de camada dinamicamente por meio de animações (consulte a [Documentação do MapStore](#) para mais detalhes).

Aviso: A linha do tempo na realidade funciona apenas com a extensão WMTS-Multidim (o tempo WMS em capacidades não é totalmente suportado).

Ao carregar uma camada temporal para o mapa, a *Linha do tempo* abre automaticamente.

No lado esquerdo do painel *Linha do tempo*, pode definir o valor de tempo no qual deseja observar os dados. Pode digitá-lo directamente preenchendo os campos de entrada correspondentes ou usando as setas para cima/para baixo.

Do outro lado estão os botões responsáveis para gerir as animações.

Em particular, pode *Reproduzir* a animação clicando em , voltar ao instante anterior através de , avançar para a próxima etapa usando  e parar a animação clicando em .

O painel *Linha do Tempo* pode ser expandido por meio do botão .

A secção expandida do painel *Linha do tempo* contém a *Lista de camadas de tempo* e um *Histograma* que mostra:



Fig. 190: Colunas de Widgets de Tabela

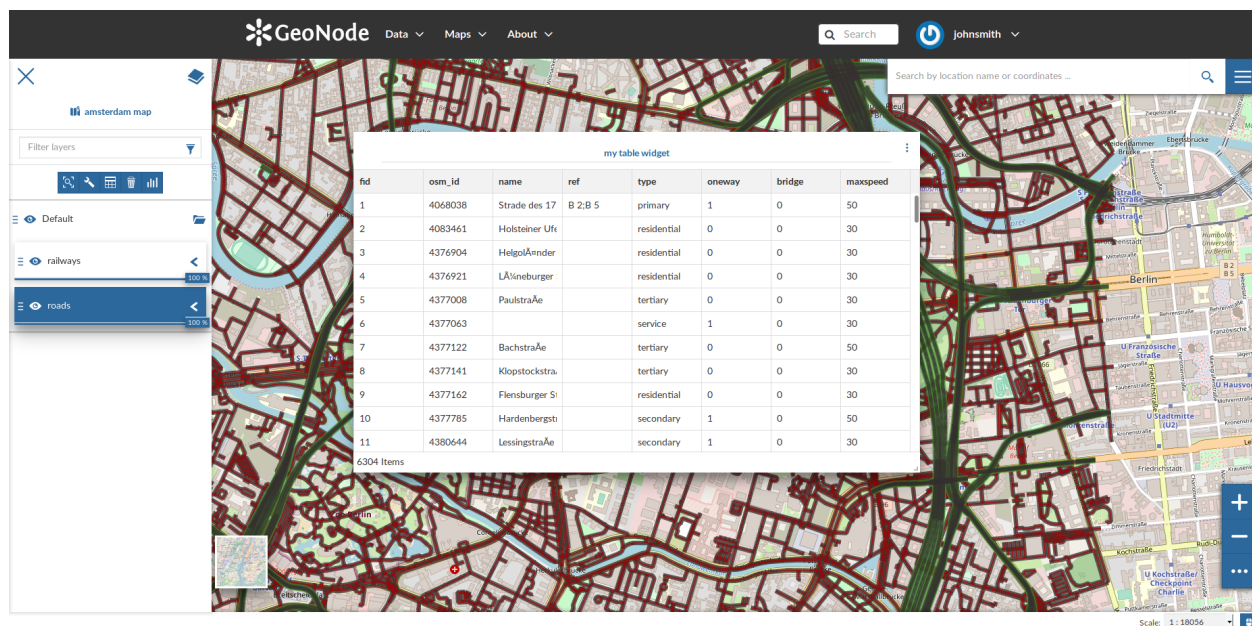


Fig. 191: Widget de Tabela

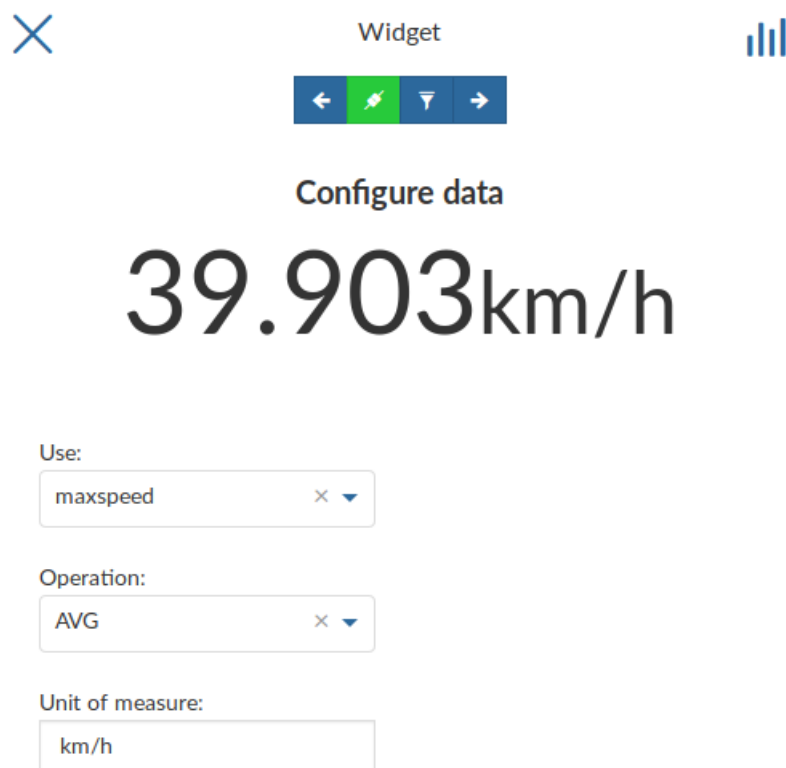


Fig. 192: Criação de Widget de Contador

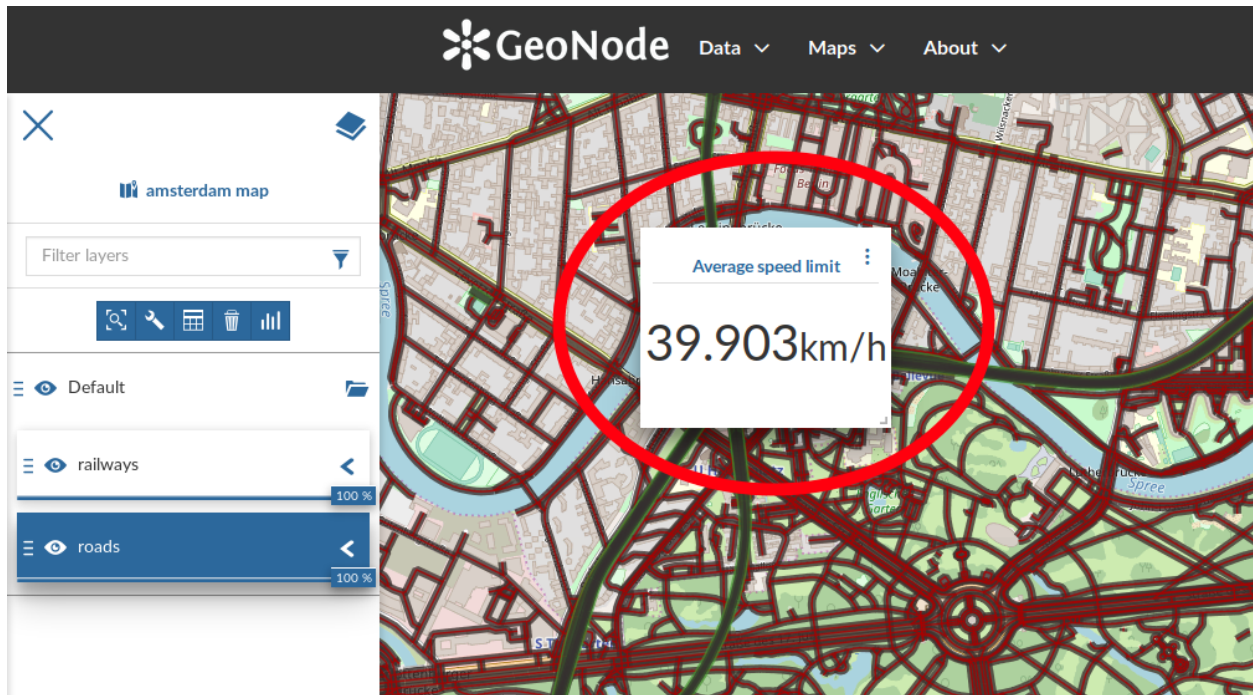


Fig. 193: Widget de Contador

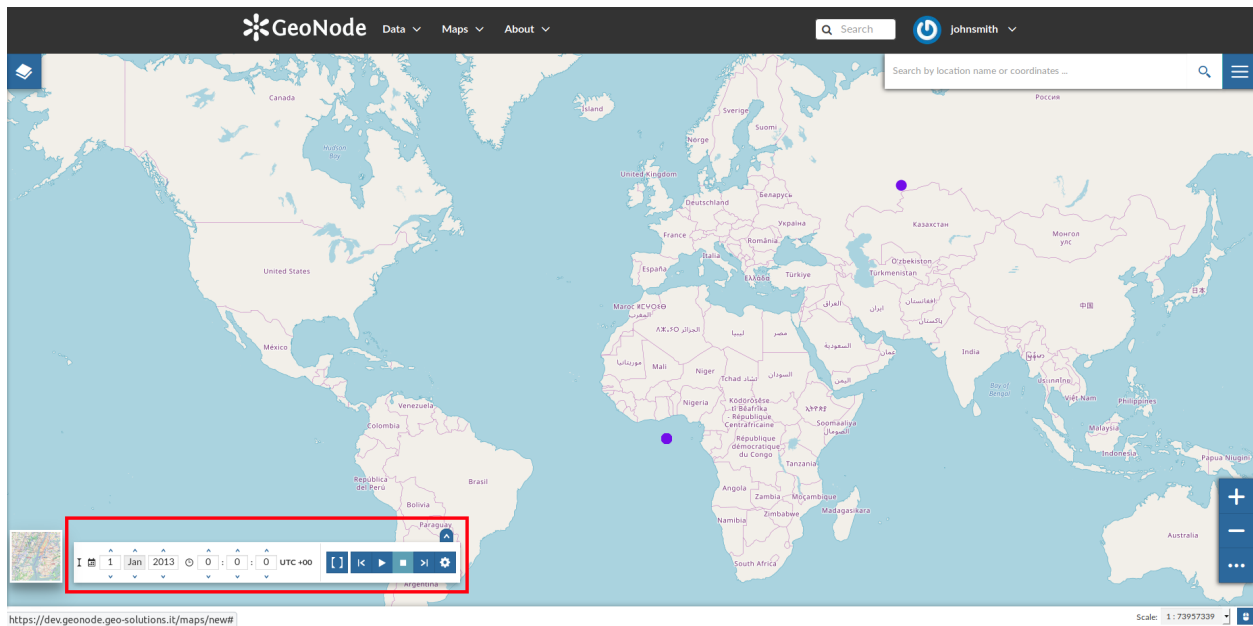


Fig. 194: A Linha do Tempo



Fig. 195: *Os Botões de Controle de Tempo*

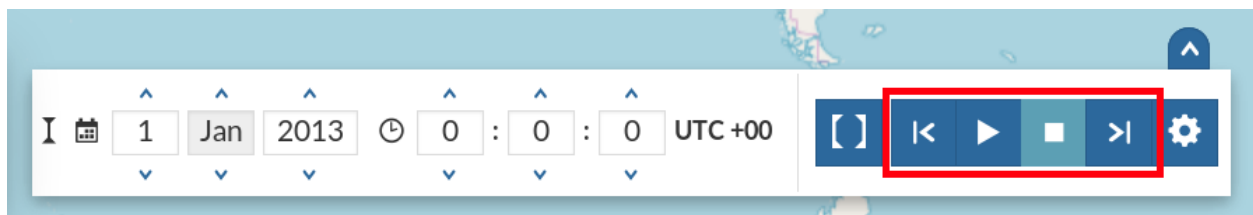


Fig. 196: *Os Botões de Controle de Animação*

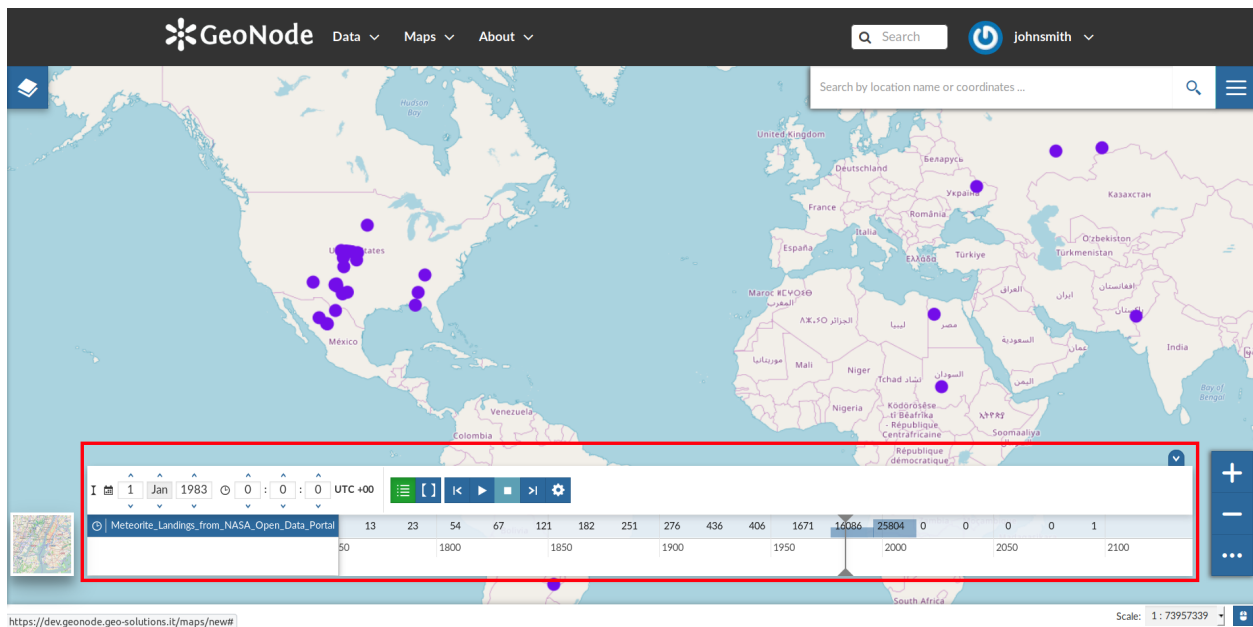


Fig. 197: *The Expanded Timeline*

- a distribuição dos dados ao longo do tempo

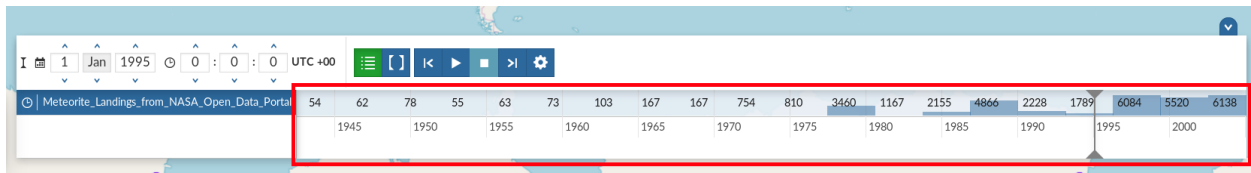


Fig. 198: O Histograma da Linha do Tempo

- o Cursor de Tempo

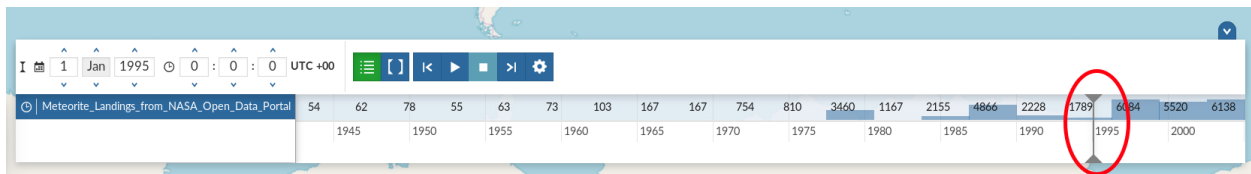


Fig. 199: O Cursor de Tempo

Pode mostrar/ocultar a lista de camadas clicando em  (está activo por padrão).

Por meio da função *Intervalo de Tempo*, pode observar os dados num intervalo temporal finito. Clique em  e defina os tempos inicial e final para usá-lo.

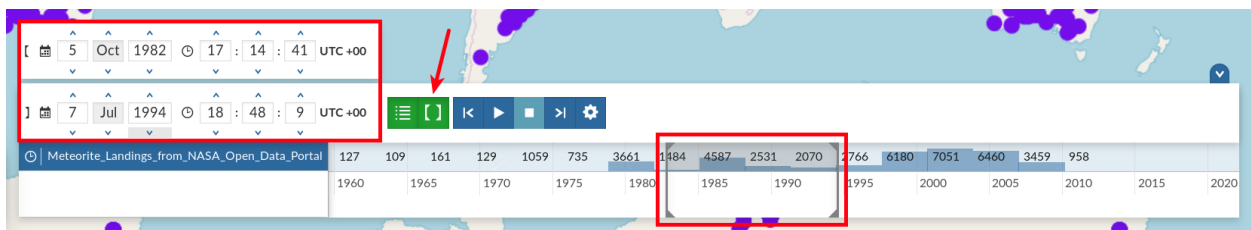


Fig. 200: As Configurações de Intervalo de Tempo

Animações

A *Linha do tempo* permite que veja as configurações de dados (uma para cada vez em que os dados são definidos) por meio de sequências ordenadas de etapas.

Como dito anteriormente, pode reproduzir a *Animação* resultante clicando no botão reproduzir . Os dados da camada exibidos no mapa serão alterados de acordo com o tempo alcançado pelo cursor no *Histograma*.

Clicando em  pode gerir algumas *Configurações de Animação*.

Pode activar *Pular para a camada de guia* para que o cursor de tempo se encaixe nos dados da camada seleccionada. Pode também configurar a *Duração do quadro* (por padrão, 5 segundos).

Se a opção **Ajustar à camada de guia** estiver desabilitada, pode forçar a etapa da animação a ser um valor fixo.

Timeline Settings

Snap to guide layer ? ☐

Playback Settings

Frame Duration

^

v

s

Animation Step ?

^

v

Day

Animation Range ☐

Follow the animation ? ☐

Fig. 201: As Configurações da Linha do Tempo

A opção *Intervalo de Animação* permite definir um intervalo temporal dentro do qual o cursor de tempo pode se mover. Veja o seguinte gif para compreender melhor como a *Animação* funciona ou verifique na [Documentação do MapStore](#).

Fig. 202: A Animação da Linha do Tempo

Ferramentas do Menu de Opções

No canto superior direito do *Mapa* existe o botão *Menu Burger* . Clique nele para abrir o painel *Opções do Mapa*.

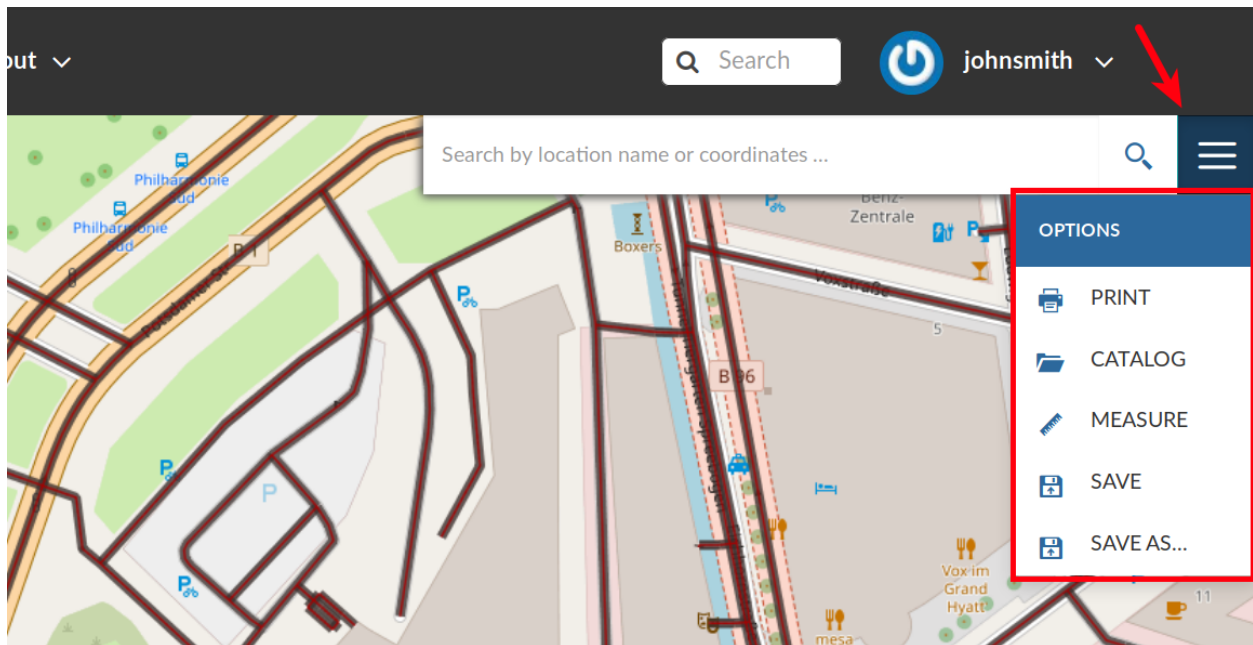


Fig. 203: O Menu de Opções do Mapa

Explicaremos estas ferramentas com mais detalhes nos próximos parágrafos.

Imprimindo um Mapa

O visualizador de mapa do GeoNode baseado em [MapStore](#) permite imprimir o seu mapa com um layout personalizado.

Clique na opção *PRINT* do *Menu de Opções do Mapa*, a **Janela de Impressão** será aberta.

Nessa janela, você pode:

- inserir *Título e Descrição*;
- escolher a *Resolução* em ppp;
- personalizar o *Layout*
 - o *Tamanho da folha* (A3, A4);

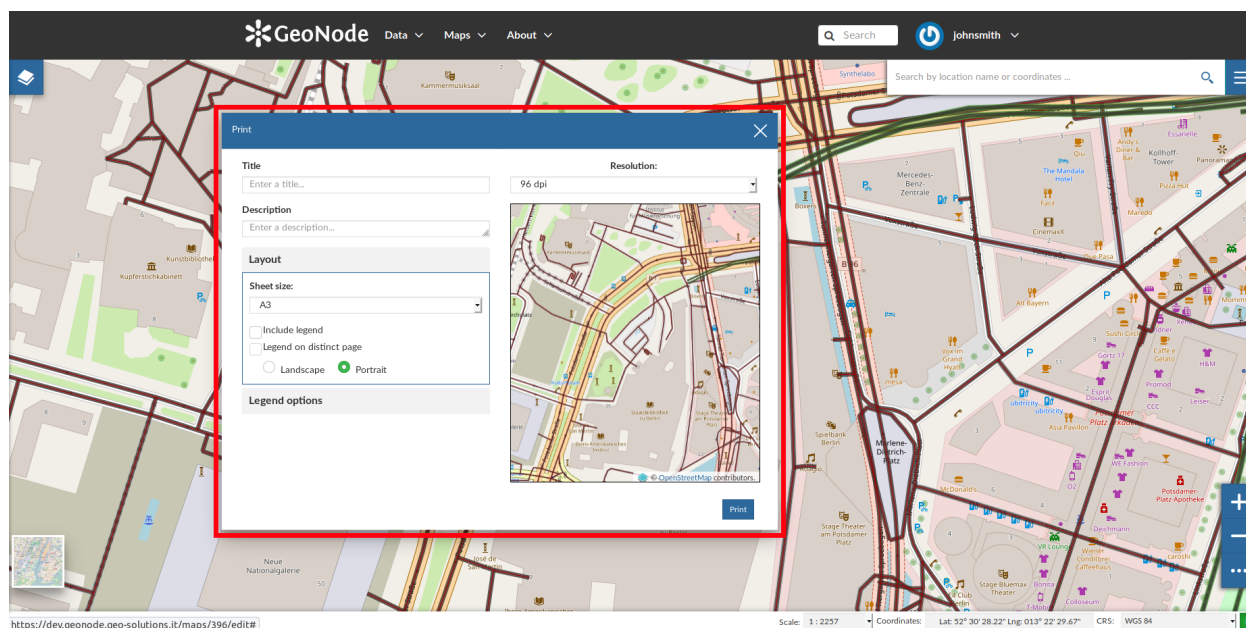


Fig. 204: A janela de Impressão

- se inclui ou não a legenda;
- se colocar a legenda numa página separada;
- a *Orientação* da página (Paisagem ou Retrato);
- personalizar a *Legenda*
 - o *Rótulo da Fonte*;
 - the *Tamanho da Fonte*;
 - a *Ênfase na Fonte* (negrito, itálico);
 - se *Forçar Rótulos*;
 - se usar *Fonte Anti-aliasing*;
 - o *Tamanho do Ícone*;
 - a *Resolução de Legenda* em ppp.

Para imprimir o mapa clique em *Print*.

O Catálogo de Camadas

Todas as camadas disponíveis no GeoNode, carregadas e remotas, podem ser adicionadas ao mapa através do *Catálogo*. Clique na opção *CATALOG* do *Menu de Opções do Mapa* para visualizar o painel do catálogo.

Pode navegar pelas camadas e ver suas imagens em *Miniatura*, *Título*, *Descrição* e *Resumo*. Clique em *Add To Map* para carregar uma camada no mapa, ela também estará visível em *Índice (TOC)*.

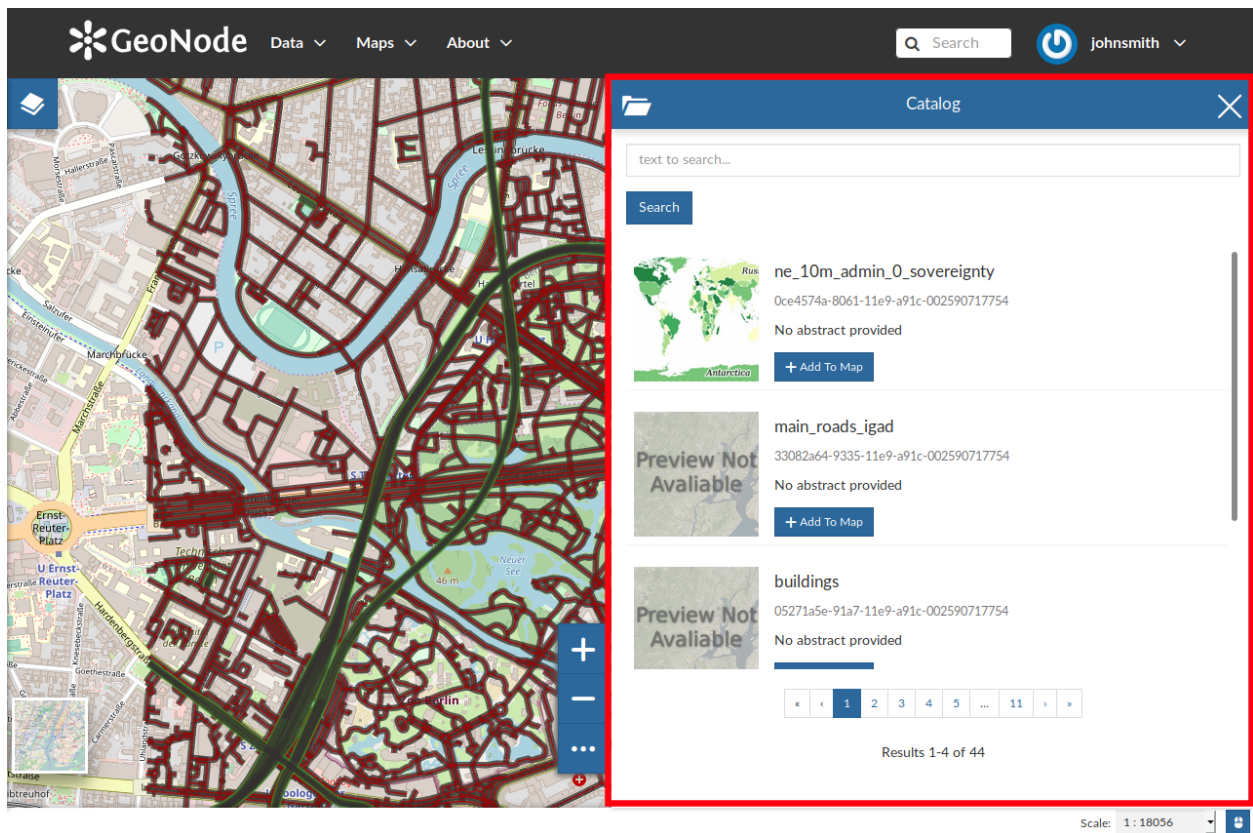


Fig. 205: O Catálogo de Camadas

Executando Medições

Clique na opção *MEASURE* do *Menu de Opções do Mapa* para realizar uma medição. Como pode ver na imagem em baixo, esta ferramenta permite medir *Distâncias*, *Áreas* e o *Rumo* das linhas.

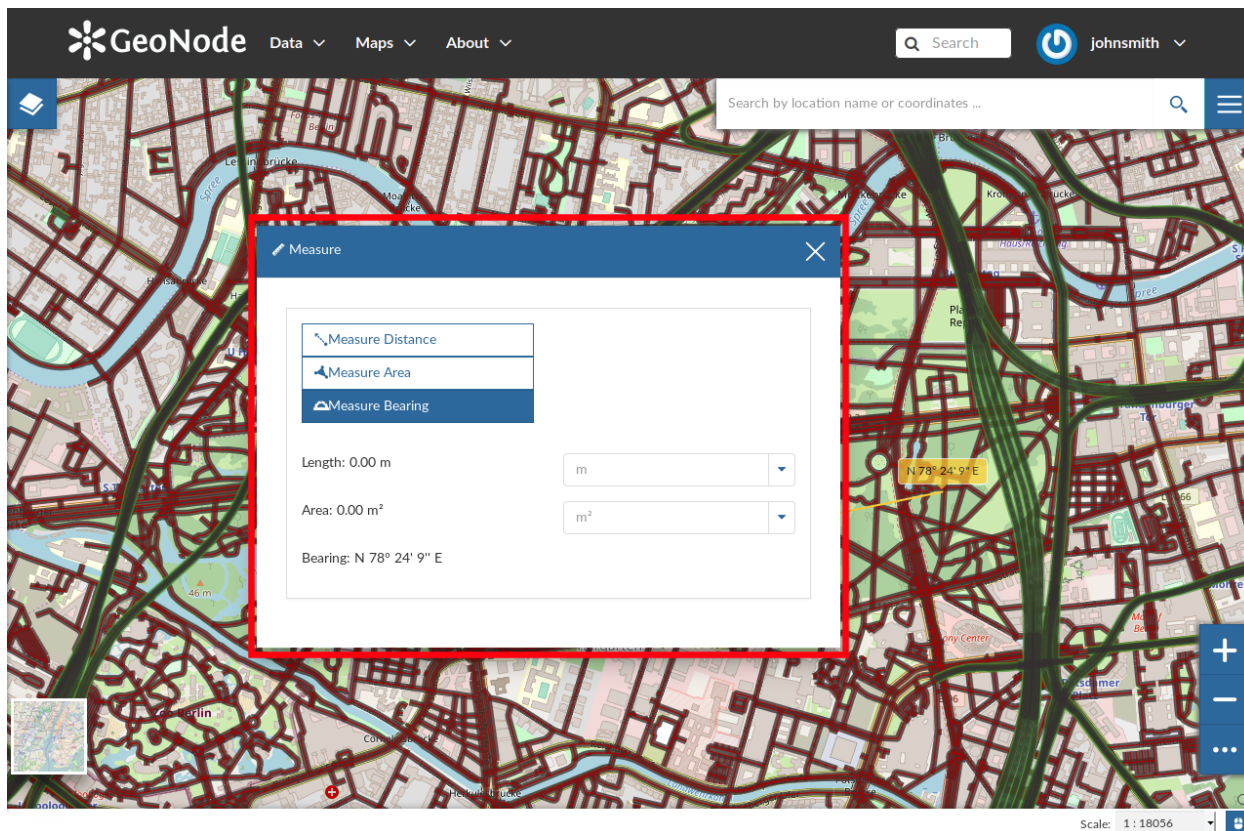


Fig. 206: A Ferramenta de Medição

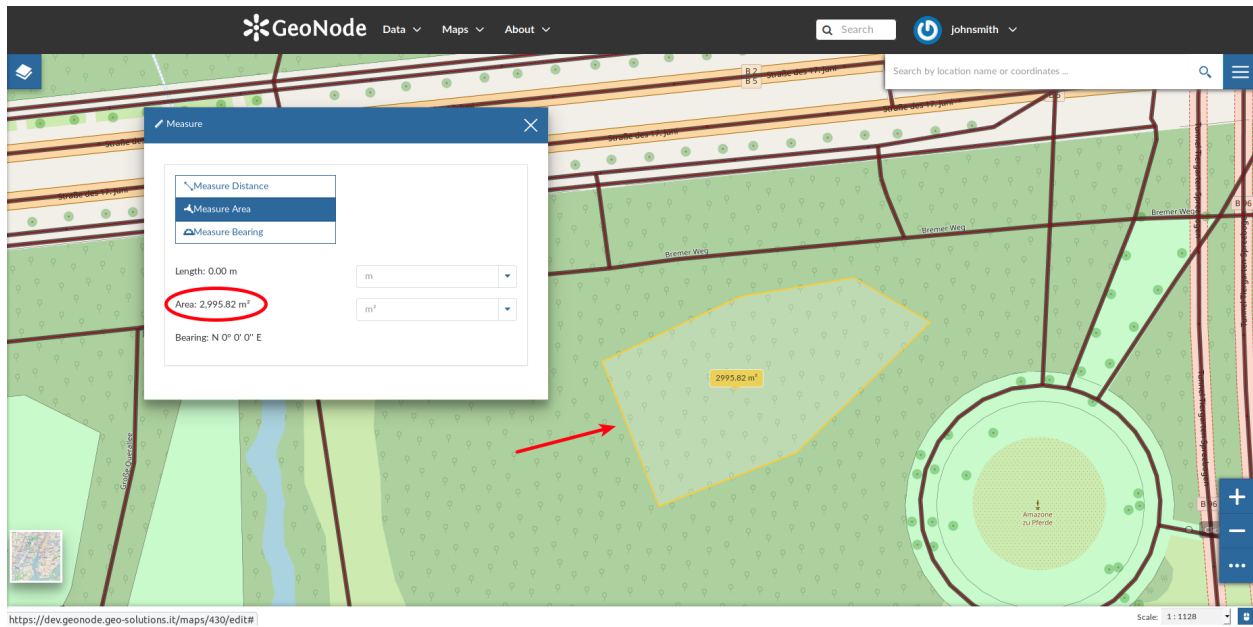
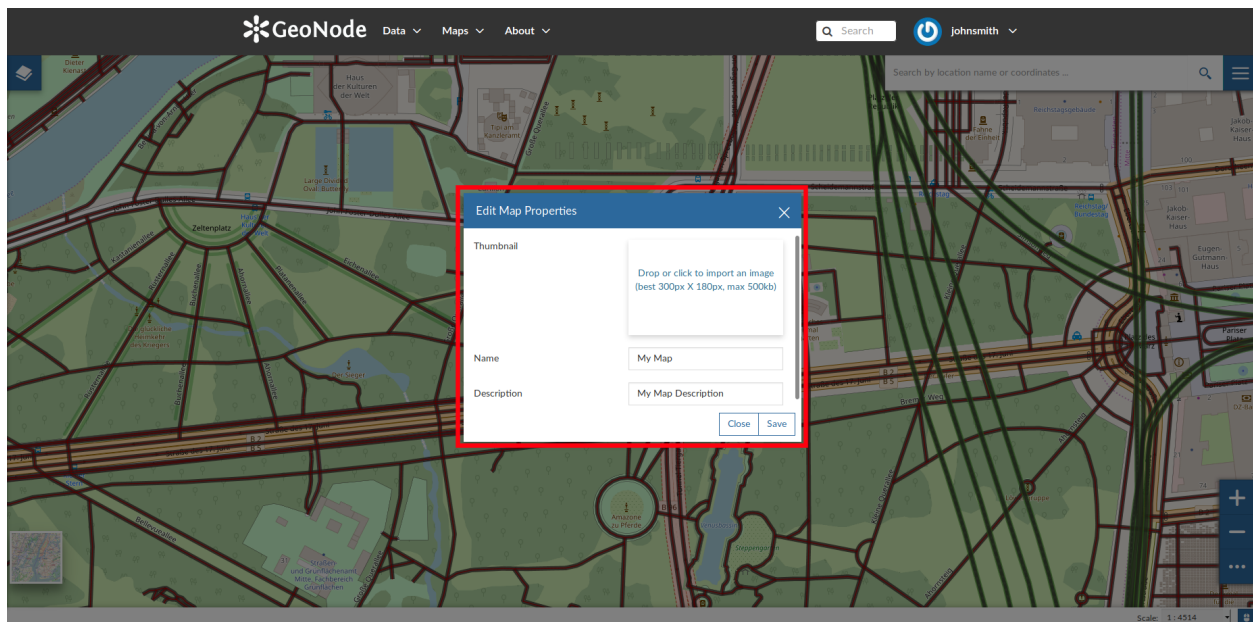
Para realizar uma medida, desenhe no mapa a geometria de seu interesse, o resultado será exibido à esquerda do menu de selecção da unidade de medida (esta ferramenta também permite alterar a unidade de medida).

Salvando um mapa

Uma vez que todas as personalizações tenham sido realizadas, pode *Salvar* seu mapa clicando na opção *SAVE AS* do *Menu de Opções do Mapa*.

Uma nova janela pop-up será aberta.

Deve preencher um *Título* e uma *Descrição* opcional, a seguir clicar em *Save*. A página será recarregada e o seu mapa deverá estar visível na lista *Mapas*.

Fig. 207: *Medição de Áreas*Fig. 208: *Salvando Mapas*

The screenshot shows the GeoNode 'Explore Maps' page. The header includes the GeoNode logo, navigation links (Data, Maps, About), a search bar, and a user profile (johnsmith). The main section is titled 'Explore Maps' and features a 'Create a New Map' button. On the left, there are filters for 'Selected Maps' (Add maps through the "checkboxes", Set permissions) and a list of filter categories: TEXT, KEYWORDS, CATEGORIES, OWNERS, GROUPS, GROUP CATEGORIES, DATE, REGIONS, and EXTENT. The main content area displays '13 Maps found' and a list of map entries. The first entry, 'My Map' by johnsmith, is highlighted with a red border. It includes a map thumbnail, the title 'My Map', a description 'My Map Description', the owner 'johnsmith', the date '21 Jun 2019', and statistics for views (0), shares (0), and stars (0). Below it are two other map entries: 'states' by Admin and 'taz2' by Admin, both dated 20 Jun 2019.

Fig. 209: *O seu Mapa na lista*

Personalizando os Modelos GetFeatureInfo das Camadas

Ao “clique” sobre um recurso de uma camada em um mapa GeoNode, uma janela pop-up de informações mostra uma representação formatada dos dados brutos identificados pelas coordenadas (ver Fig. 1)

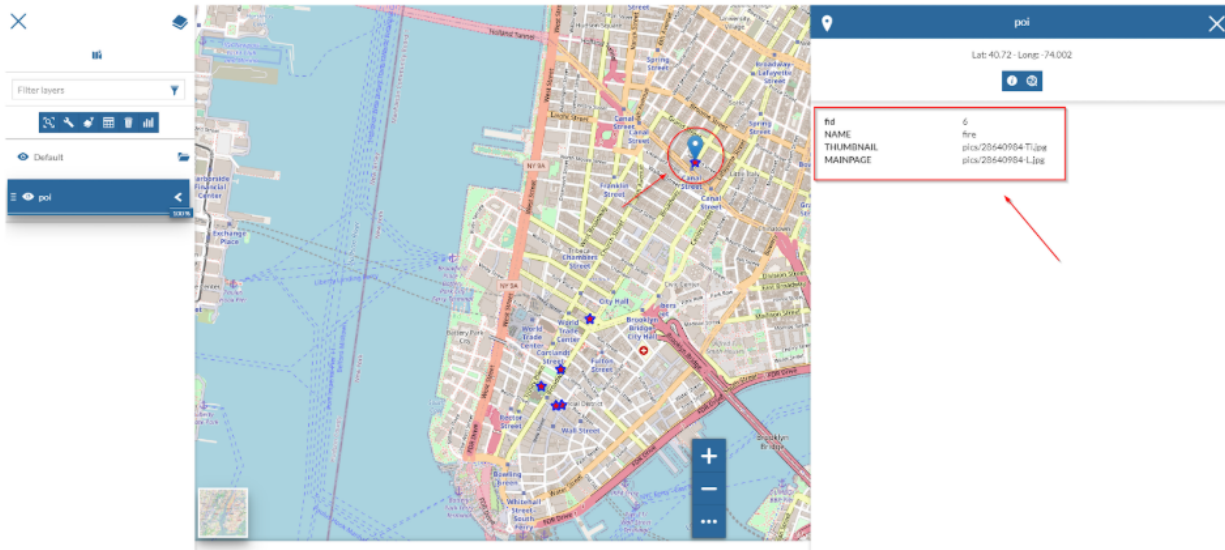


Fig. 210: Fig. 1

A forma como a informação é apresentada ao utilizador é definida pelo que chamamos de “Modelo de GetFeatureInfo”. O último é basicamente um pedaço de HTML contendo alguns marcadores de posição e códigos embutidos especiais que instruem o GeoServer sobre como gerar a saída de dados em bruto.

O resultado é um fragmento de HTML renderizado com os valores reais substituindo os espaços reservados do Modelo.

Actualmente, o GeoNode permite um mecanismo muito simples para personalizar o “Modelo de GetFeatureInfo” de uma camada.

É possível, por meio do Assistente Editor de Metadados de Camada, atribuir um nome e um rótulo aos atributos que desejamos exibir na saída GetFeatureInfo.

Observe que os atributos sem um rótulo e nome, no caso de outros estarem presentes, não serão renderizados.

Como uma instância, usando o exemplo acima, podemos personalizar um pouco os Metadados da Camada, conforme mostrado na Fig. 2

A saída “GetFeatureInfo” mudará de acordo, conforme mostrado na Fig. 3

Modelo Simples: Atribuindo um Media-Type (tipo de media) a Valores de Atributo

A maneira mais fácil de renderizar um Media-Type diferente (:guilabel: *imagem*, *audio*, *video* ou *iframe*) para um valor de propriedade, é alterá-lo no painel de atributos *Metadata Editor Wizard*.

Alterando o *Display Type* de um atributo deste painel, conforme mostrado na Fig. 4

O GeoNode criará automaticamente o tipo de media HTML ao renderizar usando o **valor** da propriedade seleccionada.

Assim, a título de exemplo, se, na figura acima, o atributo *NOME* contém valores que representam alguns *links* para outros recursos, o GeoNode irá criar esses links automaticamente para si ao clicar sobre uma geometria.

Metadata for poi

Completteness

✖Check Schema mandatory fields

55 %

EditPreviewSettings

MandatoryMandatoryOptional

1234

Basic MetadataLocation and LicensesOptional MetadataDataset Attributes

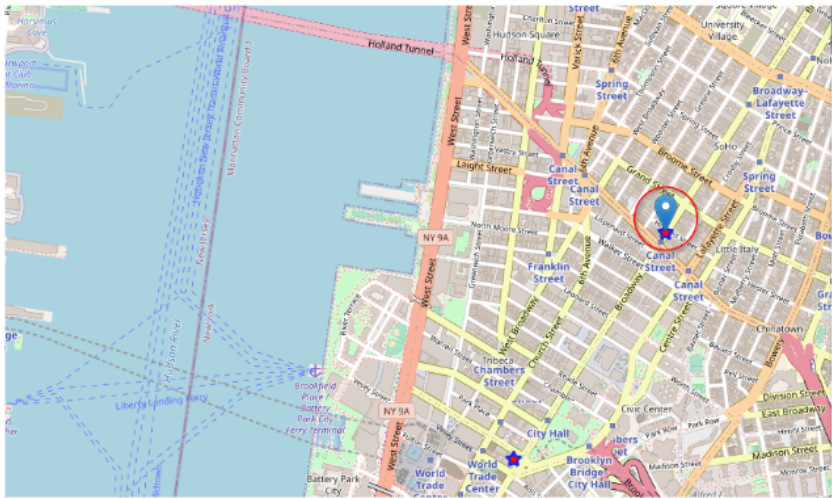
Attribute	Label	Description	Display Order
NAME	Point Of Interest	Point Of Interest	1
THUMBNAIL	Thumbnail		2
fid			3
the_geom			4
MAINPAGE			5

Return to Layer

<< Back

Update

Fig. 211: Fig. 2



poi

Lat: 40.72 - Long: -74.002

Point Of Interest

fire

Thumbnail

pics/28640984-TI.jpg

Fig. 212: Fig. 3

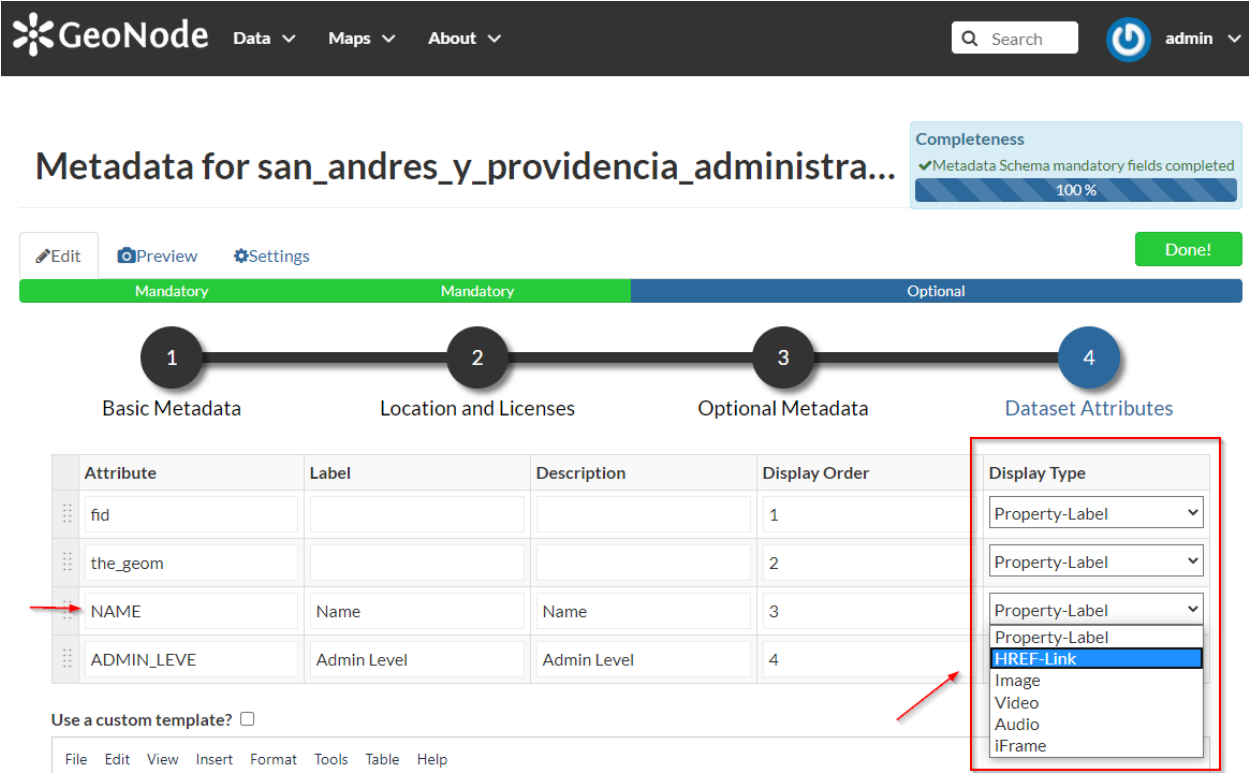


Fig. 213: Fig. 4

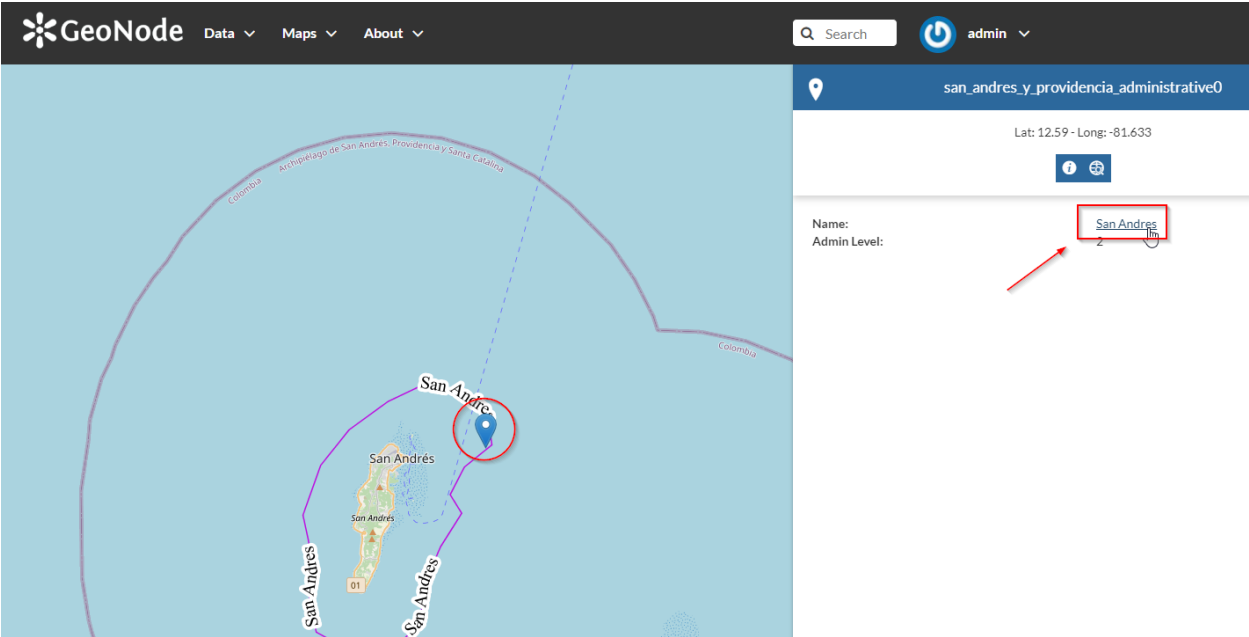


Fig. 214: Fig. 5

Seleccionando *image* como Media-Type (Fig. 6)

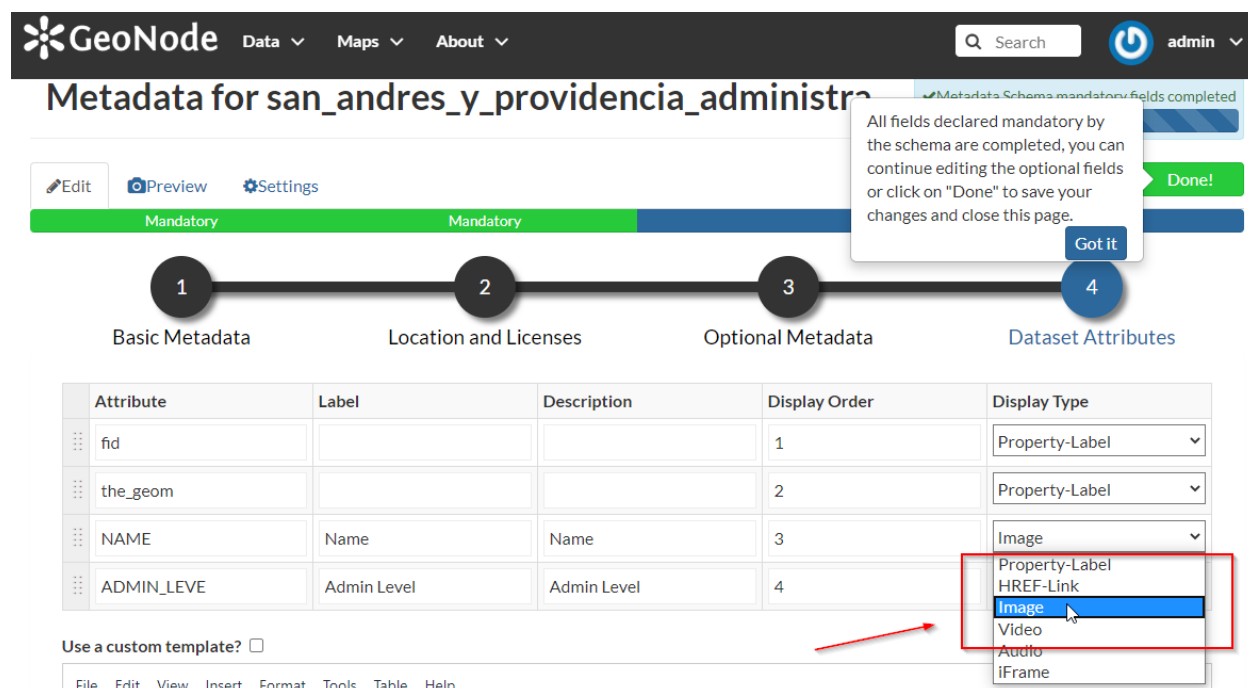


Fig. 215: Fig. 6

e editar o conteúdo de acordo (Fig. 7)

obterá um bom efeito, conforme mostrado na Fig. 8

Modelo Avançado: Use um Modelo HTML Personalizado

Seleccionando a opção *Use a custom template?* Como mostrado na Fig. 9

Poderá fornecer seu próprio modelo HTML personalizado para a saída de informações do recurso.

O exemplo abaixo mostra como é possível criar uma boa saída HTML com uma *image* tomando o ``src`` dos valores do atributo *NAME*, através do uso da palavra-chave `{properties.NAME}`

Opcional: Personalização da Barra de Menu do Editor HTML WYSIWYG

O *Barra de Menu* e *Barra de Ferramentas* do Editor de HTML, pode ser facilmente personalizado substituindo a variável `TINYMCE_DEFAULT_CONFIG` em `settings.py` (veja `TINYMCE_DEFAULT_CONFIG`)

Existem muitos plugins e opções que permitem personalizar facilmente o editor e também fornece alguns *modelos* predefinidos para acelerar a edição.

Para obter mais informações sobre a ferramenta Javascript, consulte <https://www.tiny.cloud/>

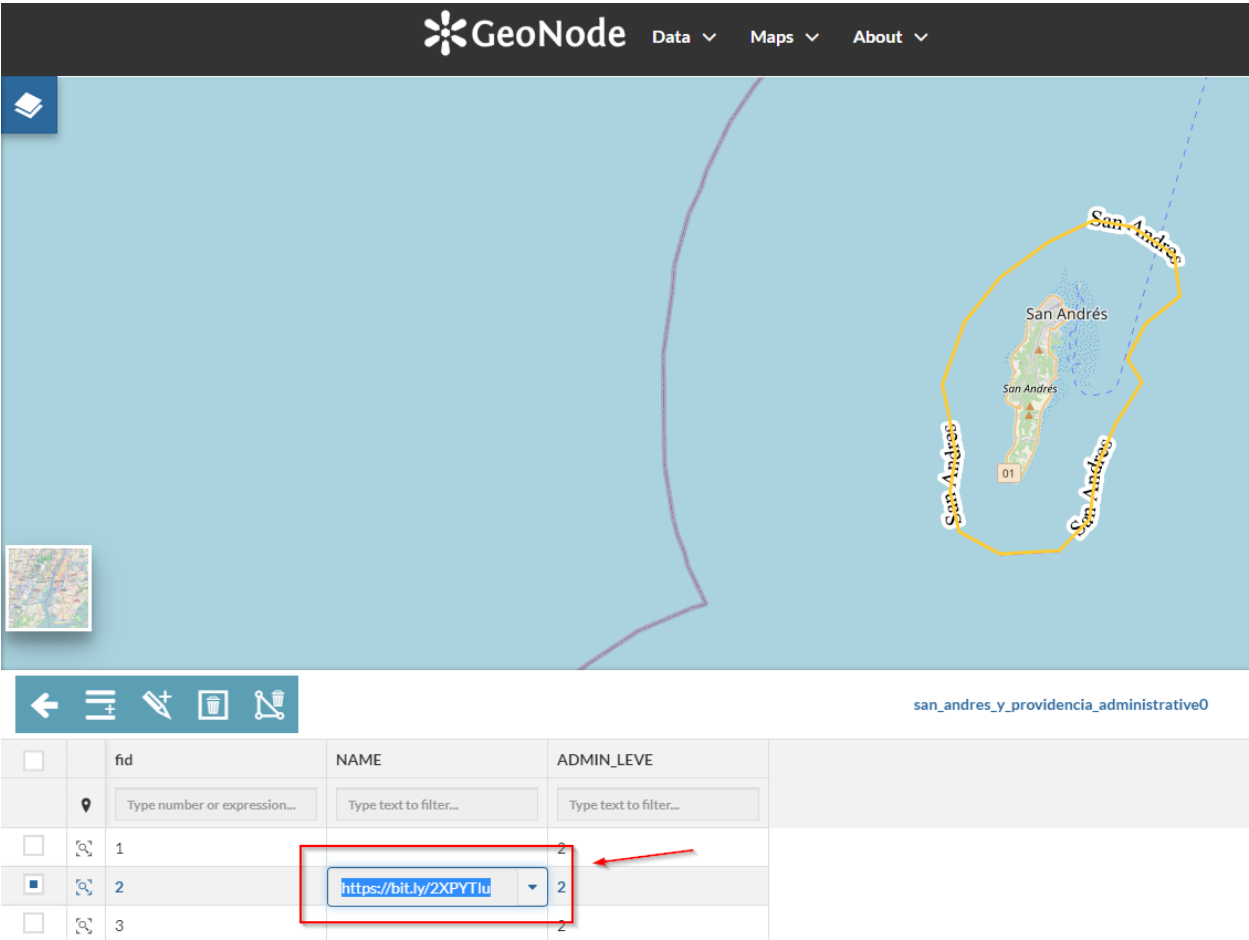


Fig. 216: Fig. 7

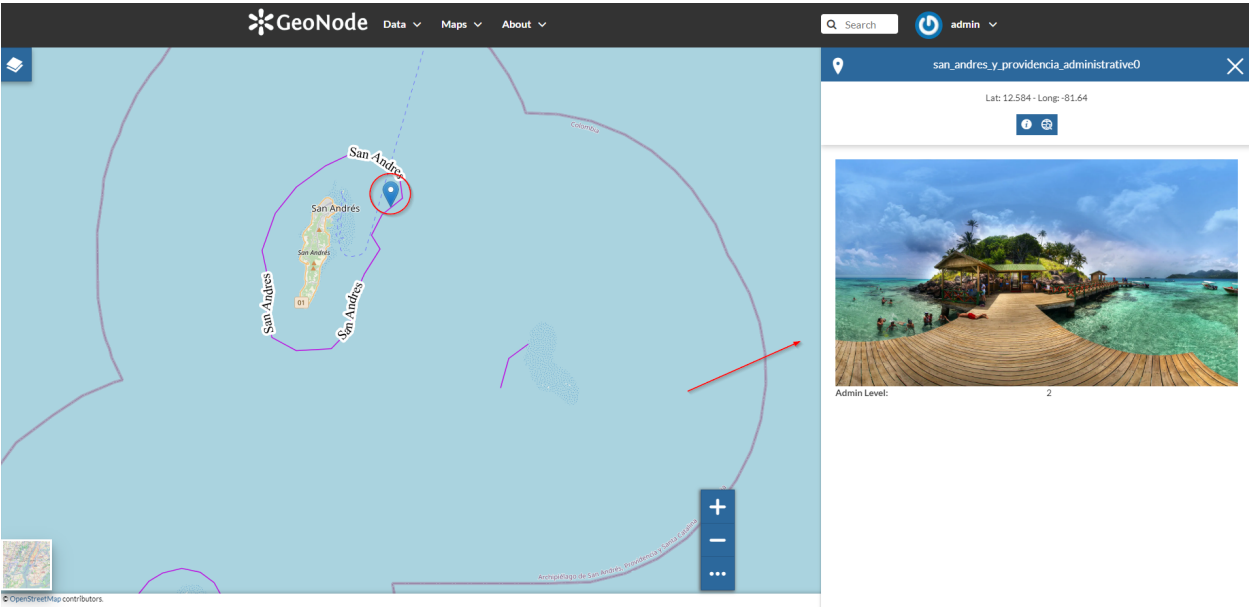


Fig. 217: Fig. 8

GeoNode Data Maps About Search

Metadata for san_andres_y_providencia_administra...

Completeness
✓ Metadata Schema

Edit Preview Settings

Mandatory Mandatory Optional

1 Basic Metadata 2 Location and Licenses 3 Optional Metadata Data

Attribute	Label	Description	Display Order	Display
fid			1	Property
the_geom			2	Property
NAME	Name	Name	3	Image
ADMIN_LEVE	Admin Level	Admin Level	4	Property

Use a custom template? ☒

File Edit View Insert Format Tools Table Help

Fig. 218: Fig. 9

Use a custom template? ☒

File Edit View Insert Format Tools Table Help

↩ ↪ B I U S <> Source code 22px

Word count

San Andrés and Providencia

A red arrow points to the 'Word count' option in the 'Tools' menu.

Fig. 219: Fig. 10

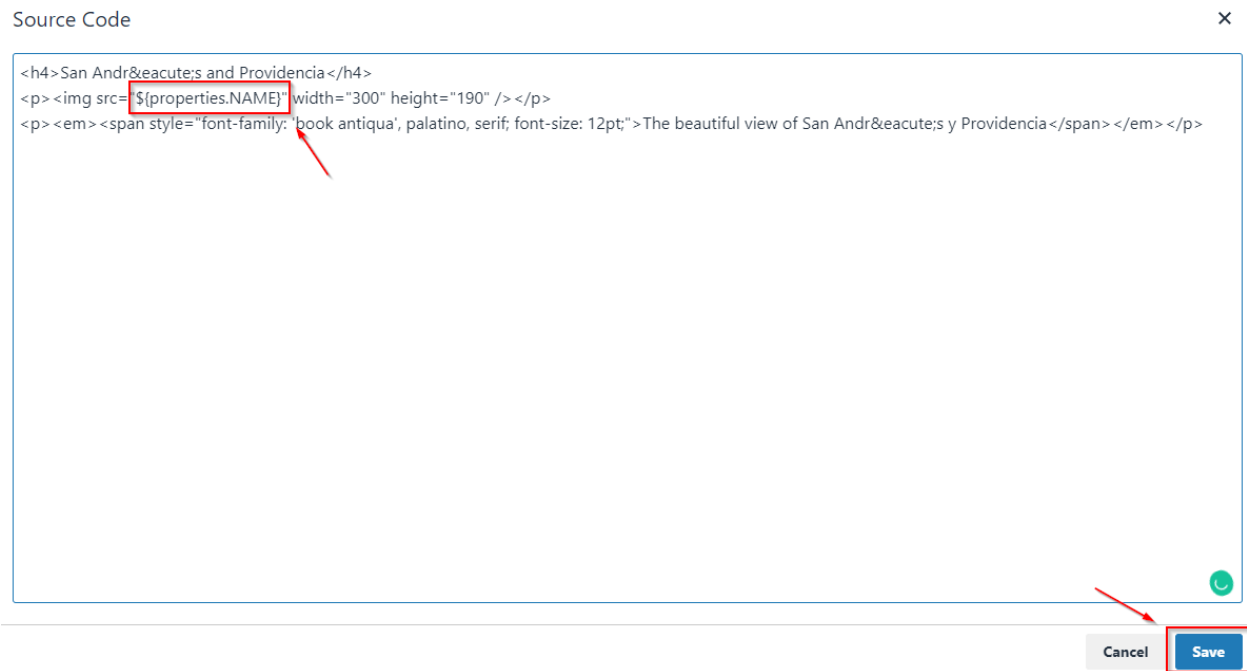


Fig. 220: Fig. 11

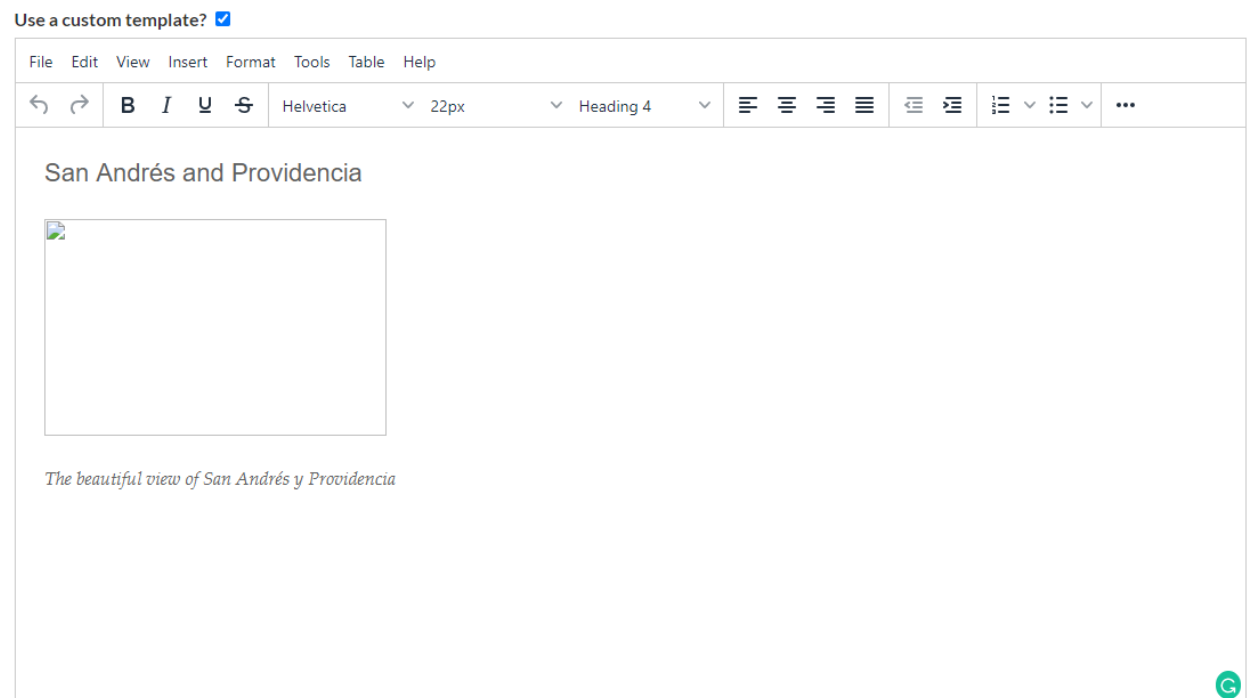


Fig. 221: Fig. 12

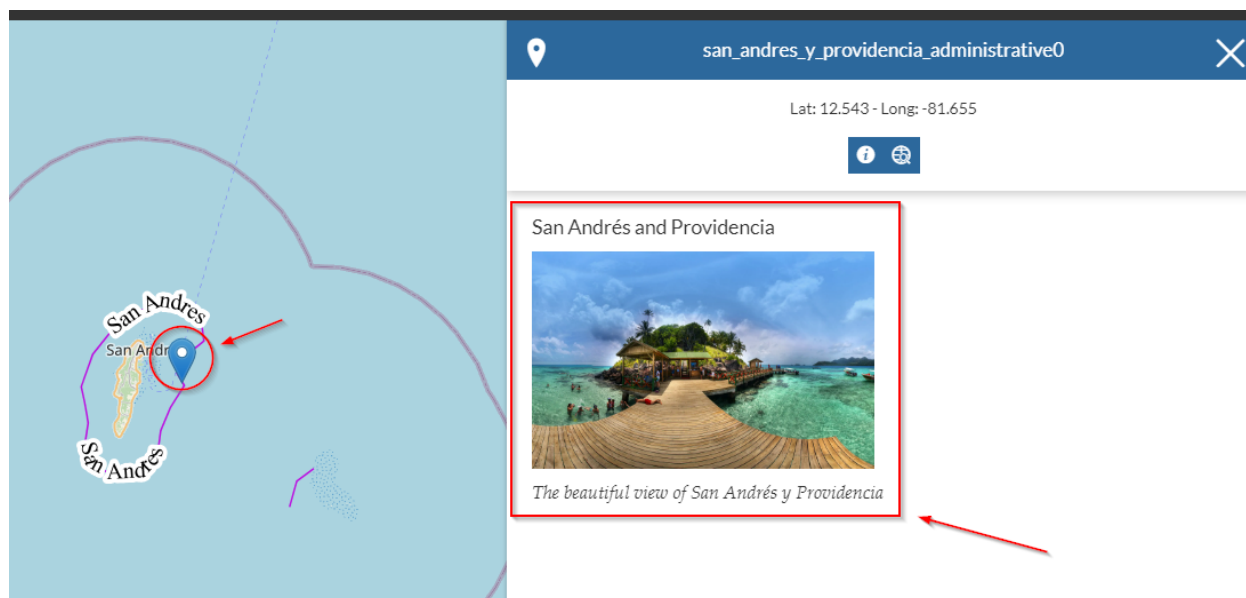


Fig. 222: Fig. 13

Barra de Pesquisa

A *Barra de Pesquisa* do visualizador do mapa permite que localizar pontos de interesse (POIs), ruas ou locais por nome.

Vamos digitar o nome de um lugar e seleccionar o primeiro registo.

O mapa será automaticamente centralizado nessa área, delimitando-a por um polígono no caso de uma área, por uma linha no caso de uma forma linear (por exemplo, ruas, riachos) e por um marcador no caso de um ponto.

Ferramentas da Barra Lateral

O *Visualizador de Mapas* disponibiliza também a *Barra Lateral*. É um painel de navegação que contém várias ferramentas que o ajudam a explorar o mapa, como ferramentas de zoom, alteração de extensão e consulta de objectos no mapa.

Por padrão, a *Barra Lateral* mostra os botões de zoom  e , outras opções podem ser exploradas clicando em  que expande/colapsa a barra de ferramentas.

A *Barra lateral* contém as seguintes ferramentas:

- A *Consulta de Objectos no Mapa* permite obter informações de características através do botão . Ele permite recuperar informações sobre os recursos de algumas camadas clicando nelas directamente no mapa.

Ao clicar no mapa, abre um novo painel. Esse painel mostrará todas as informações sobre os recursos clicados para cada camada activa carregada.

- Pode fazer *Zoom na Extensão Máxima* clicando em .

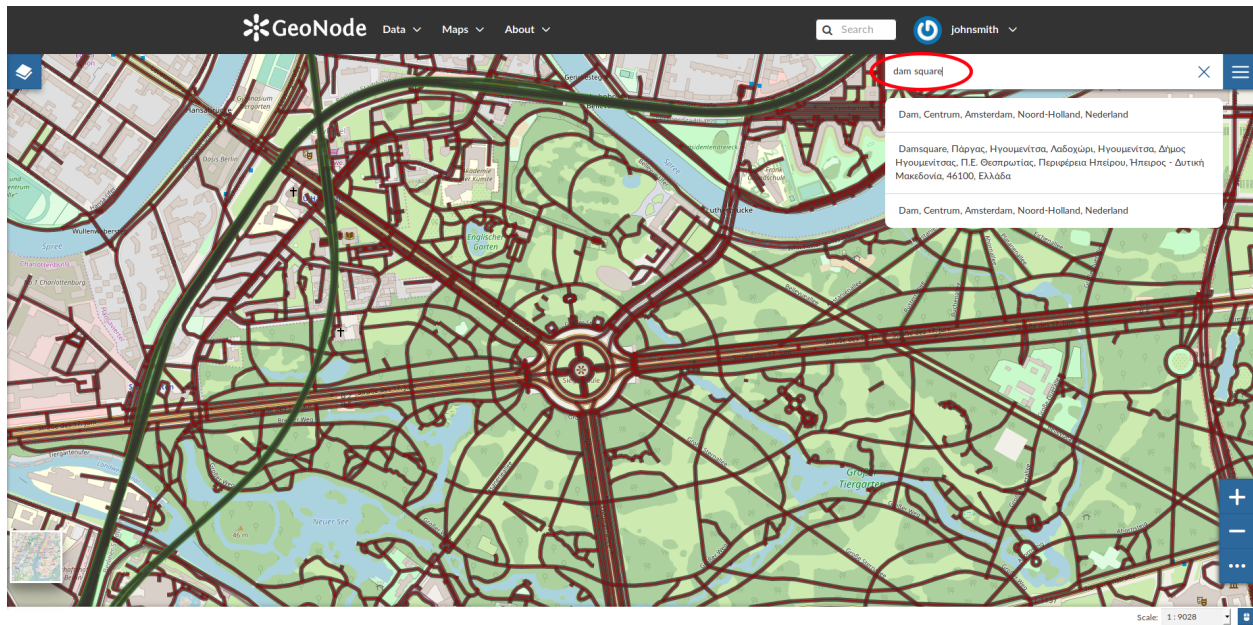


Fig. 223: A barra de Pesquisa

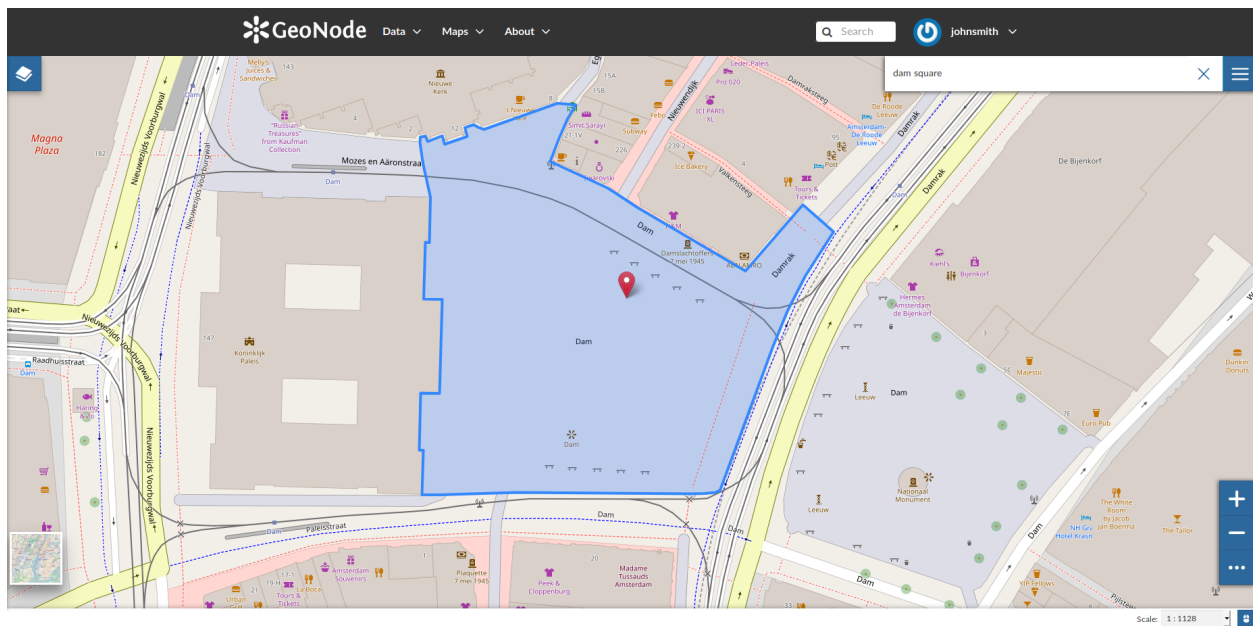


Fig. 224: Resultado de uma Pesquisa

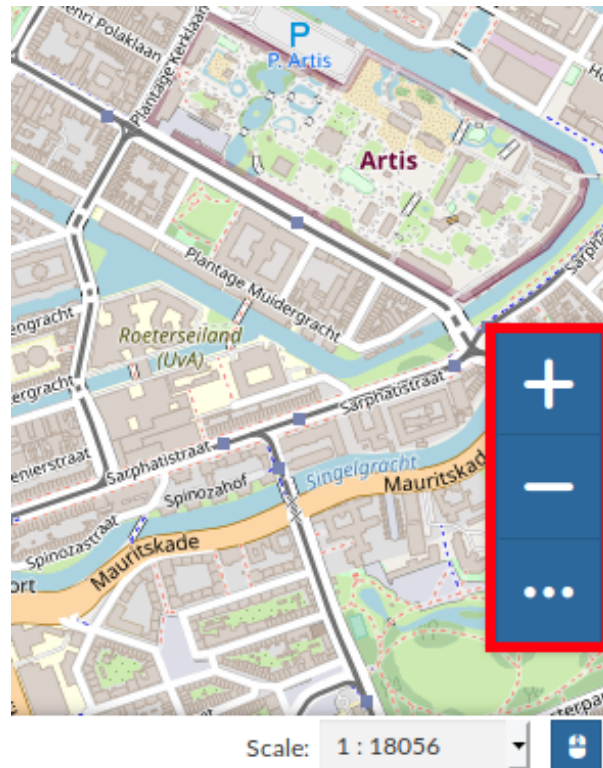


Fig. 225: A Barra lateral Padrão

- Pode alternar entre o nível de zoom anterior e o próximo por meio do botão *Voltar*  e o *Avançar* .
- O botão *Mudar para Tela Inteira*  permite ter um mapa em tela cheia.

Comutador de Mapa base

Por padrão, o GeoNode permite enriquecer mapas com muitos fundos do tipo mundo:

- *OpenStreetMap*
- *OpenTopoMap*
- *Sentinel-2-cloudless*

Pode, também, decidir ter um *Fundo vazio*.

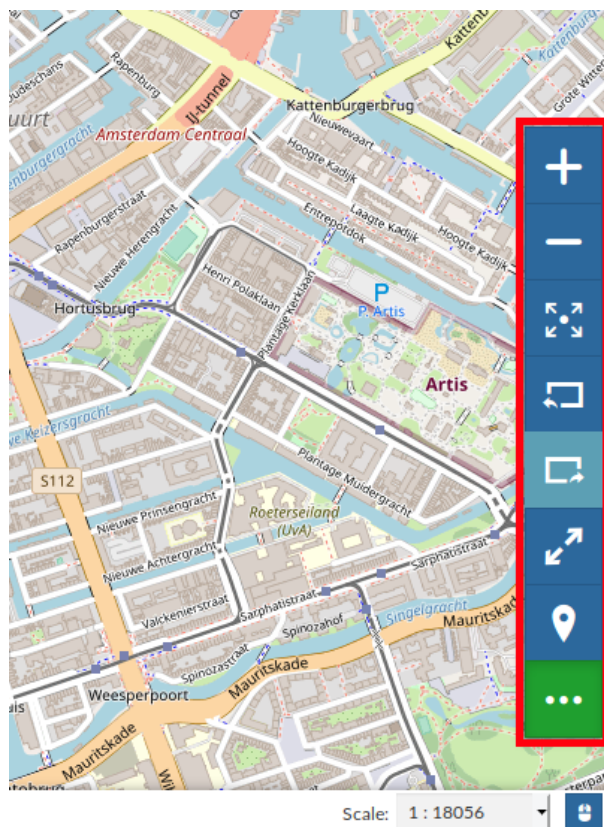


Fig. 226: A Barra lateral Expandida

Fig. 227: Consultando Objectos no mapa

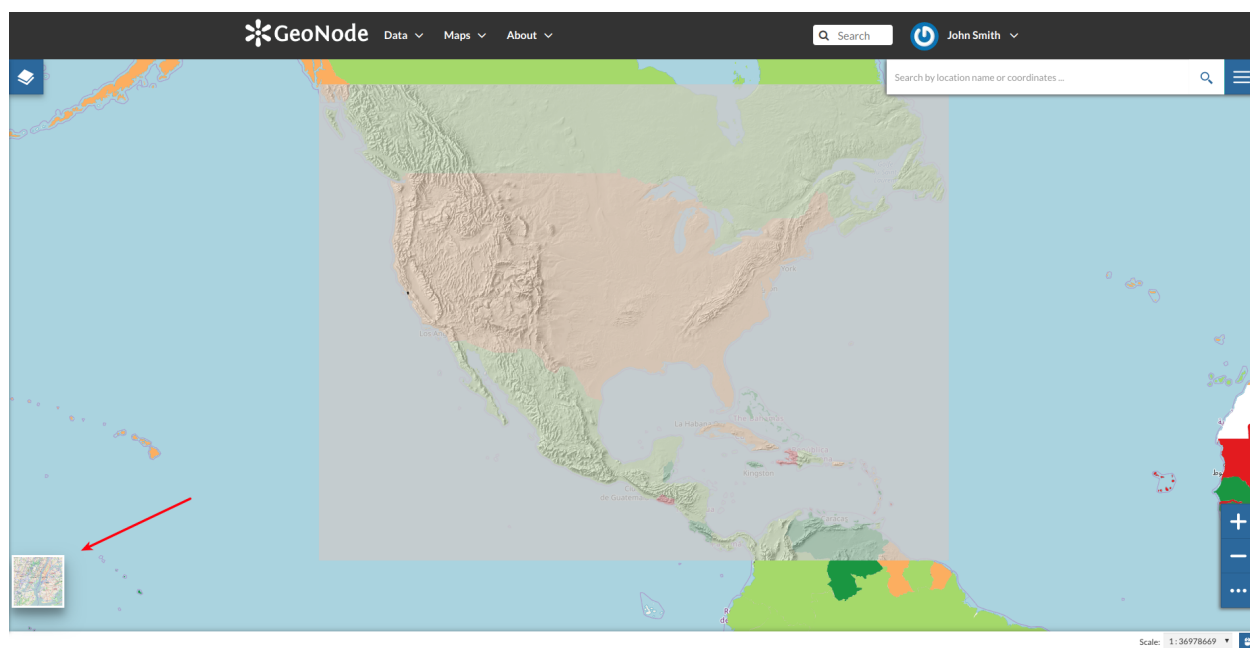


Fig. 228: A Ferramenta Basemap Switcher

Fig. 229: Mudando o Mapa Base

Ferramentas de Rodapé

Na parte inferior do mapa, o *Rodapé* mostra a *Escala* do mapa e permite que esta seja alterada.

O botão  permite que veja o ponteiro *Coordenadas* e altere o Sistema de Referência de Coordenadas (CRS), WGS 84 por padrão.

1.10.7 Publicação de Dados

No GeoNode, cada recurso pode ser publicado de forma a ser partilhado com outras pessoas. Assim que um *Mapa* for publicado, poderá incorporá-lo nas suas páginas da web, blog ou site da web.

Uma maneira fácil de conseguir isso é usar um `iframe`. Veja as seguintes etapas:

- Abra a página *Informação do Mapa* e copie o *URL*
- Adicione «/incorpore» ao URL para que fique assim «<http://master.demo.geonode.org/maps/11/embed>»
- Use este URL dentro de um `iframe` html como valor `src`

```
<iframe
  style="border: none;" height="400" width="600"
  src="http://master.demo.geonode.org/maps/11/embed">
</iframe>
```

- Coloque este bloco de código html dentro de suas páginas da web para exibir o mapa.

Ao salvar um arquivo html com este código pode testar o mapa no seu pc, veja a figura seguinte.

Como pode ver, algumas funcionalidades básicas estarão disponíveis para o utilizador: o *Índice (TOC)*, o *Comutador de Mapa base*, o *Ferramentas da Barra Lateral* e o *Ferramentas do Menu de Opções*.

1.10.8 Usando o GeoNode com Outras Aplicações

O seu projecto GeoNode é baseado em componentes centrais que são interoperáveis e, como tal, é simples integrar com aplicativos e serviços externos. Esta secção irá orientá-lo sobre como conectar à sua instância GeoNode partindo de outros aplicativos e como integrar outros serviços no seu projecto GeoNode. No final, deve ter uma boa ideia sobre as possibilidades de integração e conhecimentos básicos sobre como alcançá-la. Pode achar necessário obter mais conhecimento em como fazer uma integração mais complexa para atingir os seus objectivos, mas deve sentir-se confortável com o básico e sentir-se confiante em pedir ajuda à ampla comunidade do GeoNode.

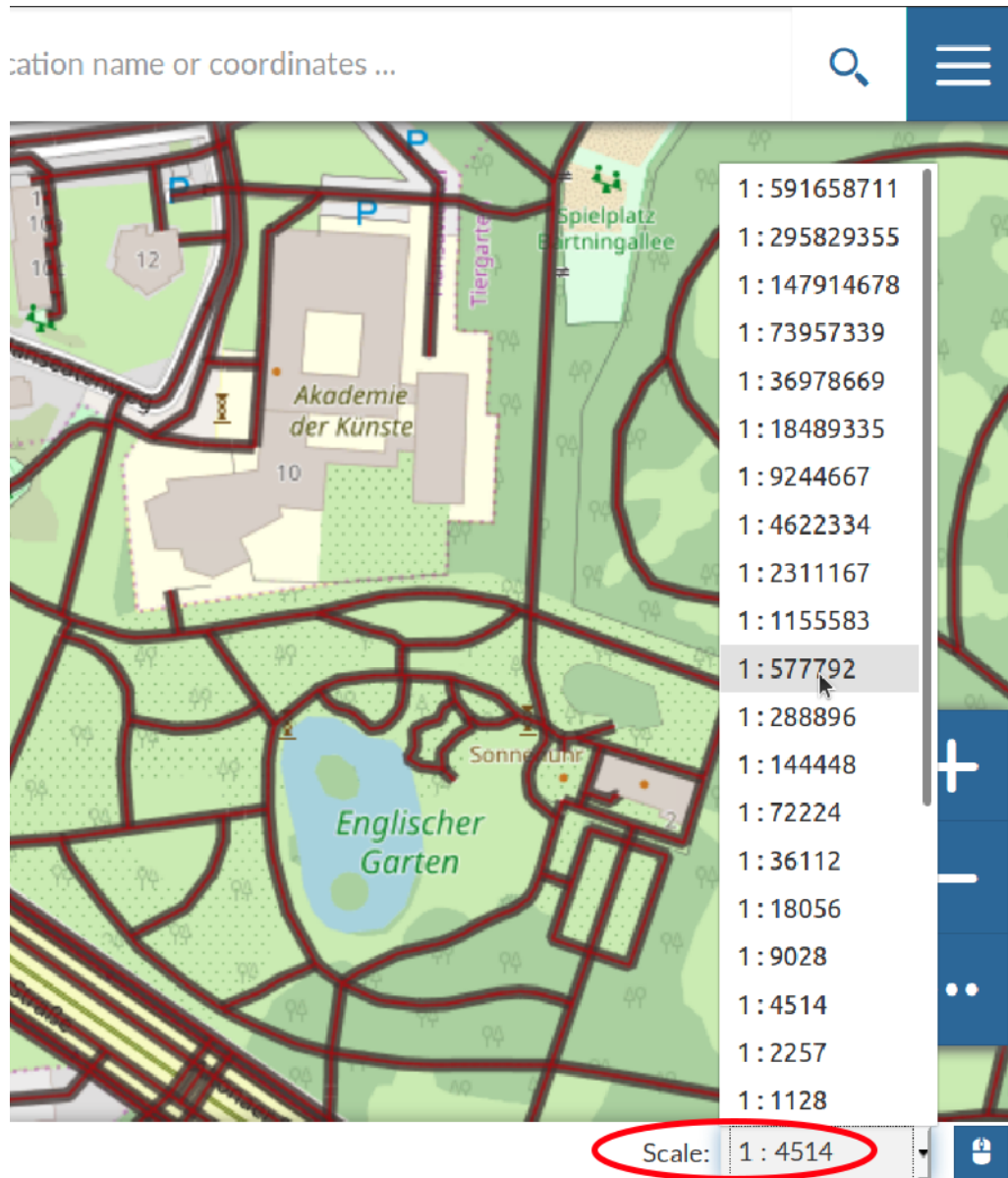


Fig. 230: A Escala do Mapa

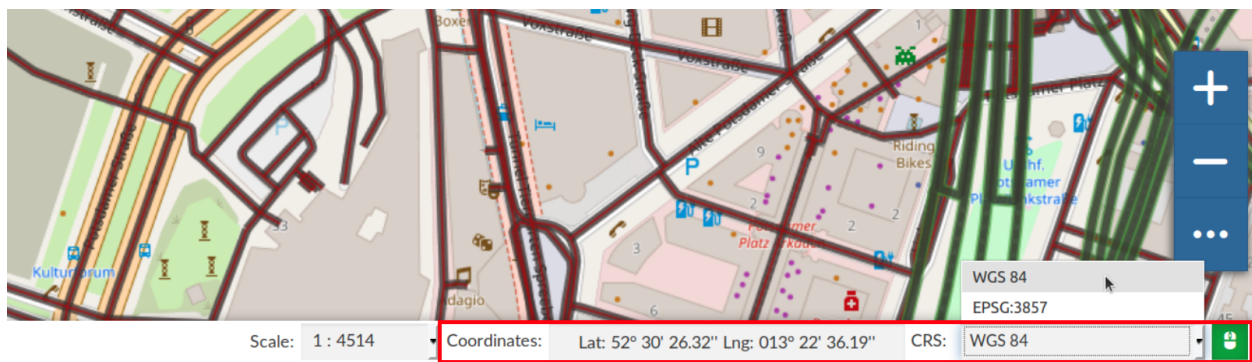


Fig. 231: *As Coordenadas do Ponteiro e o CRS *

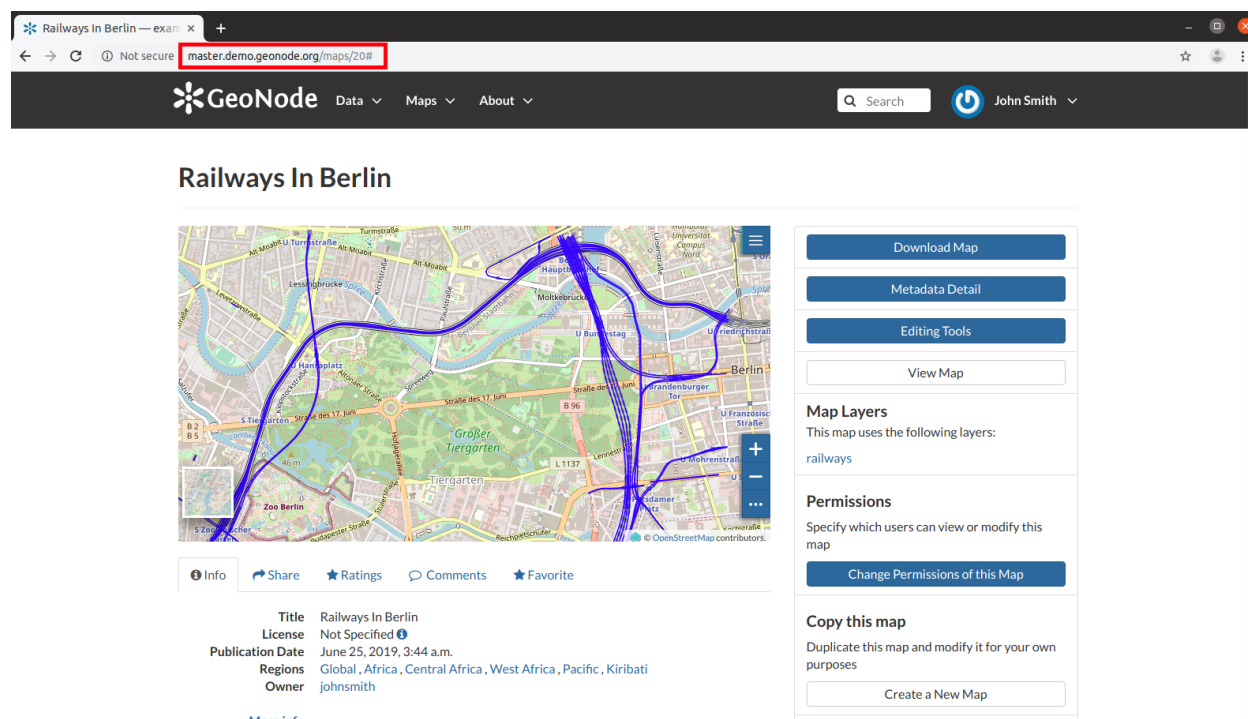


Fig. 232: O URL da página de Informações do Mapa

QGIS Desktop

QGIS é uma aplicação profissional de SIG que é construída sob e orgulhosa de ser ela mesma um Software Livre e de Código Aberto (FOSS). O QGIS é um projecto orientado por voluntários se estiver interessado, pode encontrar mais informações em <https://www.qgis.org>.

Como posso conectar-me ao Geonode?

Abra o QGIS Desktop e vá ao **Layer Menu > Data Source Manager**. Na parte inferior do Data Source Manager, poderá ver um separador com o nome e um ícone relacionado com Geonode. Isto porque o Geonode é reconhecido como uma fonte de dados dentro do QGIS.

Nota: É possível que também carregue as instâncias Geonode de um ficheiro existente, isto que é útil para partilhar entre utilizadores ou para criar cópias de segurança de conexões existentes.

Para adicionar uma nova instância Geonode, no separador Geonode seleccionado clique em **New** e verá a seguinte caixa de diálogo:

Na caixa de diálogo Preencha o nome como quiser e na URL coloque o link da instância do Geonode. É possível editar algumas opções WFS e WMS para otimizar a conexão. Se estiver tudo bem, receberá a seguinte caixa de diálogo de conexão bem sucedida:

Após a caixa de diálogo de bem-sucedido, agora é possível carregar todas as camadas da instância Geonode clicando no botão **Conectar**. Você pode ver as conexões WMS e WFS do Geonode e pode carregar no QGIS Desktop.

Após seleccionar uma camada (WMS ou WFS), clique no botão **Adicionar** e a camada será exibida na janela principal do QGIS.

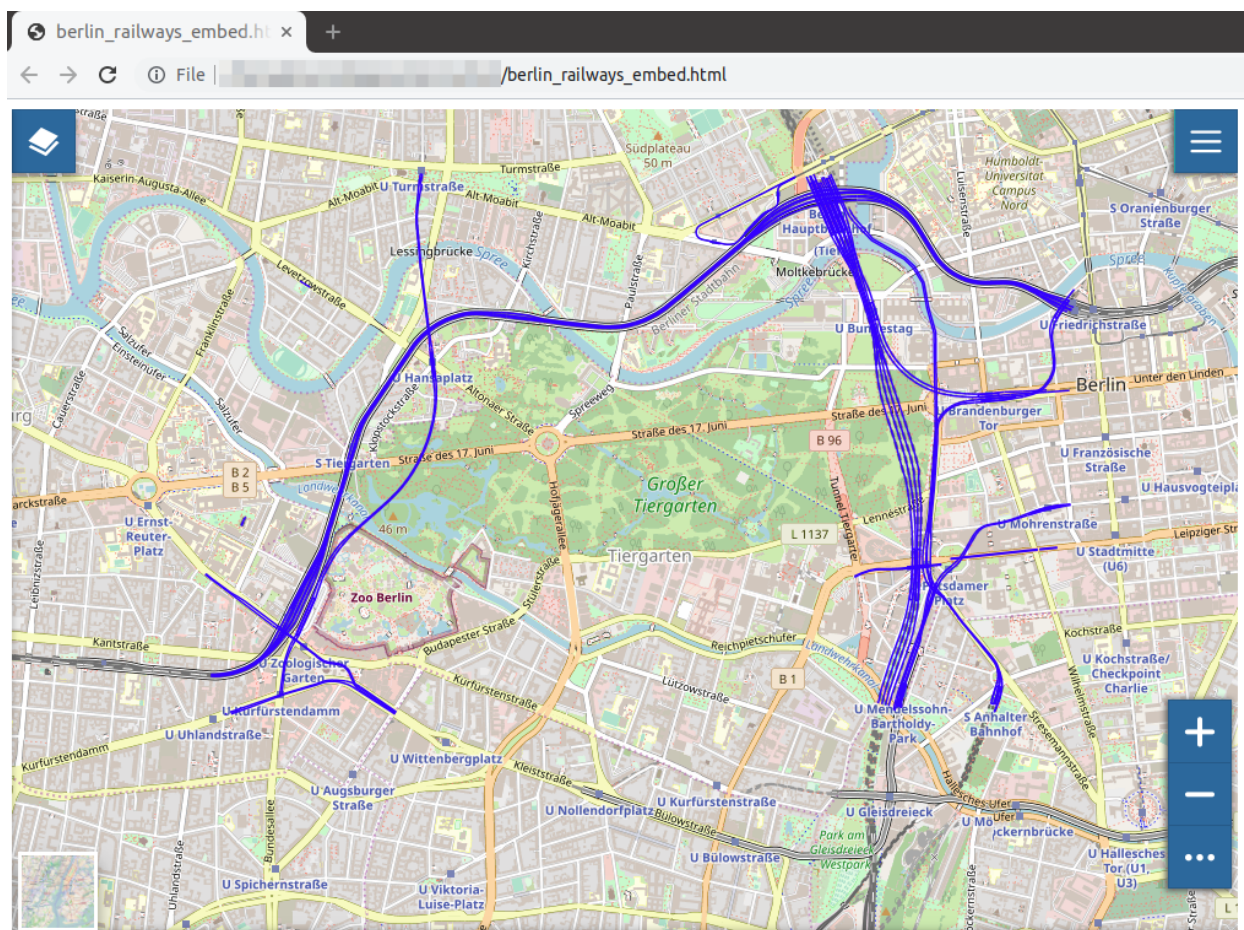


Fig. 233: O Mapa Incorporado

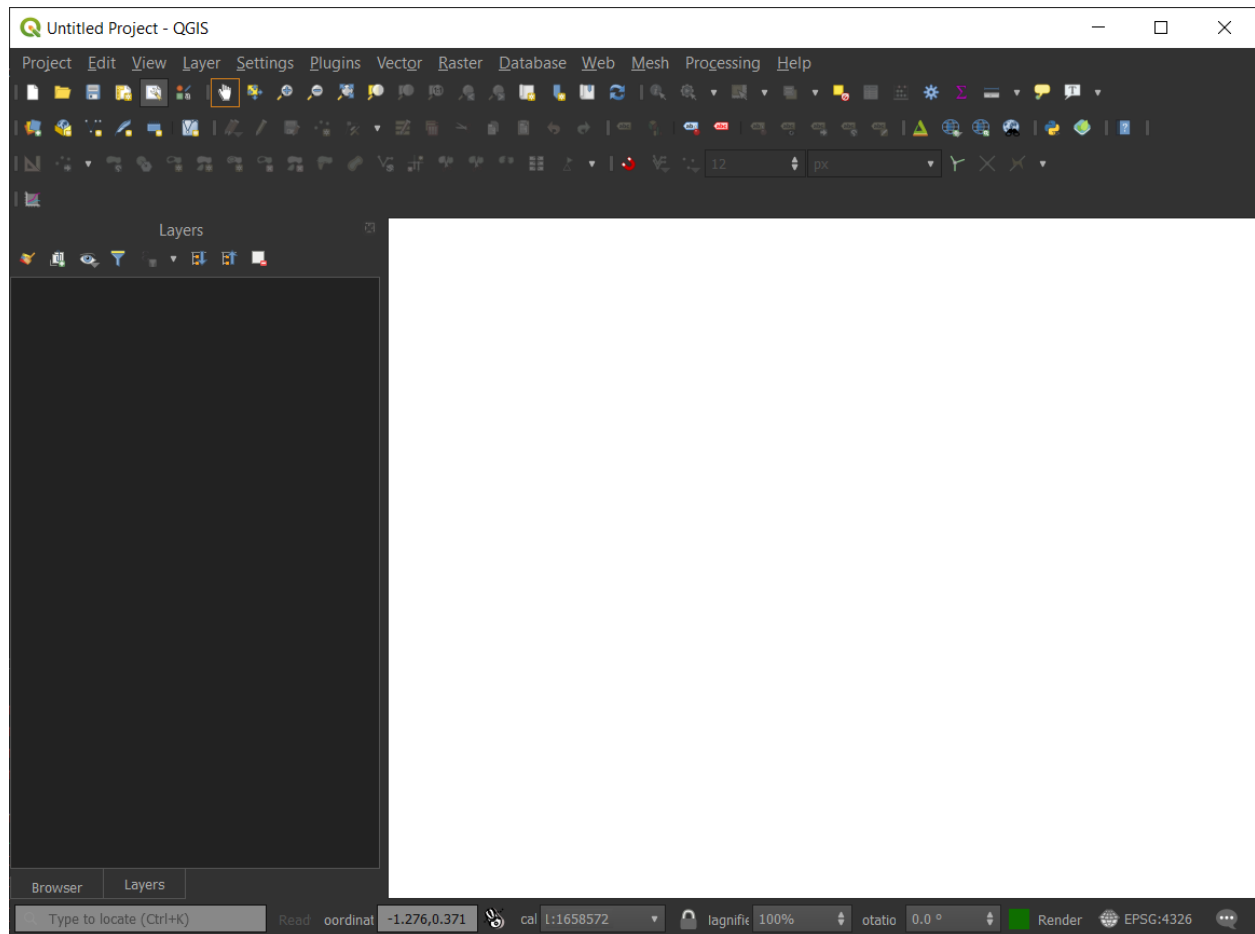


Fig. 234: *Janela Principal do Ambiente de trabalho QGIS*

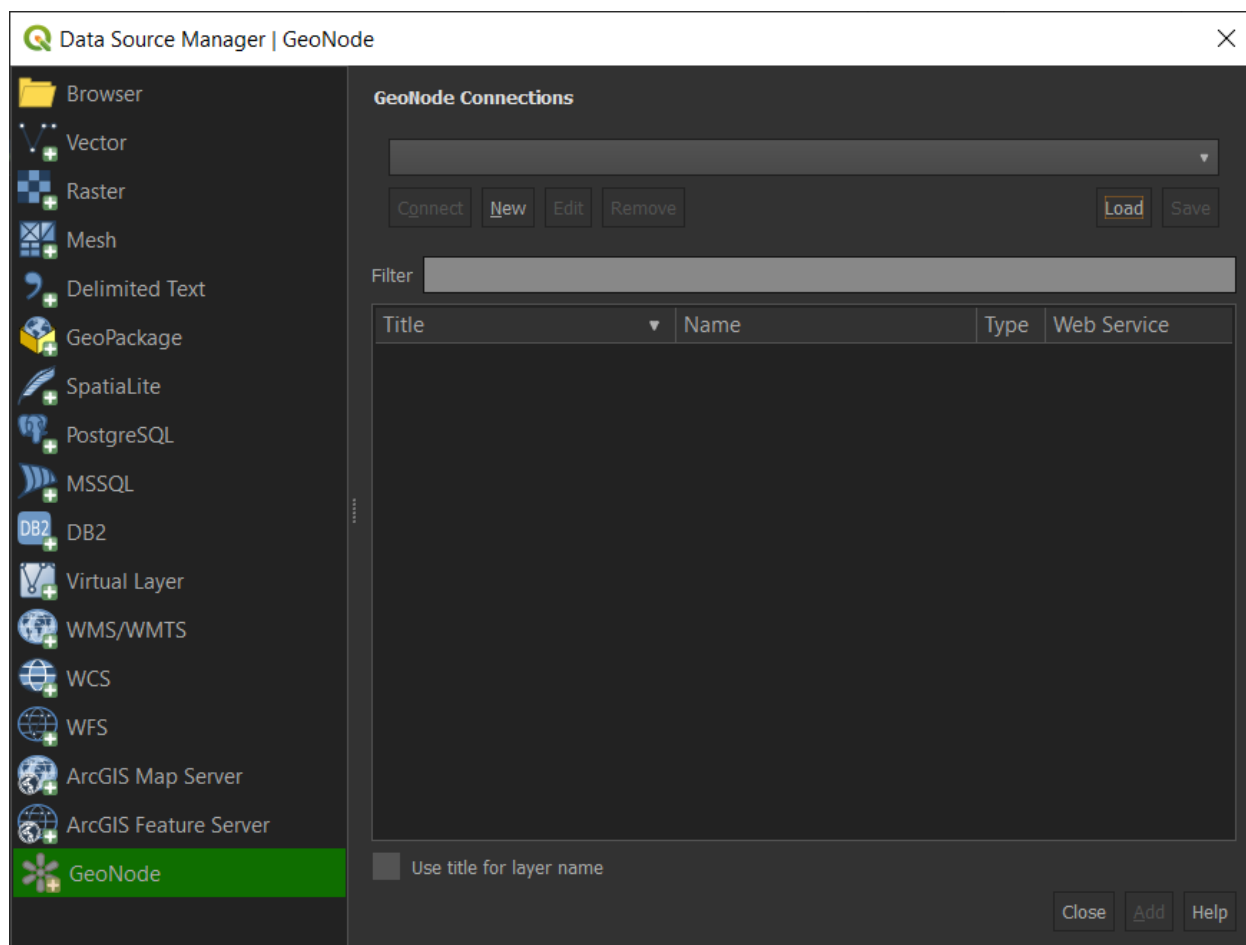


Fig. 235: Caixa de Diálogo do Gestor da Fonte de Dados

Create a New GeoNode Connection

Connection Details

Name:

URL:

WFS Options

Version:

Max. number of features:

☒ Enable feature paging

Page size:

☐ Ignore axis orientation (WFS 1.1/WFS 2.0)

☐ Invert axis orientation

WMS/WMTS Options

Referer:

DPI-Mode:

☐ Ignore GetMap/GetTile URI reported in capabilities

☐ Ignore GetFeatureInfo URI reported in capabilities

☐ Ignore axis orientation (WMS 1.3/WMTS)

☐ Ignore reported layer extents

☐ Invert axis orientation

☐ Smooth pixmap transform

Fig. 236: Detalhes da Caixa de Diálogo da instância do Geonode

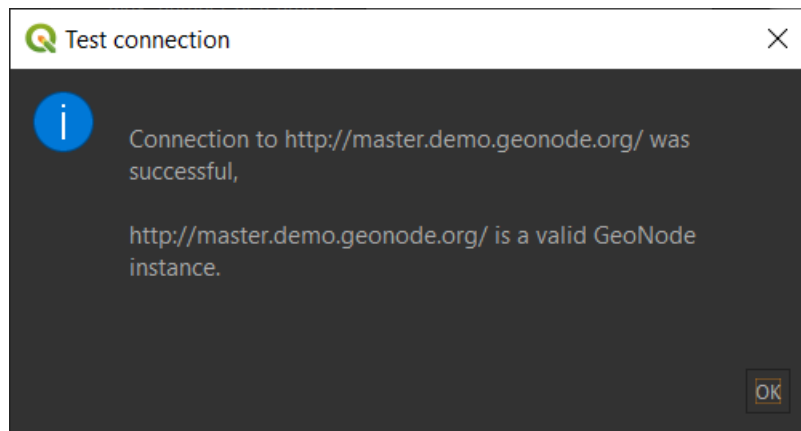


Fig. 237: Caixa de Diálogo de Conexão bem-sucedida

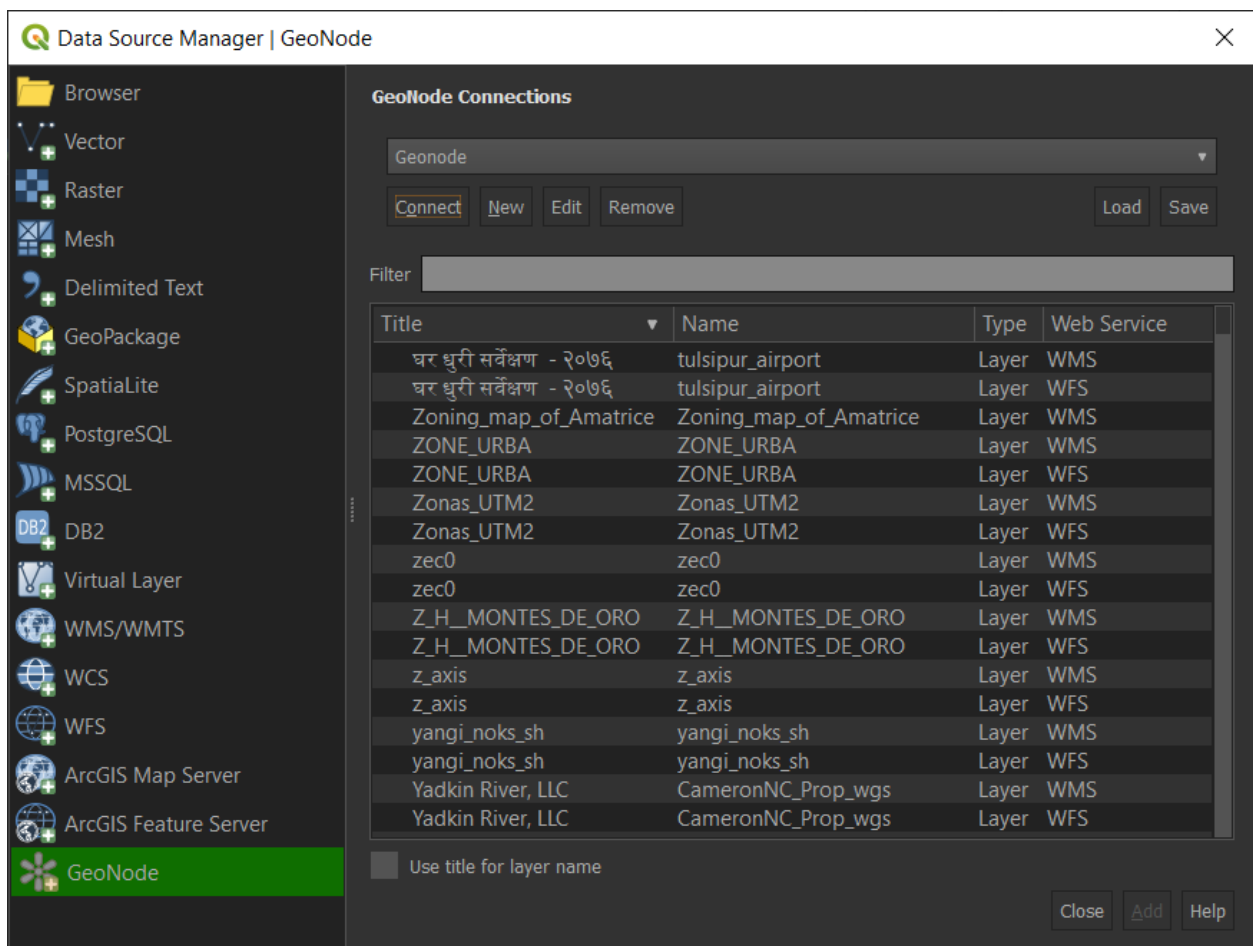


Fig. 238: Caixa de Diálogo de camadas de instância do Geonode

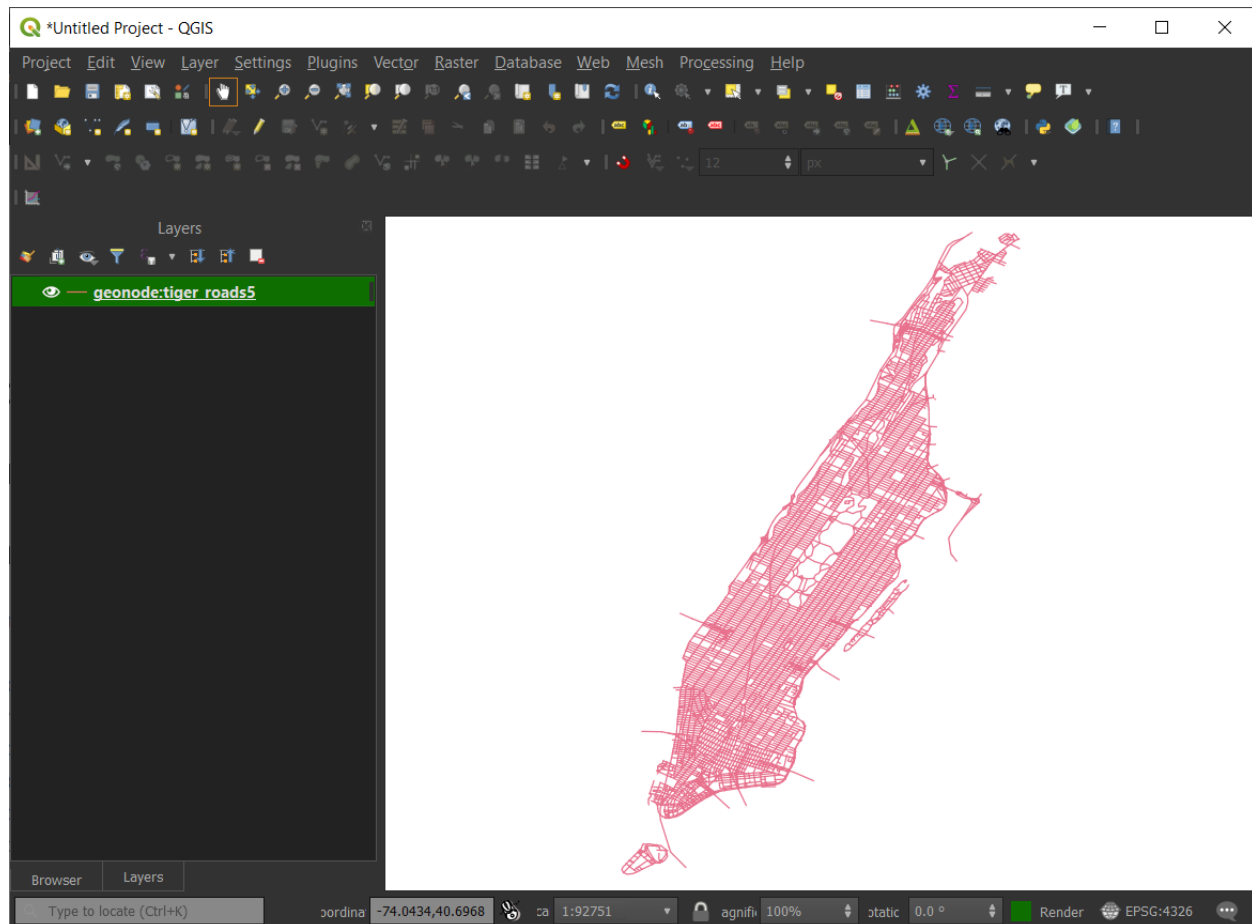


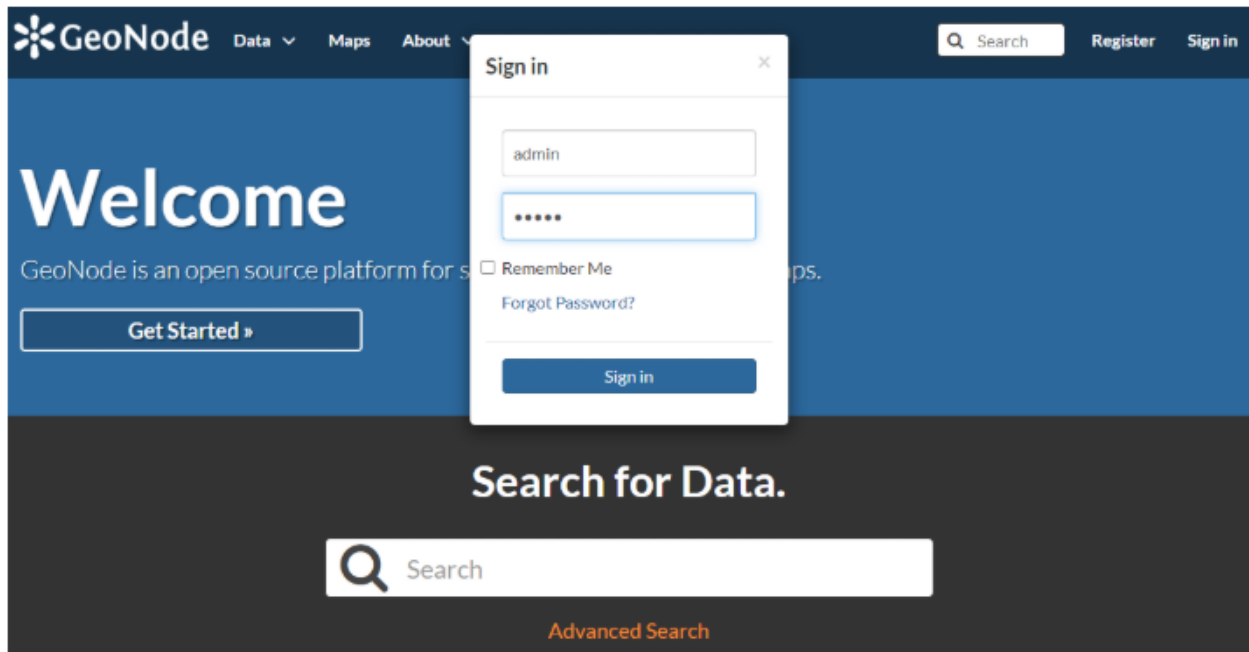
Fig. 239: Exemplo de camada Geonode

Aviso: Este procedimento só funciona com camadas públicas. Se as camadas forem para uso privado, é necessário fazer o qgis padrão, adicionar camadas WMS/WFS remotas (por meio do **Gestor da Fonte de Dados**) junto com o método de autenticação básico e terminais específicos.

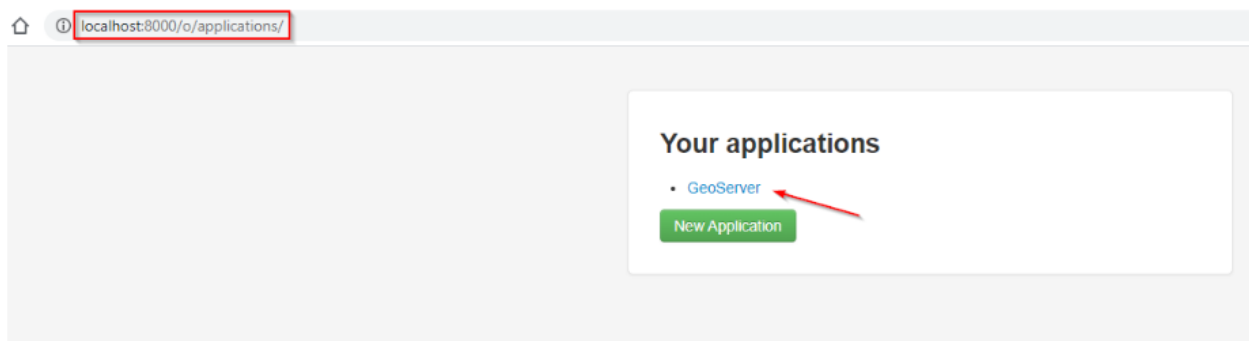
Conecte-se a camadas privadas usando OAuth2

Configuração do Aplicativo Cliente GeoNode OAuth2

Faça login no GeoNode como superutilizador



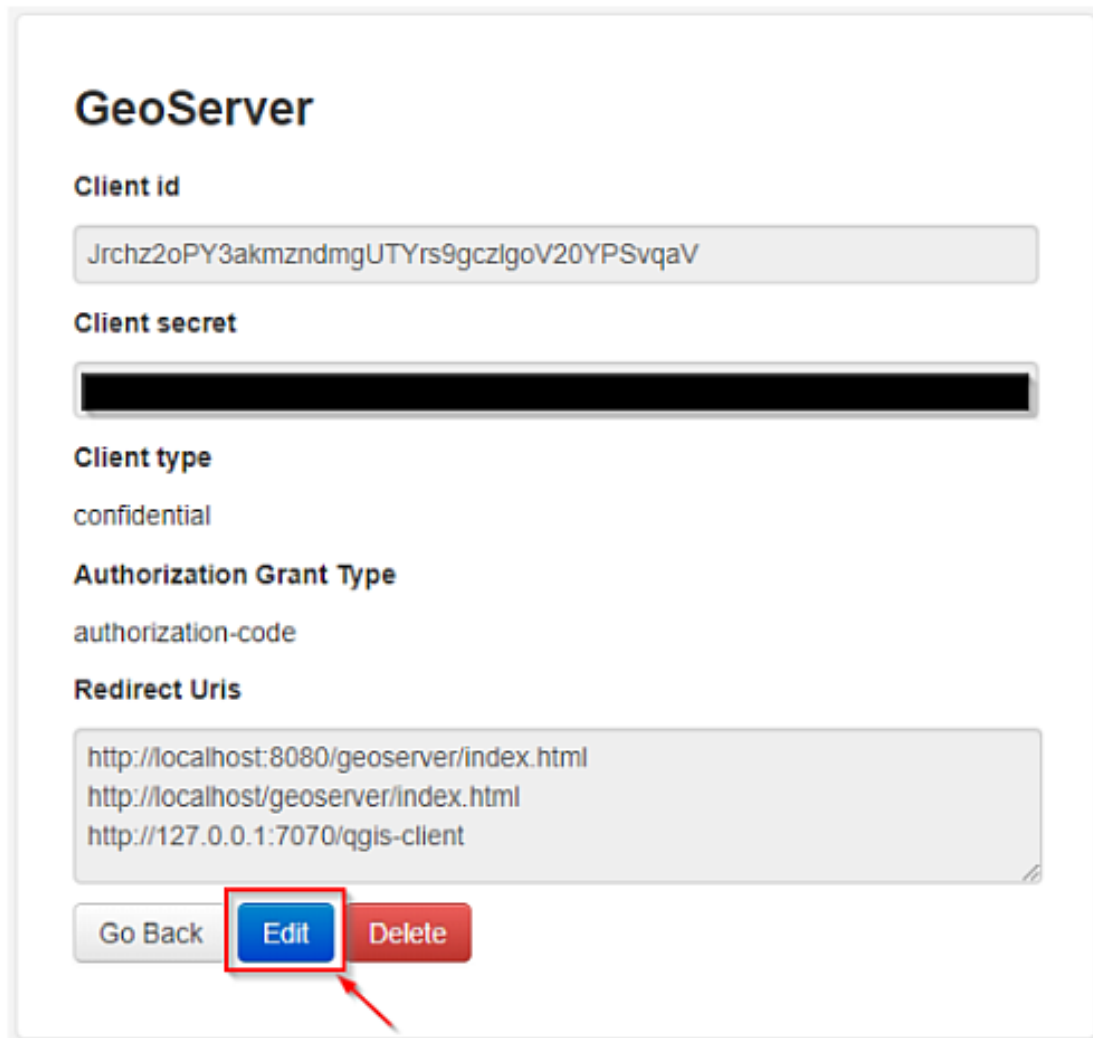
Navegue até `http://<geonode>/o/applications/`



Crie uma nova aplicação específica ou, melhor, edite o existente (“GeoServer”) com base no [Tipo de Concessão de Código de Autorização OAuth2](#)

Clique em “Editar” e adicione o URI de redirecionamento `http://127.0.0.1:7070/qgis-client` conforme mostrado abaixo

Nota: Isto é um exemplo. A porta e o caminho do URI podem ser personalizados. Eles devem ser iguais no GeoNode e no Cliente QGIS, conforme mostrado a seguir.



GeoServer

Client id

Jrchz2oPY3akmzndmgUTYrs9gczlgoV20YPSvqaV

Client secret

[Redacted]

Client type

confidential

Authorization Grant Type

authorization-code

Redirect Uris

http://localhost:8080/geoserver/index.html
http://localhost/geoserver/index.html
http://127.0.0.1:7070/qgis-client

Go Back Edit Delete

Além disso, você precisará das chaves *Client ID* e *Client Secret* posteriormente ao configurar o QGIS.

Configurar a Autenticação OAuth2 do Cliente Desktop QGIS

Abra o Cliente Desktop QGIS e adicione uma nova configuração de camada remota OWS

Crie uma nova conexão de serviço

Forneça os detalhes da conexão

Nota: É importante que o URL termine com /gs/ows

Quando terminar, clique em “+” para adicionar uma nova configuração de autenticação

Forneça as informações necessárias conforme mostrado abaixo:

Edit application GeoServer

Name

Client id

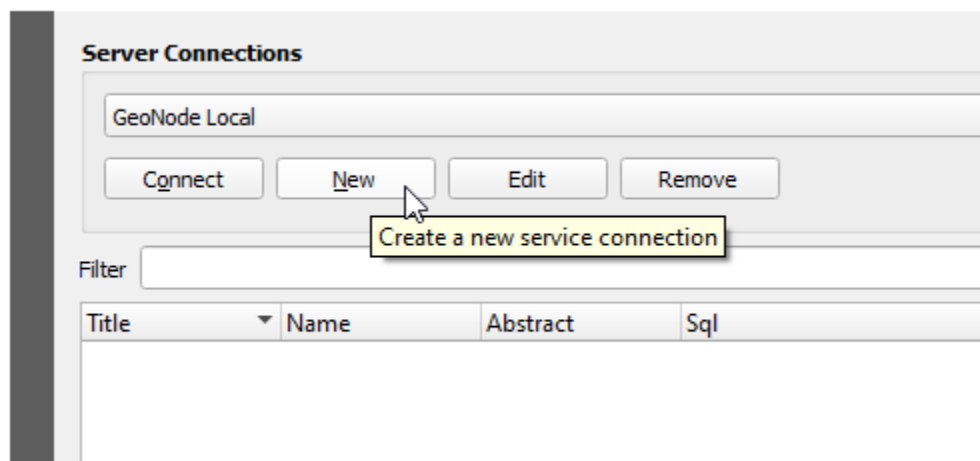
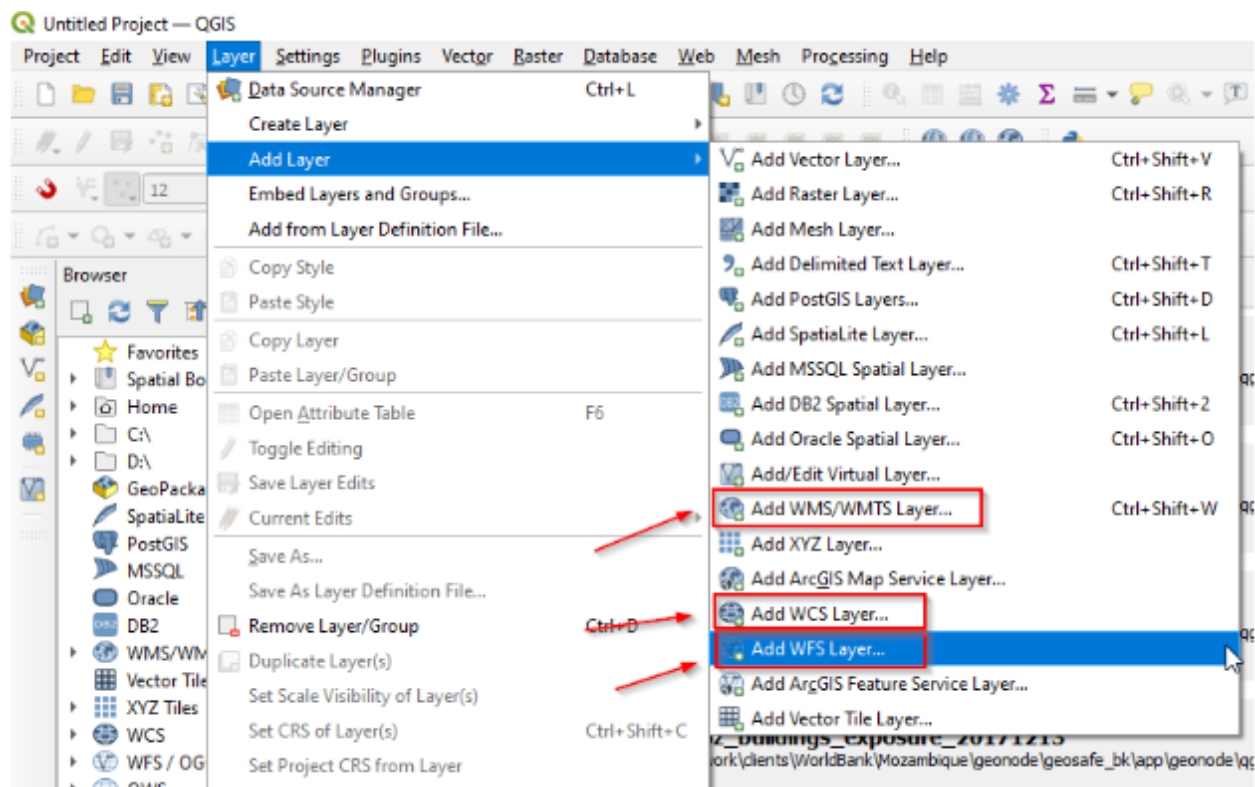
Client secret

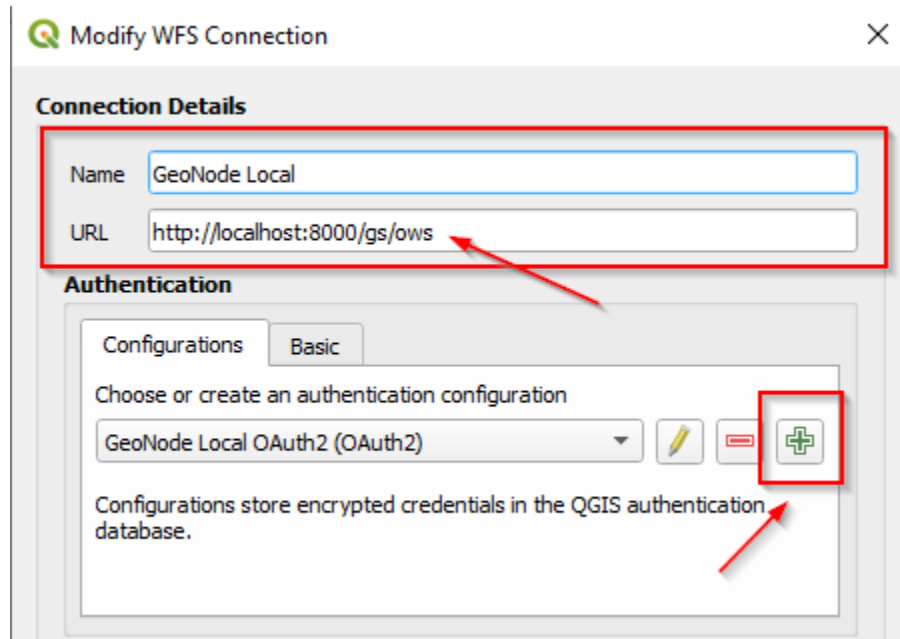
Client type

Authorization grant type

Redirect uris

Algorithm





- Nome: *qualquer string descritiva*
- Tipo: *Autenticação OAuth2*
- Fluxo de Concessão: *Código de Autorização*
- Requisitar URL: *deve terminar com /o/authorize/*
- URL de Token e URL de Actualização: *deve terminar com /o/token/*
- URL de redireccionamento: *deve corresponder ao definido acima no GeoNode*
- ID do cliente e Segredo do Cliente: *deve corresponder ao definido acima no GeoNode*
- Escopos: *escrita openid*
- Habilite a sessão de Token persistente por meio de Cabeçalhos

Salve e clique em “Connect”. O QGIS irá redirecioná-lo em uma página do navegador pedindo ao GeoNode para autenticar. Aprove as Reivindicações e volte ao QGIS.

Remover sessões de Token salvas do QGIS e fazer login com outro Utilizador

Edite a configuração QGIS

Clique em “lápiz”

Limpe o *Tokens* salvo e salve

Tente conectar novamente.

Authentication

Name: GeoNode Local OAuth2 Id: oopz0v0

Resource: Optional URL resource

OAuth2 authentication

Configure Defined Software Statement Tokens

Grant Flow: Authorization Code

Description:

Request URL: http://localhost:8000/o/authorize/

Token URL: http://localhost:8000/o/token/

Refresh Token URL: http://localhost:8000/o/token/

Redirect URL: http://127.0.0.1: 7070 /qgis-client

Client ID: 3rchz2oPY3akmzndmgUTYrs9gcZgoV20YPSvqaV

Client Secret:

Scope: openid write

API Key: Optional

Advanced

Token Session: ☒ Persist between launches

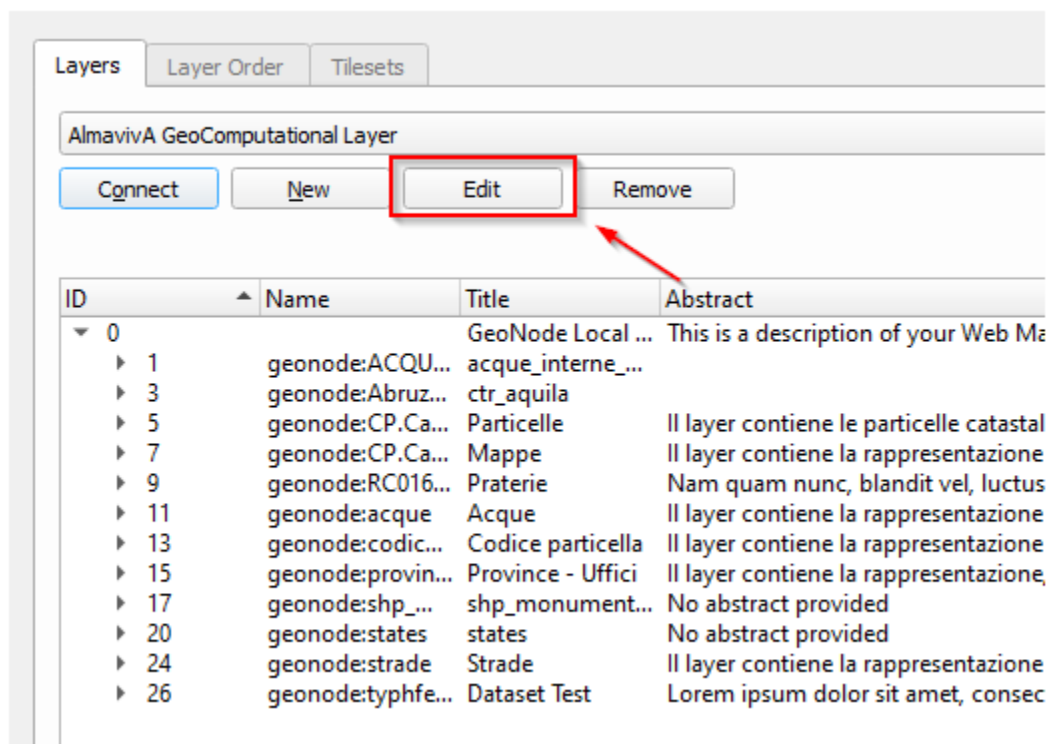
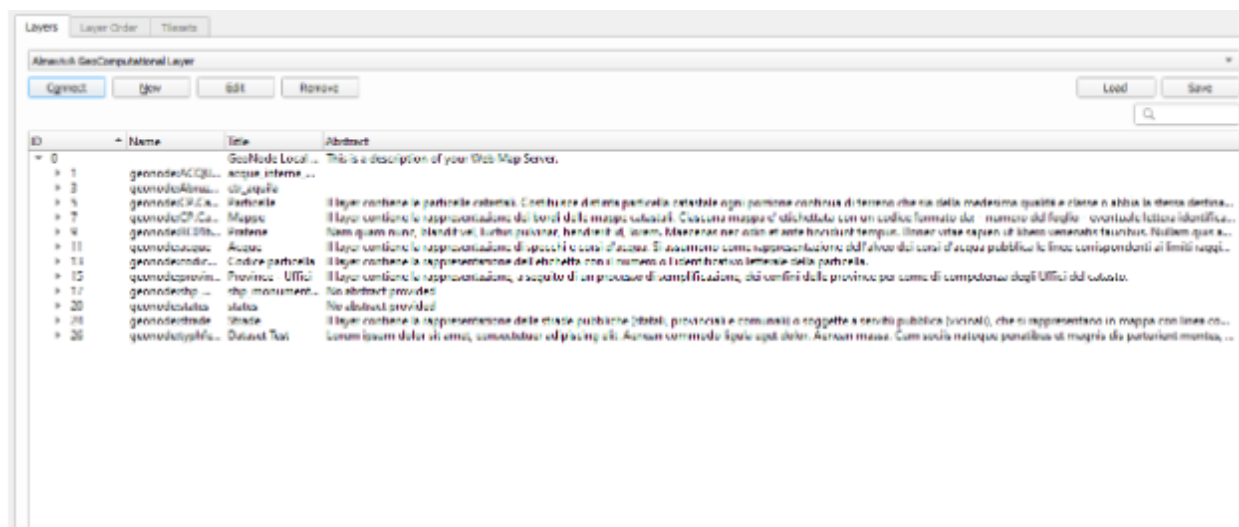
Access Method: Header

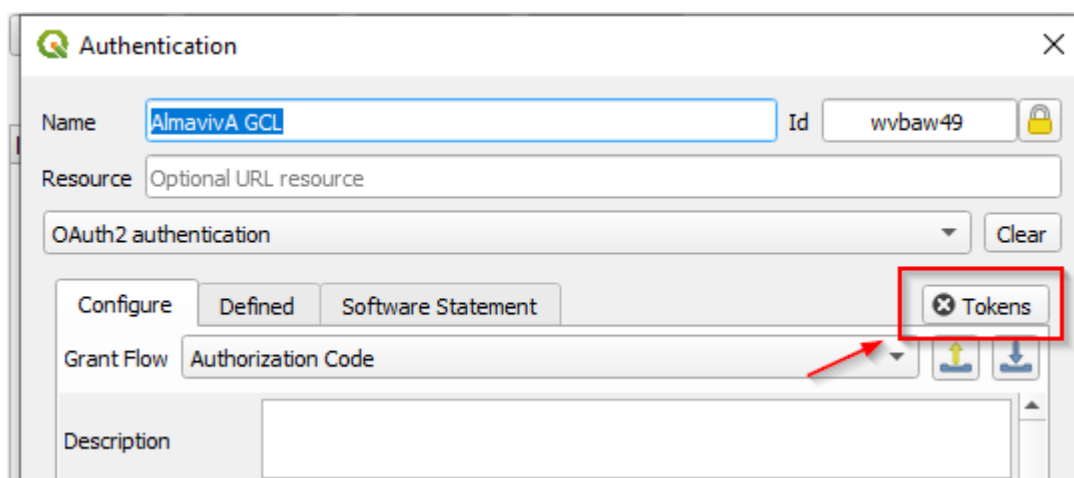
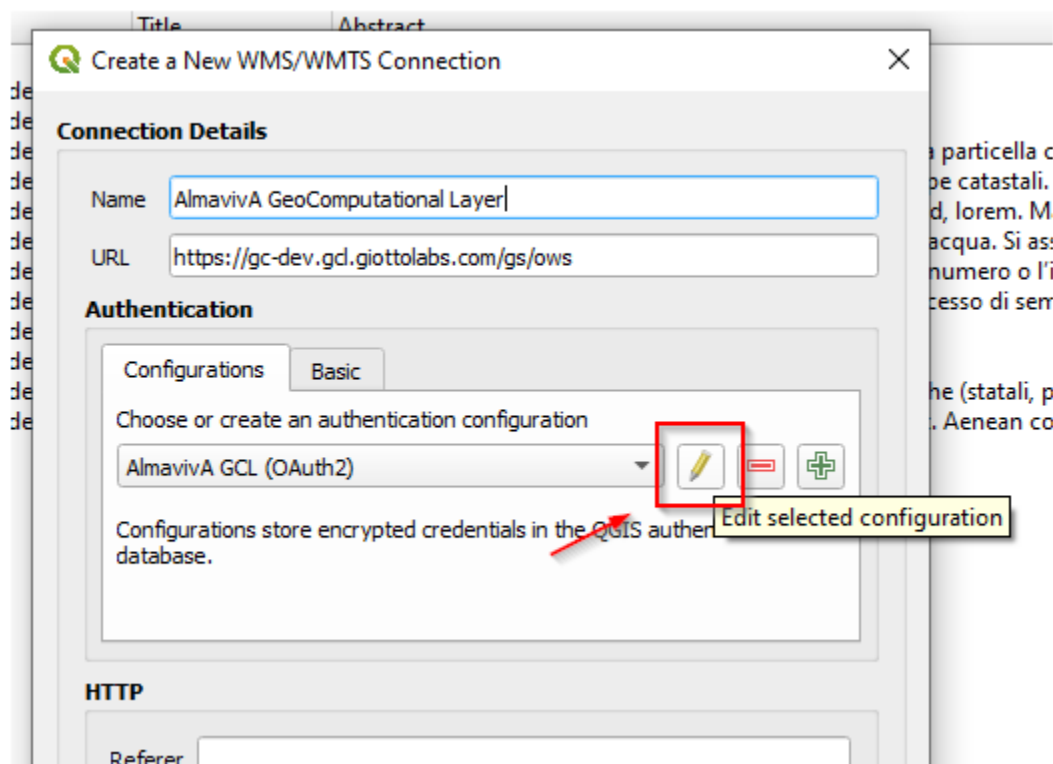
Request Timeout: 30 seconds

► Extra initial request parameters

Note: Saving writes directly to authentication database

Reset Save Cancel





GeoStory

GeoStory é uma ferramenta do MapStore integrada no GeoNode que possibilita ao utilizador criar histórias inspiradoras e imersivas combinando texto, mapas interactivos e outros conteúdos multimédia como imagens e vídeos entre outros conteúdos. Através desta ferramenta pode divulgar as suas histórias na web e depois publicá-las e partilhá-las com diferentes grupos do GeoNode ou publicá-las para todas as pessoas no mundo.

Para construir uma nova ou explorar uma GeoStory existente vá até ao menu *Apps* e dentro da página *Explore Apps* no botão de dropdown no topo direito *Create New Apps* escolher *GeoStory*.

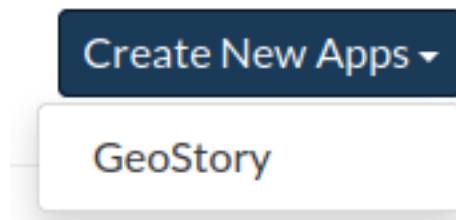


Fig. 240: Opção de Nova GeoStory

Agora entrou na página de edição da GeoStory que é composta das seguintes secções:

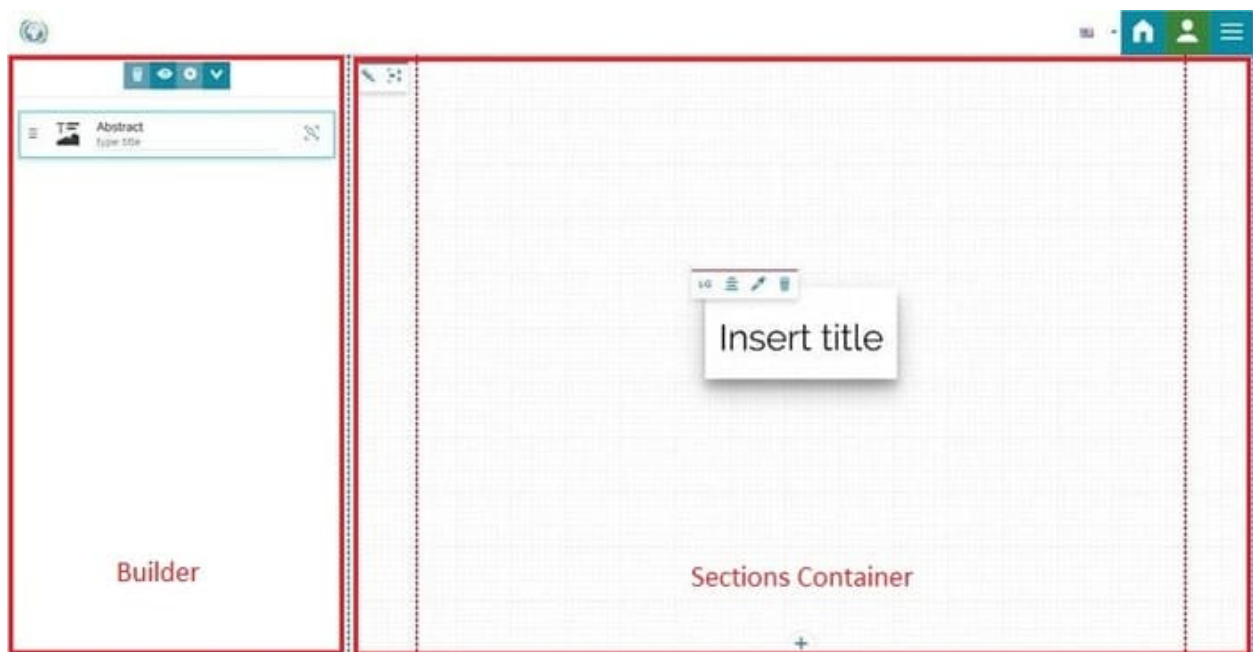


Fig. 241: Opção de Nova GeoStory

O conteúdo da GeoStory está organizado em Secções que podem ser adicionadas através do botão  na área de *Container*. Em particular, o utilizador poderá adicionar os seguintes tipos de secções à sua história:

Para mais informação sobre estas secções em específico por favor sigam a ligação para a documentação oficial do MapStore:

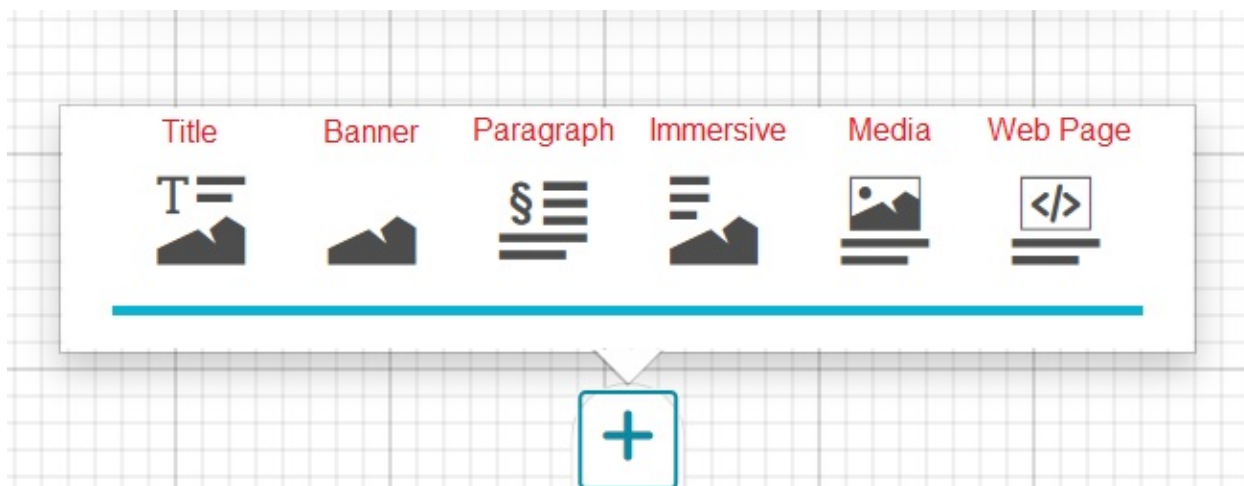


Fig. 242: Tipos de Secção de uma GeoStory

- Secção de Título
- Secção de Banner
- Secção de Parágrafo
- Secção Imersiva
- Secção de Média
- Secção de Página Web

Adicionar conteúdos do GeoNode na GeoStory

Com o GeoNode pode adicionar conteúdos para a sua GeoStory utilizando documentos e mapas internos no GeoNode bem como fontes externas. Esta possibilidade de adicionar conteúdos internos do GeoNode é muito útil na criação de uma GeoStory.

Para adicionar conteúdo à sua GeoStory utilizar o botão  no topo da secção da sua GeoStory.

Daí poderá adicionar *Images*, *Videos* e *Maps*. Para habilitar o catálogo interno do GeoNode, no menu dropdown *Services* escolher *GeoNode* como é exibido na imagem em baixo. Na lateral esquerda tem uma lista de documentos disponíveis com um campo de texto no topo de filtragem complementar.

Para salvar a GeoStory no botão hambúrguer do topo direito escolher *Save as...*

Agora a sua GeoStory já pode ser partilhada com todos!

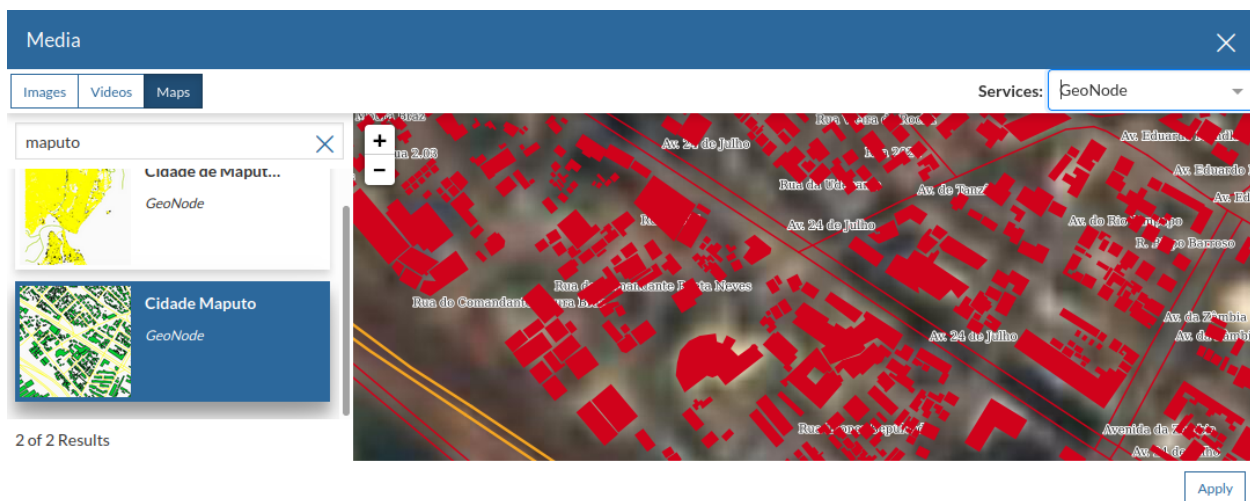


Fig. 243: Adicionar Media à GeoStory

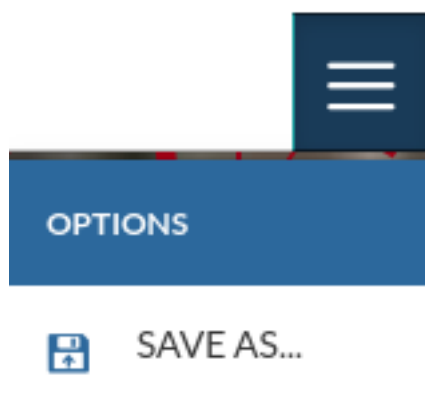


Fig. 244: Salvar a GeoStory

Leitura Complementar

Siga a ligação em baixo para ver com mais detalhe a informação de como utilizar o GeoStory.

[Documentação do GeoStory](#)

1.11 GeoNode Basic Installation

1.11.1 Overview

The followings are the easiest and recommended ways to deploy a full-stack GeoNode server on your host.

1. **First Step:** Deploy *GeoNode* on a local server, running as `http://localhost/` service. *GeoServer* will be also available at `http://localhost/geoserver/`
2. **Second Step:** Deploy *GeoNode* on a production server, running as `https://my_geonode.geonode.org/` service. *GeoServer* will be also available at `https://my_geonode.geonode.org/geoserver/`
3. **Third Step:** Customize `.env` to match your needs
4. **Fourth Step:** Secure your production deployment; change the *admin* passwords and *OAuth2* keys
5. **Further Production Enhancements**

1.11.2 First Step: Deploy GeoNode on a local server (e.g.: `http://localhost/`)

Ubuntu (18.0 +)

Nota: Recommended version 18.0.4 (Bionic Beaver).

Packages Installation

First, we are going to install all the **system packages** needed for the GeoNode setup. Login to the target machine and execute the following commands:

```
sudo apt install -y gdal-bin
sudo apt install -y python3-pip python3-dev python3-virtualenv python3-venv
↪virtualenvwrapper
sudo apt install -y libxml2 libxml2-dev gettext
sudo apt install -y libxslt1-dev libjpeg-dev libpng-dev libpq-dev libgdal-dev
sudo apt install -y software-properties-common build-essential
sudo apt install -y git unzip gcc zlib1g-dev libgeos-dev libproj-dev
sudo apt install -y sqlite3 spatialite-bin libsqlite3-mod-spatialite

# If the following does not work, you can skip it
sudo apt install -y libgdal20
```

Docker Setup (First time only)

```

sudo add-apt-repository universe
sudo apt-get update -y
sudo apt-get install -y git-core git-buildpackage debhelper devscripts
sudo apt-get install -y apt-transport-https ca-certificates curl gnupg-agent software-
↳properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
↳$(lsb_release -cs) stable"

sudo apt-get update -y
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-compose
sudo apt autoremove --purge

sudo usermod -aG docker ${USER}
su ${USER}

```

CentOS (7.0 +)

Nota: Recommended version 7.0 or higher.

Aviso: Accordingly to the version you use, the packages installation might be a bit different.

Packages Installation

First, we are going to install all the **system packages** needed for the GeoNode setup. Login to the target machine and execute the following commands:

```

sudo yum -y install epel-release
sudo yum install -y gdal
sudo yum install -y python3-pip python3-dev python3-virtualenv python3-venv
↳virtualenvwrapper
sudo pip3 install -U pip
sudo pip3 install -U virtualenv
sudo yum install -y libxml2 libxml2-dev gettext
sudo yum install -y libxslt1-dev libjpeg-dev libpng-dev libpq-dev libgdal-dev
sudo yum install -y git unzip gcc zlib1g-dev libgeos-dev libproj-dev

# If the following does not work, you can skip it
sudo apt install -y libgdal20

```

Docker Setup (First time only)

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.
↪repo
sudo yum install docker-ce docker-ce-cli containerd.io
sudo systemctl start docker

sudo curl -L "https://github.com/docker/compose/releases/download/1.23.1/docker-
↪compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

sudo usermod -aG docker ${USER}
su ${USER}
```

Create an instance of your geonode-project

Let's say your project is named *my_geonode* perform the following steps:

```
git clone https://github.com/GeoNode/geonode-project.git -b 3.2.x

# Ubuntu
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
mkvirtualenv --python=/usr/bin/python3 my_geonode

Alternatively you can also create the virtual env like below
python3.8 -m venv /home/geonode/dev/.venvs/my_geonode
source /home/geonode/dev/.venvs/my_geonode/bin/activate

pip install Django==3.2

# CentOS
virtualenv -p python3 my_geonode
source my_geonode/bin/activate

django-admin startproject --template=./geonode-project -e py,sh,md,rst,json,yml,ini,
↪env,sample,properties -n monitoring-cron -n Dockerfile my_geonode

# If the previous command does not work for some reason, try the following one
python -m django startproject --template=./geonode-project -e py,sh,md,rst,json,yml,
↪ini,env,sample,properties -n monitoring-cron -n Dockerfile my_geonode
```

Startup the containers

```
cd my_geonode
./docker-build.sh
```

- You can follow the containers startup by running the following commands from *my_geonode* root folder:

```
# GeoNode Container
docker-compose logs -f django

# GeoServer Container
docker-compose logs -f geoserver
```

(continues on next page)

(continuação da página anterior)

```
# DB Container
docker-compose logs -f db

# NGINX Container
docker-compose logs -f geonode
```

- If any error occurs, try to catch the error stacktrace by running the following commands from my_geonode root folder:

```
# GeoNode "entrypoint.sh" Logs
tail -F -n 300 invoke.log
```

Connect to <http://localhost/>

The startup typically takes some time, so be patient...

If everything goes well, you should be able to see from the `geonode startup logs` a line similar to the following one:

```
<some date> [UWSGI] Uwsgi running...
```

Connect to <http://localhost/>

The default credentials are:

- GeoNode (<http://localhost/>) admin:


```
username:  admin
password:  admin
```
- GeoServer (<http://localhost/geoserver/>) admin:


```
username:  admin
password:  geoserver
```

1.11.3 Second Step: Deploy GeoNode on a production server (e.g.: https://my_geonode.geonode.org/)

In the case you would like to deploy to, let's say, https://my_geonode.geonode.org/, you will need to change `.env` as follows:

```
--- geonode-project\.env
+++ my_geonode\.env
@@ -1,7 +1,7 @@
-COMPOSE_PROJECT_NAME={{project_name}}
+COMPOSE_PROJECT_NAME=my_geonode
BACKUPS_VOLUME_DRIVER=local

DOCKER_HOST_IP=
DOCKER_ENV=production
# See https://github.com/geosolutions-it/geonode-generic/issues/28
# to see why we force API version to 1.24
@@ -9,40 +9,40 @@

C_FORCE_ROOT=1
IS_CELERY=false
```

(continues on next page)

(continuação da página anterior)

```

IS_FIRST_START=true
FORCE_REINIT=false

-SITEURL=http://localhost/
+SITEURL=https://my_geonode.geonode.org/
ALLOWED_HOSTS=['django',]

# LANGUAGE_CODE=pt
# LANGUAGES=(('en', 'English'), ('pt', 'Portuguese'))

GEONODE_INSTANCE_NAME=geonode
-DJANGO_SETTINGS_MODULE={{project_name}}.settings
-UWSGI_CMD=uwsgi --ini /usr/src/{{project_name}}/uwsgi.ini
+DJANGO_SETTINGS_MODULE=my_geonode.settings
+UWSGI_CMD=uwsgi --ini /usr/src/my_geonode/uwsgi.ini

# #####
# backend
# #####
-GEONODE_DATABASE={{project_name}}
+GEONODE_DATABASE=my_geonode
GEONODE_DATABASE_PASSWORD=geonode
-GEONODE_GEODATABASE={{project_name}}_data
+GEONODE_GEODATABASE=my_geonode_data
GEONODE_GEODATABASE_PASSWORD=geonode

-DATABASE_URL=postgis://{{project_name}}:geonode@db:5432/{{project_name}}
-GEODATABASE_URL=postgis://{{project_name}}_data:geonode@db:5432/{{project_name}}_data
+DATABASE_URL=postgis://my_geonode:geonode@db:5432/my_geonode
+GEODATABASE_URL=postgis://my_geonode_data:geonode@db:5432/my_geonode_data
DEFAULT_BACKEND_DATASTORE=datastore
BROKER_URL=amqp://guest:guest@rabbitmq:5672/

# #####
# geoserver
# #####
-GEOSERVER_WEB_UI_LOCATION=http://localhost/geoserver/
-GEOSERVER_PUBLIC_LOCATION=http://localhost/geoserver/
+GEOSERVER_WEB_UI_LOCATION=https://my_geonode.geonode.org/geoserver/
+GEOSERVER_PUBLIC_LOCATION=https://my_geonode.geonode.org/geoserver/
GEOSERVER_LOCATION=http://geoserver:8080/geoserver/
GEOSERVER_ADMIN_PASSWORD=geoserver

OGC_REQUEST_TIMEOUT=30
OGC_REQUEST_MAX_RETRIES=1
OGC_REQUEST_BACKOFF_FACTOR=0.3
@@ -58,50 +58,50 @@
MOSAIC_ENABLED=False

# #####
# nginx
# HTTPD Server
# #####
-GEONODE_LB_HOST_IP=localhost
+GEONODE_LB_HOST_IP=my_geonode.geonode.org
GEONODE_LB_PORT=80

```

(continues on next page)

(continuação da página anterior)

```

# IP or domain name and port where the server can be reached on HTTPS (leave HOST_
↳empty if you want to use HTTP only)
# port where the server can be reached on HTTPS
-HTTP_HOST=localhost
-HTTPS_HOST=
+HTTP_HOST=
+HTTPS_HOST=my_geonode.geonode.org

HTTP_PORT=80
HTTPS_PORT=443

# Let's Encrypt certificates for https encryption. You must have a domain name as_
↳HTTPS_HOST (doesn't work
# with an ip) and it must be reachable from the outside. This can be one of the_
↳following :
# disabled : we do not get a certificate at all (a placeholder certificate will be_
↳used)
# staging : we get staging certificates (are invalid, but allow to test the process_
↳completely and have much higher limit rates)
# production : we get a normal certificate (default)
-LESENCRYPT_MODE=disabled
+# LESENCRYPT_MODE=disabled
# LESENCRYPT_MODE=staging
-# LESENCRYPT_MODE=production
+LESENCRYPT_MODE=production

RESOLVER=127.0.0.11

# #####
# Security
# #####
# Admin Settings
ADMIN_PASSWORD=admin
-ADMIN_EMAIL=admin@localhost
+ADMIN_EMAIL=admin@my_geonode.geonode.org

# EMAIL Notifications
EMAIL_ENABLE=False
DJANGO_EMAIL_BACKEND=django.core.mail.backends.smtp.EmailBackend
DJANGO_EMAIL_HOST=localhost
DJANGO_EMAIL_PORT=25
DJANGO_EMAIL_HOST_USER=
DJANGO_EMAIL_HOST_PASSWORD=
DJANGO_EMAIL_USE_TLS=False
DJANGO_EMAIL_USE_SSL=False
-DEFAULT_FROM_EMAIL='GeoNode <no-reply@geonode.org>'
+DEFAULT_FROM_EMAIL='GeoNode <no-reply@my_geonode.geonode.org>'

# Session/Access Control
LOCKDOWN_GEONODE=False
CORS_ORIGIN_ALLOW_ALL=True
SESSION_EXPIRED_CONTROL_ENABLED=True
DEFAULT_ANONYMOUS_VIEW_PERMISSION=True

```

Restart the containers

Whenever you change something on `.env` file, you will need to rebuild the container

Aviso: Be careful! The following command drops any change you might have done manually inside the containers, except for the static volumes.

```
docker-compose up -d
```

Troubleshooting

If for some reason you are not able to reach the server on the *HTTPS* channel, please check the *NGINX* configuration files below:

1. Enter the *NGINX* container

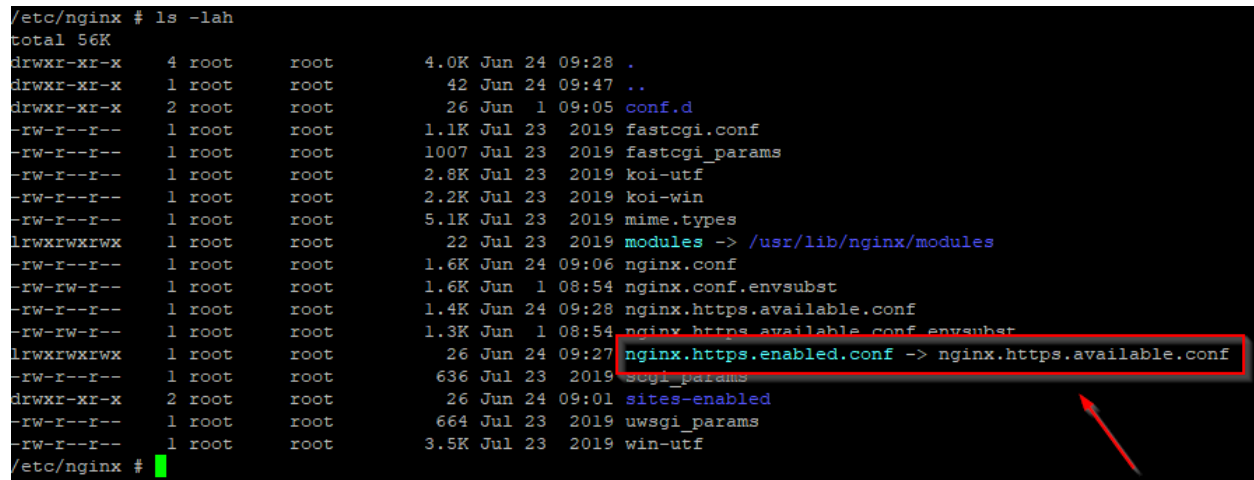
```
docker-compose exec geonode sh
```

2. Install an editor if not present

```
apk add nano
```

3. Double check that the `nginx.https.enabled.conf` link has been correctly created

```
ls -lah
```



```
/etc/nginx # ls -lah
total 56K
drwxr-xr-x  4 root   root    4.0K Jun 24 09:28 .
drwxr-xr-x  1 root   root    42 Jun 24 09:47 ..
drwxr-xr-x  2 root   root    26 Jun  1 09:05 conf.d
-rw-r--r--  1 root   root    1.1K Jul 23 2019 fastcgi.conf
-rw-r--r--  1 root   root   1007 Jul 23 2019 fastcgi_params
-rw-r--r--  1 root   root    2.8K Jul 23 2019 koi-utf
-rw-r--r--  1 root   root    2.2K Jul 23 2019 koi-win
-rw-r--r--  1 root   root    5.1K Jul 23 2019 mime.types
lrwxrwxrwx  1 root   root     22 Jul 23 2019 modules -> /usr/lib/nginx/modules
-rw-r--r--  1 root   root    1.6K Jun 24 09:06 nginx.conf
-rw-rw-r--  1 root   root    1.6K Jun  1 08:54 nginx.conf.envsubst
-rw-r--r--  1 root   root    1.4K Jun 24 09:28 nginx.https.available.conf
-rw-rw-r--  1 root   root    1.3K Jun  1 08:54 nginx.https.available.conf.envsubst
lrwxrwxrwx  1 root   root    26 Jun 24 09:27 nginx.https.enabled.conf -> nginx.https.available.conf
-rw-r--r--  1 root   root    636 Jul 23 2019 scgi_params
drwxr-xr-x  2 root   root    26 Jun 24 09:01 sites-enabled
-rw-r--r--  1 root   root    664 Jul 23 2019 uwsgi_params
-rw-r--r--  1 root   root    3.5K Jul 23 2019 win-utf
/etc/nginx #
```

If the list does not match exactly the figure above, please run the following commands, and check again

```
rm nginx.https.enabled.conf
ln -s nginx.https.available.conf nginx.https.enabled.conf
```

4. Inspect the `nginx.https.enabled.conf` contents

```
nano nginx.https.enabled.conf
```


Make sure the contents match the following

Aviso: Change the *Hostname* accordingly. **This is only an example!**

```
# NOTE : $VARIABLES are env variables replaced by entrypoint.sh using
↳envsubst
# not to be mistaken for nginx variables (also starting with $, but
↳usually lowercase)

# This file is to be included in the main nginx.conf configuration if
↳HTTPS_HOST is set
ssl_session_cache    shared:SSL:10m;
ssl_session_timeout  10m;

# this is the actual HTTPS host
server {
    listen              443 ssl;
    server_name         my_geonode.geonode.org;
    keepalive_timeout   70;

    ssl_certificate     /certificate_symlink/fullchain.pem;
    ssl_certificate_key /certificate_symlink/privkey.pem;
    ssl_protocols       TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers         HIGH:!aNULL:!MD5;

    include sites-enabled/*.conf;
}

# if we try to connect from http, we redirect to https
server {
    listen 80;
    server_name my_geonode.geonode.org; # TODO : once geoserver supports
↳relative urls, we should allow access though both HTTP and HTTPS at the
↳same time and hence remove HTTP_HOST from this line

    # Except for let's encrypt challenge
    location /.well-known {
        alias /geonode-certificates/.well-known;
        include /etc/nginx/mime.types;
    }

    # Redirect to https
    location / {
        return 302 https://my_geonode.geonode.org/$request_uri; # TODO : we
↳should use 301 (permanent redirect, but not practical for debug)
    }
}
```

Aviso: Save the changes, if any, and exit!

5. Reload the NGINX configuration

```
nginx -s reload
2020/06/24 10:00:11 [notice] 112#112: signal process started
```

(continues on next page)

(continuação da página anterior)

```
/etc/nginx# exit
```

6. It may be helpful to disable https to isolate the source of errors. After reverting the HTTPS-related changes in the `.env` file, repeat the above steps and ensure that the `nginx.http.enabled.conf` link has been correctly created.

```
ln -s nginx.conf nginx.http.enabled.conf
nano nginx.http.enabled.conf
```

1.11.4 Third Step: Customize `.env` to match your needs

In the case you would like to modify the GeoNode behavior, always use the `.env` file in order to update the *settings*.

If you need to change a setting which does not exist in `.env`, you can force the values inside `my_geonode/settings.py`

Refer to the section: *Settings*

You can add here any property referred as

Env: `PROPERTY_NAME`

Restart the containers

Whenever you change something on `.env` file, you will need to rebuild the containers.

Aviso: Be careful! The following command drops any change you might have done manually inside the containers, except for the static volumes.

```
docker-compose up -d django
```

1.11.5 Fourth Step: Secure your production deployment; change the *admin* passwords and *OAuth2* keys

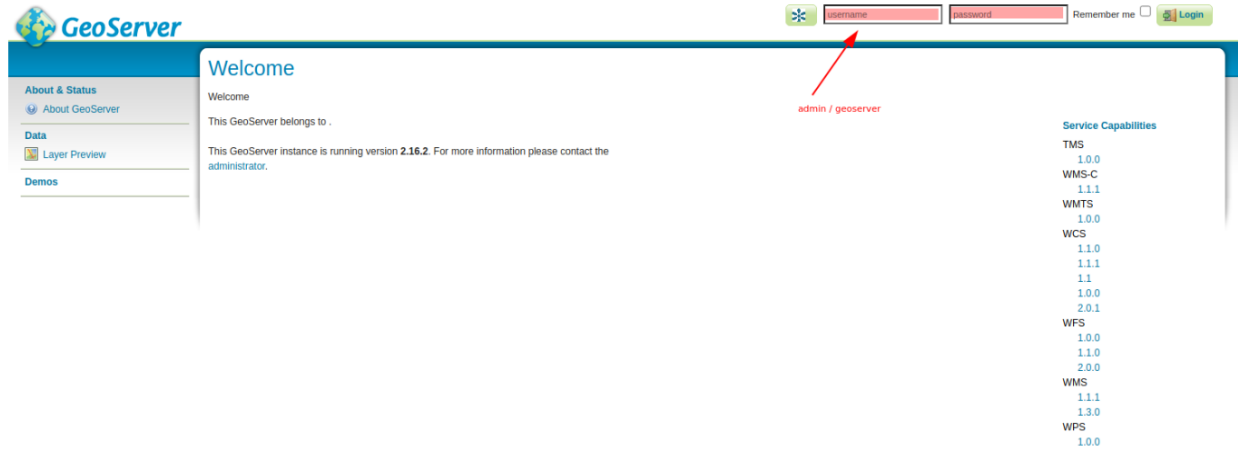
GeoServer Setup

Admin Password Update

OAuth2 REST API Key

Nota: In order to generate new strong random passwords you can use an online service like <https://passwordsgenerator.net/>

Avoid using Symbols (e.g. `@#$$%`) as they might conflict with `.env` file



GeoServer Disk Quota

Update the passwords and keys on .env file

Nota: In order to generate new strong random passwords you can use an online service like <https://passwordsgenerator.net/>

Avoid using Symbols (e.g. @\$%\$) as they might conflict with .env file

```
--- my_geonode\.env
+++ my_geonode\.prod.env
@@ -6,13 +6,13 @@
# See https://github.com/geosolutions-it/geonode-generic/issues/28
# to see why we force API version to 1.24
DOCKER_API_VERSION="1.24"

C_FORCE_ROOT=1
IS_CELERY=false
-IS_FIRST_START=true
+IS_FIRST_START=false
FORCE_REINIT=false

SITEURL=https://my_geonode.geonode.org/
ALLOWED_HOSTS=['django',]

# LANGUAGE_CODE=pt
@@ -38,13 +38,14 @@
# #####
# geoserver
# #####
GEOSERVER_WEB_UI_LOCATION=https://my_geonode.geonode.org/geoserver/
GEOSERVER_PUBLIC_LOCATION=https://my_geonode.geonode.org/geoserver/
GEOSERVER_LOCATION=http://geoserver:8080/geoserver/
-GEOSERVER_ADMIN_PASSWORD=geoserver
+GEOSERVER_ADMIN_USER=admin
+GEOSERVER_ADMIN_PASSWORD=<new_geoserver_admin_password>
```

(continues on next page)

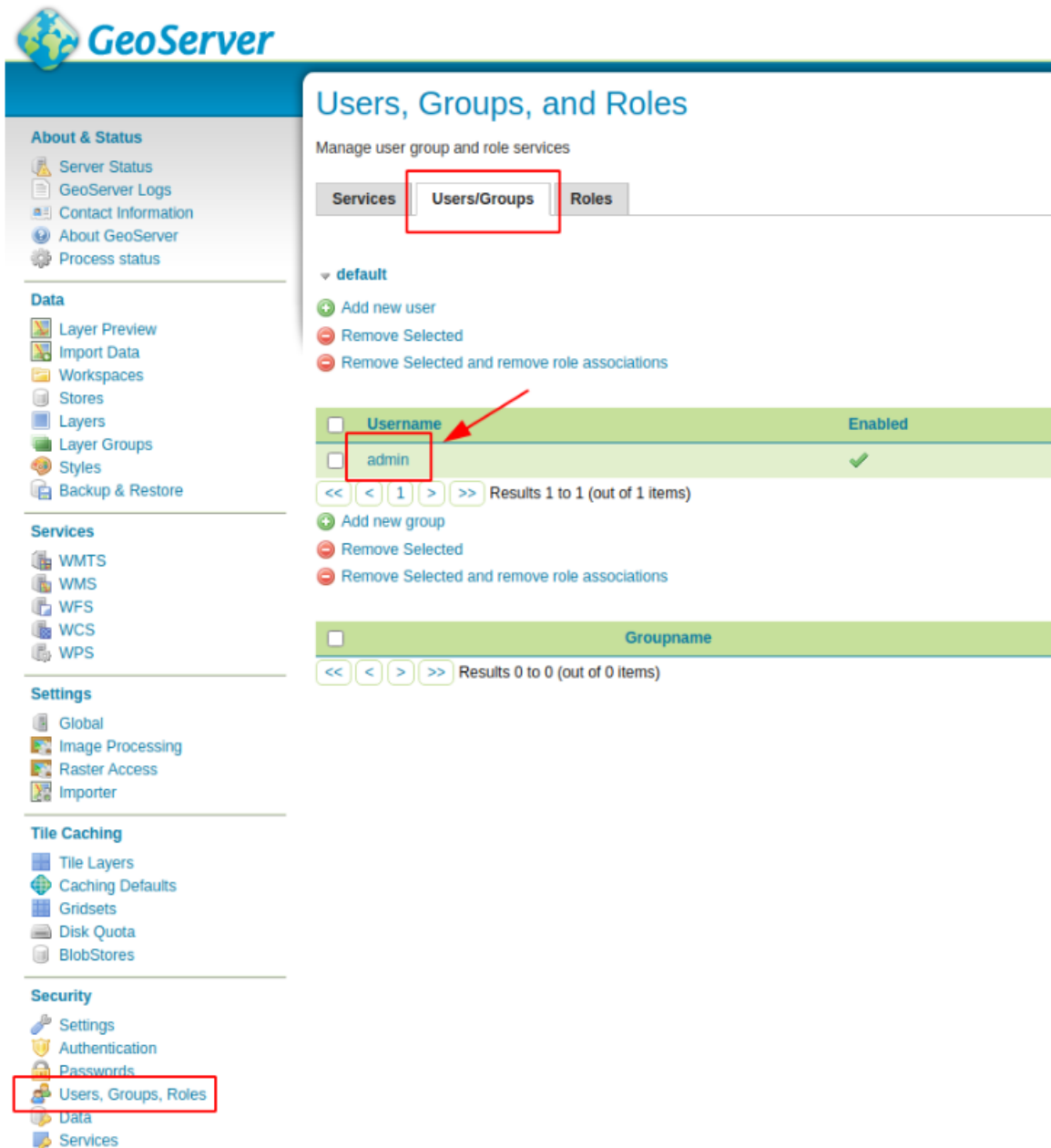


Fig. 245: GeoServer Admin Password Update

GeoServer

About & Status

- Server Status
- GeoServer Logs
- Contact Information
- About GeoServer
- Process status

Data

- Layer Preview
- Import Data
- Workspaces
- Stores
- Layers
- Layer Groups
- Styles
- Backup & Restore

Services

- WMTS
- WMS
- WFS
- WCS
- WPS

Settings

- Global
- Image Processing
- Raster Access
- Importer

Tile Caching

- Tile Layers
- Caching Defaults
- Gridsets
- Disk Quota
- BlobStores

Security

- Settings
- Authentication
- Passwords
- Users, Groups, Roles**
- Data
- Services
- WPS security

AuthKEY REST Role Service geonode REST role service

Role service from REST endpoint

Settings **Roles**

Name
geonode REST role service

Administrator role
ROLE_ADMIN

Group administrator role
ROLE_ADMIN

REST Role Service Settings

Base Server URL
http://geonode:80

Roles REST Endpoint
/api/roles

Admin Role REST Endpoint
/api/adminRole

Users REST Endpoint
/api/users

Roles JSON Path
\$.groups

Admin Role JSON Path
\$.adminRole

Users JSON Path
\$.users[?(@.username=="\${username}")].groups

REST Rules Cache Concurrency Level
4

REST Rules Cache Maximum Size (# keys)
60000

REST Rules Cache Expiration Time (ms)
60000

REST Api Key (optional)
87j3JvaCqjzXR9Le68CH6h4

Fig. 246: OAuth2 REST API Key Update

GeoServer

About & Status

- Server Status
- GeoServer Logs
- Contact Information
- About GeoServer
- Process status

Data

- Layer Preview
- Import Data
- Workspaces
- Stores
- Layers
- Layer Groups
- Styles
- Backup & Restore

Services

- WMTS
- WMS
- WFS
- WCS
- WPS

Settings

- Global
- Image Processing
- Raster Access
- Importer

Tile Caching

- Tile Layers
- Caching Defaults
- Gridsets
- Disk Quota**
- BlobStores

Security

- Settings
- Authentication
- Passwords
- Users, Groups, Roles
- Data
- Services
- WPS security

Disk Quota

Configure the disk quota limits and expiration policy for the tile cache

Disk Quota

☒ Enable disk quota

Disk quota check frequency:
10 Seconds
(Last run: 5 s ago.)

Maximum tile cache size
500 MIB

Using 0.0 B of a maximum 500.0 MB

When enforcing disk quota limits, remove tiles that are:

☐ Least frequently used

☒ Least recently used

Disk quota store type

External database

Target database type
PostgreSQL

JDBC data source
GeoServer managed connection pool

JDBC Driver class name
org.postgresql.Driver

JDBC connection URL
jdbc:postgresql://db:5432/geonode_data

User name
geonode

Password

Min. connections
1

Max. connections
10

Connection time out (ms)
10000

Validation query

Max open prepared statements
10

Fig. 247: GeoServer Disk Quota Update

(continuação da página anterior)

```

OGC_REQUEST_TIMEOUT=30
OGC_REQUEST_MAX_RETRIES=1
OGC_REQUEST_BACKOFF_FACTOR=0.3
OGC_REQUEST_POOL_MAXSIZE=10
OGC_REQUEST_POOL_CONNECTIONS=10
@@ -84,13 +85,13 @@
RESOLVER=127.0.0.11

# #####
# Security
# #####
# Admin Settings
-ADMIN_PASSWORD=admin
+ADMIN_PASSWORD=<new_geonode_admin_password>
ADMIN_EMAIL=admin@my_geonode.geonode.org

# EMAIL Notifications
EMAIL_ENABLE=False
DJANGO_EMAIL_BACKEND=django.core.mail.backends.smtp.EmailBackend
DJANGO_EMAIL_HOST=localhost
@@ -114,15 +115,15 @@
ACCOUNT_CONFIRM_EMAIL_ON_GET=False
ACCOUNT_EMAIL_VERIFICATION=optional
ACCOUNT_EMAIL_CONFIRMATION_EMAIL=False
ACCOUNT_EMAIL_CONFIRMATION_REQUIRED=False

# OAuth2
-OAUTH2_API_KEY=
-OAUTH2_CLIENT_ID=Jrchz2oPY3akmzndmgUTYrs9gczlgoV20YPSvqaV
-OAUTH2_CLIENT_
→SECRET=rCnp5txobUo83EpQEblM8fVj3QT5zb5qRfxNsuPzCqZaiRyIoxM4jdgMiZKffePBHYXCLd7B8N1kfDBY9HKeIQPcy5Cp
+OAUTH2_API_KEY=<new_OAUTH2_API_KEY>
+OAUTH2_CLIENT_ID=<new_OAUTH2_CLIENT_ID>
+OAUTH2_CLIENT_SECRET=<new_OAUTH2_CLIENT_SECRET>

# GeoNode APIs
API_LOCKDOWN=False
TASTYPIE_APIKEY=

# #####

```

[Optional] Update your SSH Certificates

In production deployment mode, GeoNode uses by default *Let's Encrypt* certificates

You may want to provide your own certificates to GeoNode

```

docker exec -it nginx4my_geonode_geonode sh -c 'mkdir /geonode-certificates/my_geonode
→ '

wget --no-check-certificate 'http://<url_to_your_chain.crt>' \
-O chain.crt

wget --no-check-certificate 'http://<url_to_your_key.key>' \

```

(continues on next page)

(continuação da página anterior)

```

-O my_geonode.key

docker cp chain.crt nginx4my_geonode_geonode:/geonode-certificates/my_geonode

docker cp my_geonode.key nginx4my_geonode_geonode:/geonode-certificates/my_geonode

docker-compose exec geonode sh
apk add vim

vim nginx.https.enabled.conf

```

```

-ssl_certificate      /certificate_symlink/fullchain.pem;
-ssl_certificate_key  /certificate_symlink/privkey.pem;
+ssl_certificate      /geonode-certificates/my_geonode/chain.crt;
+ssl_certificate_key  /geonode-certificates/my_geonode/my_geonode.key;

```

```

nginx -s reload
exit

```

Restart the GeoNode and NGINX containers

Whenever you change something on `.env` file, you will need to rebuild the container

Aviso: Be careful! The following command drops any change you might have done manually inside the containers, except for the static volumes.

```

docker-compose up -d django
docker-compose restart geonode

```

1.11.6 Further Production Enhancements

GeoServer Production Settings

JVM Settings: Memory And GeoServer Options

The `.env` file provides a way to customize GeoServer JVM Options.

The variable `GEOSERVER_JAVA_OPTS` allows you to tune-up the GeoServer container and to enable specific GeoServer options.

```

GEOSERVER_JAVA_OPTS=
-Djava.awt.headless=true -Xms2G -Xmx4G -XX:PerfDataSamplingInterval=500
-XX:SoftRefLRUPolicyMSPerMB=36000 -XX:-UseGCTimeLimit -XX:+UseConcMarkSweepGC
-XX:+UseParNewGC -XX:ParallelGCThreads=4 -Dfile.encoding=UTF8 -Djavax.servlet.
↪request.encoding=UTF-8
-Djavax.servlet.response.encoding=UTF-8 -Duser.timezone=GMT
-Dorg.geotools.shapefile.datetime=false -DGEOSERVER_CSRF_DISABLED=true -DPRINT_
↪BASE_URL=http://geoserver:8080/geoserver/pdf

```



```
-Djava.awt.headless (true)
```

Work with graphics-based applications in Java without an actual display, keyboard, or mouse. A typical use case of UI components running in a headless environment could be an image converter app. Though it needs graphics data for image processing, a display is not really necessary. The app could be run on a server and converted files saved or sent over the network to another machine for display.

```
-Xms2G -Xmx4G
```

This means that your JVM will be started with Xms amount of memory and will be able to use a maximum of Xmx amount of memory. Above will start a JVM like with 2 GB of memory and will allow the process to use up to 4 GB of memory. You need to adjust this value depending on your available RAM.

```
-DGEOSERVER_CSRF_DISABLED (True)
```

The GeoServer web admin employs a CSRF (Cross-Site Request Forgery) protection filter that will block any form submissions that didn't appear to originate from GeoServer. This can sometimes cause problems for certain proxy configurations. You can disable the CSRF filter by setting the GEOSERVER_CSRF_DISABLED property to true. <https://docs.geoserver.org/stable/en/user/security/webadmin/csrf.html>

Whenever you need to change one or more of the JVM options, you will need to restart the GeoServer Docker container.

```
# Hard restart of the container: the only way to update the .env variables
docker-compose up -d geoserver
```

This command will **preserve** all the GeoServer configuration and data, since the GEOSERVER_DATA_DIR is stored on a Docker static volume.

Nevertheless, any change you have made manually to the container, e.g. added a new plugin to GeoServer or updated some JARs into the WEB-INF/lib library folder, will be lost.

You will need to add the JARs again and restart GeoServer *softly*

```
# Soft restart of the container: the .env variables won't be updated
docker-compose restart geoserver
```

Global And Services Settings

- Check the GeoServer Memory usage and status; ensure the GEOSERVER_DATA_DIR path points to the static volume

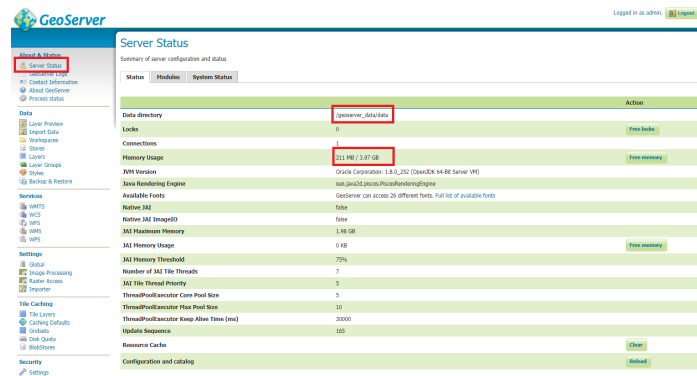


Fig. 248: GeoServer Status

- GeoServer *Global Settings*; make sure the Proxy Base Url points to the public URL and the LOGGING levels are set to *Production Mode*

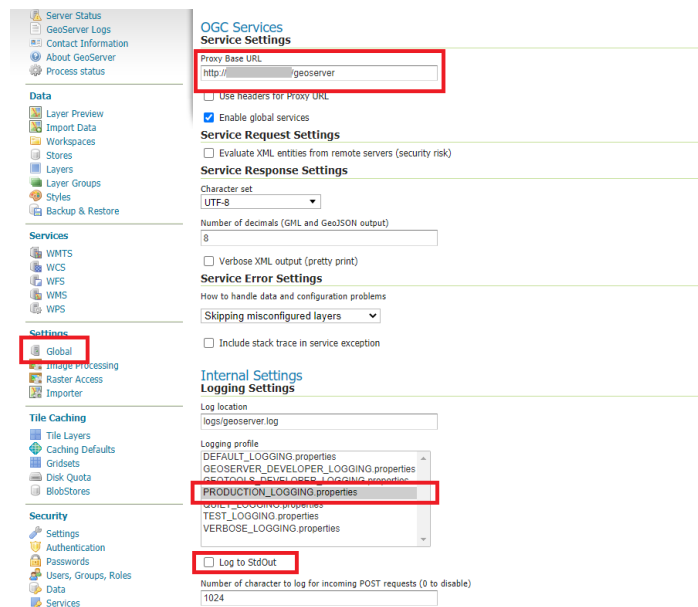


Fig. 249: Global Settings

- GeoServer *Image Processing Settings*; unless you are using some specific renderer or GeoServer plugin, use the following recommended options

Nota: Further details at https://docs.geoserver.org/stable/en/user/configuration/image_processing/index.html#image-processing

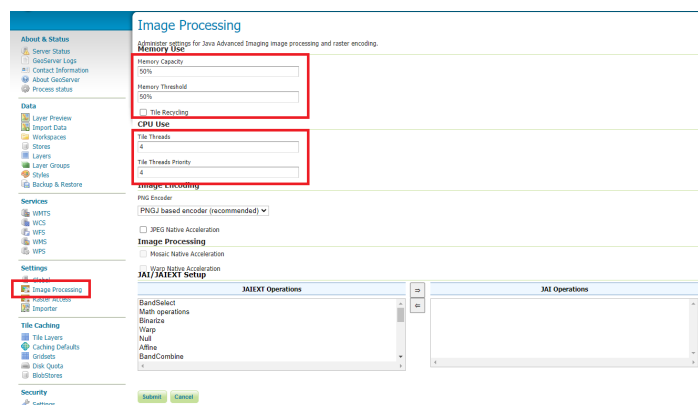


Fig. 250: Image Processing Settings

- Tune up *GeoServer Services Configuration*; WCS, WFS, WMS and WPS;
 - **WCS:** Update the limits accordingly to your needs. Do not use very high values, this will set GeoServer prone to DoS Attacks.
 - **WMS:** Specify here the SRS List you are going to use. Empty means all the ones supported by GeoServer, but be carefull since the GetCapabilities output will become huge.

Resource Consumption Limits

Maximum amount of data read (KB, 0 for no limit)

Maximum amount of data generated (KB, 0 for no limit)

Max number of dimension values

Fig. 251: WCS Resource Consumption Limits

Limited SRS list

☒ Output bounding box for every supported CRS

Fig. 252: WMS Supported SRS List

- **WMS: Raster Rendering Options** allows you to tune up the WMS output for better performance or quality. Best Performance: Nearest Neighbour - Best Quality: Bicubic

Aviso: Raster Images should be always optimized before ingested into GeoNode. The general recommendation is to **never** upload a non-processed GeoTIFF image to GeoNode.

Further details at:

- <https://geoserver.geo-solutions.it/edu/en/enterprise/raster.html>
- https://geoserver.geo-solutions.it/edu/en/raster_data/advanced_gdal/index.html

Raster Rendering Options

Default Interpolation

Nearest neighbor ▼
Nearest neighbor
Bilinear
Bicubic

Fig. 253: WMS Raster Rendering Options

- **WMS:** Update the limits accordingly to your needs. Do not use very high values, this will set GeoServer prone to DoS Attacks.

Resource consumption limits

Max rendering memory (KB)

65536

Max rendering time (s)

60

Max rendering errors (count)

1000

Max number of dimension values

100

Fig. 254: WMS Resource Consumption Limits

GeoWebCache DiskQuota On Postgis

By default GeoWebCache DiskQuota is disabled. That means that the layers cache might potentially grow up indefinitely.

GeoWebCache DiskQuota should be always enabled on a production system. In the case it is enabled, this **must** be configured to make use of a DB engine like Postgis to store its indexes.

- First of all ensure *Tile Caching* is enabled on all available layers

Nota: GeoNode typically does this automatically for you. It is worth to double check anyway.

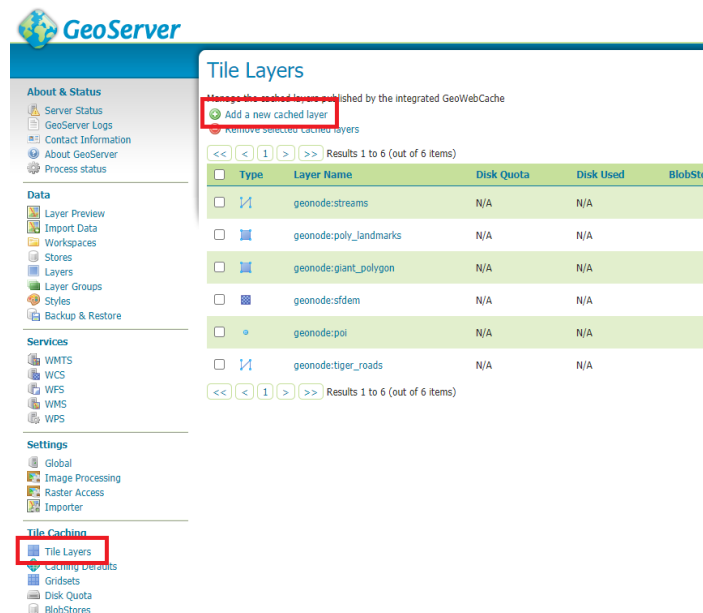


Fig. 255: Tile Caching: Tiled Layers

- Configure *Disk Quota* by providing the connection string to the DB Docker Container as specified in the *.env* file

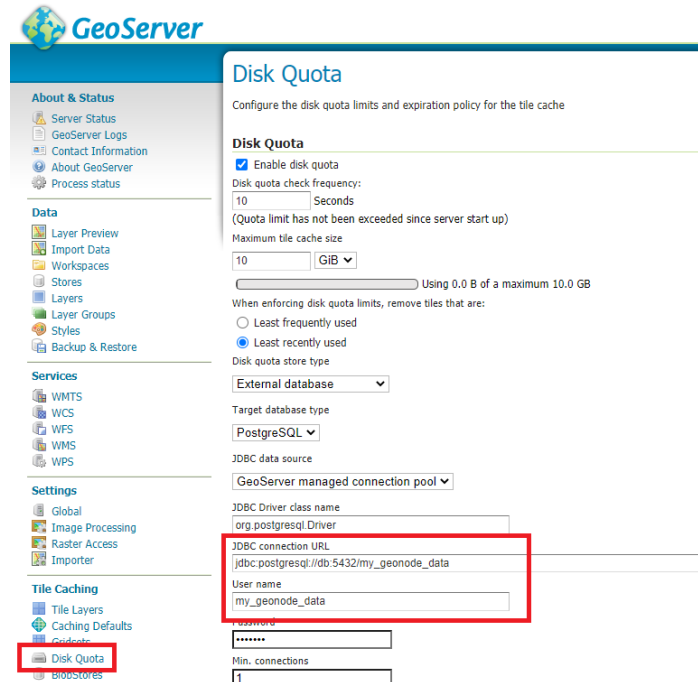


Fig. 256: Tile Caching: Disk Quota Configuration

GeoFence Security Rules On Postgis

By default GeoFence stores the security rules on an *H2* db.

On a production system, this is not really recommended. You will need to update the GeoServer Docker container in order to enable GeoFence storing the rules into the DB Docker Container instead.

In order to do that, follow the procedure below:

```
# Enter the GeoServer Docker Container
docker-compose exec geoserver bash

# Install a suitable editor
apt update
apt install nano

# Edit the GeoFence DataStore .properties file
nano /geoserver_data/data/geofence/geofence-datasource-ovr.properties
```

Nota: Make sure to provide the same connection parameters specified in the *.env* file

```
geofenceVendorAdapter.databasePlatform=org.hibernate.spatial.postgis.PostgisDialect
geofenceDataSource.driverClassName=org.postgresql.Driver
geofenceDataSource.url=jdbc:postgresql://db:5432/my_geonode_data
geofenceDataSource.username=my_geonode_data
geofenceDataSource.password=*****
geofenceEntityManagerFactory.jpaPropertyMap[hibernate.default_schema]=public
```

```
# Update the GeoServer WEB-INF/lib JARs accordingly
wget --no-check-certificate "https://www.dropbox.com/s/psolxleimaft0t7/postgis-jdbc-1.
↪3.3.jar?dl=1" -O postgis-jdbc-1.3.3.jar && \
wget --no-check-certificate "https://www.dropbox.com/s/ilowulvd27j2cs1/hibernate-
↪spatial-postgis-1.1.3.2.jar?dl=1" -O hibernate-spatial-postgis-1.1.3.2.jar && \
rm /usr/local/tomcat/webapps/geoserver/WEB-INF/lib/hibernate-spatial-h2-geodb-1.1.3.1.
↪jar && \
mv hibernate-spatial-postgis-1.1.3.2.jar /usr/local/tomcat/webapps/geoserver/WEB-INF/
↪lib/ && \
mv postgis-jdbc-1.3.3.jar /usr/local/tomcat/webapps/geoserver/WEB-INF/lib/
```

The container is ready to be restarted now.

Aviso: Remember to do a **soft restart** otherwise the WEB-INF/lib JARs will be reset to the original state

```
# Exit the GeoServer container
exit

# Soft Restart GeoServer Docker Container
docker-compose restart geoserver
```

IMPORTANT: The first time you perform this procedure, GeoFence won't be able to retrieve the old security rules anymore.

You will need to *Fixup GeoNode Layers Permissions* in order to regenerate the security rules.

Fixup GeoNode Layers Permissions

The list of the GeoFence Security Rules is available from the *GeoFence Data Rules* section.

Always double check the list is accessible and the data rules are there. If empty, no layer will be accessible by standard users other than admin.

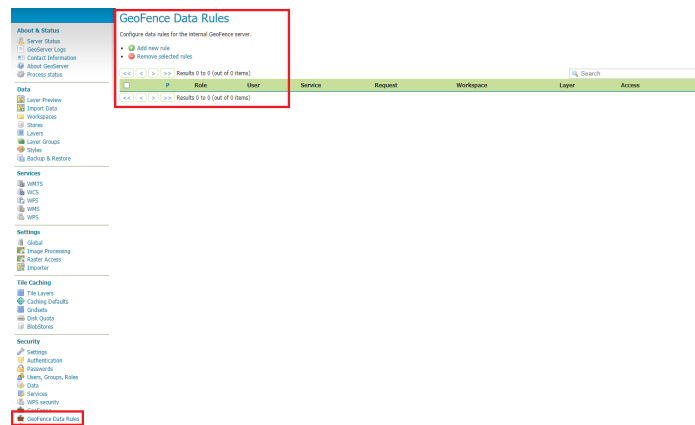


Fig. 257: *GeoFence Data Rules*

In order to re-sync the GeoFence security rules, follow the procedure below:

```
# Enter the GeoNode Docker Container
docker-compose exec django bash
```

(continues on next page)

(continuação da página anterior)

```
# Run the `sync_geonode_layers` management command
./manage.sh sync_geonode_layers --updatepermissions
```

Regenerate GeoNode Layers Thumbnails

The following procedure allows you to *batch* regenerate all Layers Thumbnails:

```
# Enter the GeoNode Docker Container
docker-compose exec django bash

# Run the `sync_geonode_layers` management command
./manage.sh sync_geonode_layers --updatethumbnails
```

Fixup GeoNode Layers Metadata And Download Links

The following procedure allows you to fix-up broken or incorrect Metadata Links:

```
# Enter the GeoNode Docker Container
docker-compose exec django bash

# Run the `set_all_layers_metadata` management command
./manage.sh set_all_layers_metadata -d
```

It is also possible to *force* purging the links before regenerating:

```
# Enter the GeoNode Docker Container
docker-compose exec django bash

# Run the `set_all_layers_metadata` management command
./manage.sh set_all_layers_metadata -d --prune
```

Migrate GeoNode To A New Hostname

In the case you will need to move your instance to another domain, as an example from `https://my_geonode.geonode.org/` to `https://prod_geonode.geonode.org/`, follow the procedure below:

- Update the `.env` file by specifying the new name accordingly.
- Restart the GeoNode Docker Container.

```
docker-compose up -d geonode
```

- Run the following management commands from inside the GeoNode Docker Container.

```
# Enter the GeoNode Docker Container
docker-compose exec django bash

# Run the `migrate_baseurl` management command
./manage.sh migrate_baseurl --source-address=my_geonode.geonode.org --
↳target-address=prod_geonode.geonode.org
```

(continues on next page)

(continuação da página anterior)

```
# Run the `set_all_layers_metadata` management command
./manage.sh set_all_layers_metadata -d
```

Add Huge Or DB Datasets To Your Instance

Uploading huge datasets, or DB tables, to GeoNode from the *Web Upload Interface* is not really possible sometimes.

The suggested procedure in such cases is the following one:

- Add the dataset to *GeoServer* first directly.

You must upload the data into the GeoServer Docker Container Static Volume first and then adding manually the layer through the *GeoServer Admin GUI*.

- Once the dataset is correctly configured on GeoServer, run the following management command from inside the GeoNode Docker Container

```
# Enter the GeoNode Docker Container
docker-compose exec django bash

# Run the `updatelayers` management command
./manage.sh updatelayers -w <workspace_name> -f <layer_name>
```

Update GeoNode Core To The Latest Commit

In the case you will need to update the GeoNode Core codebase to a specific version or commit, please follow the steps below:

```
# Enter the GeoNode Docker Container
docker-compose exec django bash

# Update GeoNode
cd /usr/src/geonode/
git fetch --all --prune
git checkout <commit or branch>

# Update the pip dependencies
pip install -r requirements.txt --upgrade --no-cache
pip install -e . --upgrade

# Synchronize the GeoNode Project
cd /usr/src/my_geonode/
./manage.sh makemigrations
./manage.sh migrate
./manage.sh collectstatic

# Refresh UWSGI Daemons
touch /usr/src/my_geonode/my_geonode/wsgi.py

# Follow the logs and make sure non errors occur
tail -F -n 30 /var/log/geonode.log
```


1.12 GeoNode Advanced Installation

1.12.1 GeoNode Core

Overview

The following steps will guide you to a fresh setup of GeoNode.

All guides will first install and configure the system to run it in `DEBUG` mode (also known as `DEVELOPMENT` mode) and then by configuring an HTTPD server to serve GeoNode through the standard HTTP (80) port.

Aviso: Those guides **are not** meant to be used on a production system. There will be dedicated chapters that will show you some *hints* to optimize GeoNode for a production-ready machine. In any case, we strongly suggest to task an experienced *DevOp* or *System Administrator* before exposing your server to the `WEB`.

Ubuntu 20.04LTS

This part of the documentation describes the complete setup process for GeoNode on an Ubuntu 20.04LTS **64-bit** clean environment (Desktop or Server).

All examples use shell commands that you must enter on a local terminal or a remote shell.

- If you have a graphical desktop environment you can open the terminal application after login;
- if you are working on a remote server the provider or sysadmin should have given you access through an ssh client.

1. Install the dependencies

In this section, we are going to install all the basic packages and tools needed for a complete GeoNode installation.

Aviso: To follow this guide, a basic knowledge about Ubuntu Server configuration and working with a shell is required.

Nota: This guide uses `vim` as the editor; feel free to use `nano`, `gedit` or others.

Upgrade system packages

Check that your system is already up-to-date with the repository running the following commands:

```
sudo add-apt-repository ppa:ubuntugis/ubuntugis-unstable
sudo apt update -y; sudo apt upgrade -y;
```

Packages Installation

Nota: You don't need to install the **system packages** if you want to run the project using Docker

We will use **example.org** as fictitious Domain Name.

First, we are going to install all the **system packages** needed for the GeoNode setup. Login to the target machine and execute the following commands:

```
# Install packages from GeoNode core
sudo apt install -y build-essential gdal-bin \
    python3.8-dev python3.8-venv virtualenvwrapper \
    libxml2 libxml2-dev gettext \
    libxslt1-dev libjpeg-dev libpng-dev libpq-dev libgdal-dev \
    software-properties-common build-essential \
    git unzip gcc zlib1g-dev libgeos-dev libproj-dev \
    sqlite3 spatialite-bin libsqlite3-mod-spatialite libsqlite3-dev

# Install Openjdk
sudo apt install openjdk-8-jdk-headless default-jdk-headless -y
sudo update-java-alternatives --jre-headless --jre --set java-1.8.0-openjdk-amd64

# Verify GDAL version
gdalinfo --version
$> GDAL 3.0.4, released 2020/01/28

# Verify Python version
python3.8 --version
$> Python 3.8.5

which python3.8
$> /usr/bin/python3.8

# Verify Java version
java -version
$> openjdk version "1.8.0_265"
$> OpenJDK Runtime Environment (build 1.8.0_265-8u265-b01-0ubuntu2~20.04-b01)
$> OpenJDK 64-Bit Server VM (build 25.265-b01, mixed mode)

# Install VIM
sudo apt install -y vim

# Cleanup the packages
sudo apt update -y; sudo apt upgrade -y; sudo apt autoremove --purge
```

Aviso: GeoNode 3.x is not compatible with Python < 3.7

2. GeoNode Installation

This is the most basic installation of GeoNode. It won't use any external server like Apache Tomcat, PostgreSQL or HTTPD.

First of all we need to prepare a new Python Virtual Environment

Since geonode needs a large number of different python libraries and packages, its recommended to use a python virtual environment to avoid conflicts on dependencies with system wide python packages and other installed software. See also documentation of [Virtualenvwrapper](#) package for more information

Nota: The GeoNode Virtual Environment must be created only the first time. You won't need to create it again everytime.

```
which python3.8 # copy the path of python executable

# Create the GeoNode Virtual Environment (first time only)
export WORKON_HOME=~/.virtualenvs
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
mkvirtualenv --python=/usr/bin/python3.8 geonode # Use the python path from above

# Alternatively you can also create the virtual env like below
mkdir -p ~/.virtualenvs
python3.8 -m venv ~/.virtualenvs/geonode
source ~/.virtualenvs/geonode/bin/activate
```

At this point your command prompt shows a (geonode) prefix, this indicates that your virtualenv is active.

Nota: The next time you need to access the Virtual Environment just run

```
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
workon geonode

# Alternatively you can also create the virtual env like below
source ~/.virtualenvs/geonode/bin/activate
```

Nota: In order to save permanently the virtualenvwrapper environment

```
vim ~/.bashrc

# Write to the bottom of the file the following lines
export WORKON_HOME=~/.virtualenvs
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
```

```
# Let's create the GeoNode core base folder and clone it
sudo mkdir -p /opt/geonode/; sudo usermod -a -G www-data $USER; sudo chown -Rf
↪$USER:www-data /opt/geonode/; sudo chmod -Rf 775 /opt/geonode/

# Clone the GeoNode source code on /opt/geonode
cd /opt; git clone https://github.com/GeoNode/geonode.git -b 3.2.x geonode
```

```
# Install the Python packages
cd /opt/geonode
pip install -r requirements.txt --upgrade --no-cache --no-cache-dir
pip install -e . --upgrade
pip install pygdal=="gdal-config --version`.*"
```

3. Postgis database Setup

Be sure you have successfully completed all the steps of the section *1. Install the dependencies*.

In this section, we are going to setup users and databases for GeoNode in PostgreSQL.

Install and Configure the PostgreSQL Database System

In this section we are going to install the PostgreSQL packages along with the PostGIS extension. Those steps must be done **only** if you don't have the DB already installed on your system.

```
# Ubuntu 20.04 (focal)
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg_
↳main" >> /etc/apt/sources.list.d/pgdg.list'
sudo wget --no-check-certificate --quiet -O - https://www.postgresql.org/media/keys/
↳ACCC4CF8.asc | sudo apt-key add -
sudo apt update -y; sudo apt install -y postgresql-13 postgresql-13-postgis-3_
↳postgresql-13-postgis-3-scripts postgresql-13 postgresql-client-13
```

We now must create two databases, geonode and geonode_data, belonging to the role geonode.

Aviso: This is our default configuration. You can use any database or role you need. The connection parameters must be correctly configured on settings, as we will see later in this section.

Databases and Permissions

First, create the geonode user. GeoNode is going to use this user to access the database

```
sudo service postgresql start
sudo -u postgres createuser -P geonode

# Use the password: geonode
```

You will be prompted asked to set a password for the user. **Enter geonode as password.**

Aviso: This is a sample password used for the sake of simplicity. This password is very **weak** and should be changed in a production environment.

Create database geonode and geonode_data with owner geonode

```
sudo -u postgres createdb -O geonode geonode
sudo -u postgres createdb -O geonode geonode_data
```

Next let's create PostGIS extensions

```

sudo -u postgres psql -d geonode -c 'CREATE EXTENSION postgis;'
sudo -u postgres psql -d geonode -c 'GRANT ALL ON geometry_columns TO PUBLIC;'
sudo -u postgres psql -d geonode -c 'GRANT ALL ON spatial_ref_sys TO PUBLIC;'
sudo -u postgres psql -d geonode -c 'GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA_
↳public TO geonode;'
sudo -u postgres psql -d geonode -c 'GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA_
↳public TO geonode;'

sudo -u postgres psql -d geonode_data -c 'CREATE EXTENSION postgis;'
sudo -u postgres psql -d geonode_data -c 'GRANT ALL ON geometry_columns TO PUBLIC;'
sudo -u postgres psql -d geonode_data -c 'GRANT ALL ON spatial_ref_sys TO PUBLIC;'
sudo -u postgres psql -d geonode_data -c 'GRANT ALL PRIVILEGES ON ALL TABLES IN_
↳SCHEMA public TO geonode;'
sudo -u postgres psql -d geonode_data -c 'GRANT ALL PRIVILEGES ON ALL SEQUENCES IN_
↳SCHEMA public TO geonode;'

```

Final step is to change user access policies for local connections in the file `pg_hba.conf`

```
sudo vim /etc/postgresql/13/main/pg_hba.conf
```

Scroll down to the bottom of the document. We want to make local connection trusted for the default user.

Make sure your configuration looks like the one below.

```

...
# DO NOT DISABLE!
# If you change this first entry you will need to make sure that the
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local    all             postgres                                trust

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local    all             all                                md5
# IPv4 local connections:
host     all             all             127.0.0.1/32            md5
# IPv6 local connections:
host     all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local    replication     all                                peer
host     replication     all             127.0.0.1/32            md5
host     replication     all             ::1/128                 md5

```

Aviso: If your PostgreSQL database resides on a **separate/remote machine**, you'll have to **allow** remote access to the databases in the `/etc/postgresql/13/main/pg_hba.conf` to the `geonode` user and tell PostgreSQL to **accept** non-local connections in your `/etc/postgresql/13/main/postgresql.conf` file

Restart PostgreSQL to make the change effective.

```
sudo service postgresql restart
```

PostgreSQL is now ready. To test the configuration, try to connect to the `geonode` database as `geonode` role.

```
psql -U postgres geonode
# This should not ask for any password

psql -U geonode geonode
# This should ask for the password geonode

# Repeat the test with geonode_data DB
psql -U postgres geonode_data
psql -U geonode geonode_data
```

4. Install GeoServer

In this section, we are going to install the Apache Tomcat 8 Servlet Java container, which will be started by default on the internal port 8080.

We will also perform several optimizations to:

1. Correctly setup the Java VM Options, like the available heap memory and the garbage collector options.
2. Externalize the GeoServer and GeoWebcache catalogs in order to allow further updates without the risk of deleting our datasets.

Nota: This is still a basic setup of those components. More details will be provided on sections of the documentation concerning the hardening of the system in a production environment. Nevertheless, you will need to tweak a bit those settings accordingly with your current system. As an instance, if your machine does not have enough memory, you will need to lower down the initial amount of available heap memory. **Warnings** and **notes** will be placed below the statements that will require your attention.

Install Apache Tomcat 9 (ref. <https://yallalabs.com/linux/ubuntu/how-to-install-apache-tomcat-9-ubuntu-20-04/>)

Aviso: Apache Tomcat 9 requires Java 8 or newer to be installed on the server. Check the steps before in order to be sure you have OpenJDK 8 correctly installed on your system.

First, it is not recommended to run Apache Tomcat as user root, so we will create a new system user which will run the Apache Tomcat server

```
sudo useradd -m -U -d /opt/tomcat -s /bin/bash tomcat
sudo usermod -a -G www-data tomcat
```

Aviso: Now, go to the official Apache Tomcat [website](#) and download the most recent version of the software to your server. But don't use Tomcat10 because there are still some errors between Geoserver and Tomcat.

```
VERSION=9.0.48; wget https://www-eu.apache.org/dist/tomcat/tomcat-9/v${VERSION}/bin/
↪apache-tomcat-${VERSION}.tar.gz
```

Once the download is complete, extract the tar file to the /opt/tomcat directory:

```
sudo tar -xf apache-tomcat-${VERSION}.tar.gz -C /opt/tomcat/; rm apache-tomcat-${VERSION}.tar.gz
```

Apache Tomcat is updated regularly. So, to have more control over versions and updates, we'll create a symbolic link as below:

```
sudo ln -s /opt/tomcat/apache-tomcat-${VERSION} /opt/tomcat/latest
```

Now, let's change the ownership of all Apache Tomcat files as below:

```
sudo chown -R tomcat:www-data /opt/tomcat/
```

Make the shell scripts inside the bin directory executable:

```
sudo sh -c 'chmod +x /opt/tomcat/latest/bin/*.sh'
```

Create the a systemd file with the following content:

```
# Check the correct JAVA_HOME location
JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
echo $JAVA_HOME
$> /usr/lib/jvm/java-8-openjdk-amd64/jre/

# Let's create a symbolic link to the JRE
sudo ln -s /usr/lib/jvm/java-8-openjdk-amd64/jre/ /usr/lib/jvm/jre

# Let's create the tomcat service
sudo vim /etc/init.d/tomcat9
```

```
#!/bin/bash

### BEGIN INIT INFO
# Provides:          tomcat9
# Required-Start:    $local_fs $remote_fs $network $time
# Required-Stop:     $local_fs $remote_fs $network $time
# Should-Start:      $syslog
# Should-Stop:       $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Apache Tomcat init script
### END INIT INFO

#Location of JAVA_HOME (bin files)
export JAVA_HOME=/usr/lib/jvm/jre
export JAVA_OPTS=-Djava.security.egd=file:///dev/urandom

#Add Java binary files to PATH
export PATH=$JAVA_HOME/bin:$PATH

#CATALINA_HOME is the location of the bin files of Tomcat
export CATALINA_HOME=/opt/tomcat/latest

#CATALINA_BASE is the location of the configuration files of this instance of Tomcat
export CATALINA_BASE=/opt/tomcat/latest
export CATALINA_PID=/opt/tomcat/latest/temp/tomcat.pid
```

(continues on next page)

```

#TOMCAT_USER is the default user of tomcat
export TOMCAT_USER=tomcat

#TOMCAT_USAGE is the message if this script is called without any options
TOMCAT_USAGE="Usage: $0 {\e[00;32mstart\e[00m|\e[00;31mstop\e[00m|\e[00;31mkill\
↪\e[00m|\e[00;32mstatus\e[00m|\e[00;31mrestart\e[00m}"

#SHUTDOWN_WAIT is wait time in seconds for java proccess to stop
SHUTDOWN_WAIT=20

tomcat_pid() {
    echo `ps -fe | grep $CATALINA_BASE | grep -v grep | tr -s " "|cut -d" " -f2`
}

start() {
    pid=$(tomcat_pid)
    if [ -n "$pid" ]
    then
        echo -e "\e[00;31mTomcat is already running (pid: $pid)\e[00m"
    else
        # Start tomcat
        echo -e "\e[00;32mStarting tomcat\e[00m"
        #ulimit -n 100000
        #umask 007
        #/bin/su -p -s /bin/sh $TOMCAT_USER
        if [ `user_exists $TOMCAT_USER` = "1" ]
        then
            /bin/su $TOMCAT_USER -c $CATALINA_HOME/bin/startup.sh
        else
            echo -e "\e[00;31mTomcat user $TOMCAT_USER does not exists. Starting_
↪with $(id)\e[00m"
            sh $CATALINA_HOME/bin/startup.sh
        fi
        status
    fi
    return 0
}

status(){
    pid=$(tomcat_pid)
    if [ -n "$pid" ]
    then echo -e "\e[00;32mTomcat is running with pid: $pid\e[00m"
    else
        echo -e "\e[00;31mTomcat is not running\e[00m"
        return 3
    fi
}

terminate() {
    echo -e "\e[00;31mTerminating Tomcat\e[00m"
    kill -9 $(tomcat_pid)
}

stop() {
    pid=$(tomcat_pid)
    if [ -n "$pid" ]

```

(continues on next page)

(continuação da página anterior)

```

then
    echo -e "\e[00;31mStoping Tomcat\e[00m"
    #/bin/su -p -s /bin/sh $TOMCAT_USER
    sh $CATALINA_HOME/bin/shutdown.sh

    let kwait=$SHUTDOWN_WAIT
    count=0;
    until [ `ps -p $pid | grep -c $pid` = '0' ] || [ $count -gt $kwait ]
    do
        echo -n -e "\n\e[00;31mwaiting for processes to exit\e[00m";
        sleep 1
        let count=$count+1;
    done

    if [ $count -gt $kwait ]; then
        echo -n -e "\n\e[00;31mkilling processes didn't stop after $SHUTDOWN_WAIT_
↪seconds\e[00m"
        terminate
    fi
else
    echo -e "\e[00;31mTomcat is not running\e[00m"
fi

return 0
}

user_exists(){
    if id -u $1 >/dev/null 2>&1; then
        echo "1"
    else
        echo "0"
    fi
}

case $1 in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        stop
        start
        ;;
    status)
        status
        exit $?
        ;;
    kill)
        terminate
        ;;
    *)
        echo -e $TOMCAT_USAGE
        ;;
esac
exit 0

```

Now you can start the Apache Tomcat 9 server and enable it to start on boot time using the following command:

```
sudo chmod +x /etc/init.d/tomcat9
sudo /etc/init.d/tomcat9 start
```

For verification, type the following ss command, which will show you the 8080 open port number, the default open port reserved for Apache Tomcat Server.

```
ss -ltn
```

If your server is protected by a firewall and you want to access Tomcat from the outside of your local network, you need to open port 8080.

Use the following command to open the necessary port:

```
sudo ufw allow 8080/tcp
```

Aviso: Generally, when running Tomcat in a production environment, you should use a load balancer or reverse proxy.

It's a best practice to allow access to port 8080 only from your internal network.

We will use NGINX in order to provide Apache Tomcat through the standard HTTP port.

Nota: Alternatively you can define the Tomcat Service as follow, in case you would like to use systemctl

```
sudo vim /usr/lib/systemd/system/tomcat9.service
```

```
[Unit]
Description=Apache Tomcat Server
After=syslog.target network.target

[Service]
Type=forking
User=tomcat
Group=tomcat

Environment=JAVA_HOME=/usr/lib/jvm/jre
Environment=JAVA_OPTS=-Djava.security.egd=file:///dev/urandom
Environment=CATALINA_PID=/opt/tomcat/latest/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat/latest
Environment=CATALINA_BASE=/opt/tomcat/latest

ExecStart=/opt/tomcat/latest/bin/startup.sh
ExecStop=/opt/tomcat/latest/bin/shutdown.sh

RestartSec=30
Restart=always

[Install]
WantedBy=multi-user.target
```

```
sudo systemctl daemon-reload
sudo systemctl enable tomcat9.service
sudo systemctl start tomcat9.service
```

Install GeoServer on Tomcat9

Let's externalize the GEOSERVER_DATA_DIR and logs

```
# Create the target folders
sudo mkdir -p /opt/data
sudo chown -Rf $USER:www-data /opt/data
sudo chmod -Rf 775 /opt/data
sudo mkdir -p /opt/data/logs
sudo chown -Rf $USER:www-data /opt/data/logs
sudo chmod -Rf 775 /opt/data/logs

# Download and extract the default GEOSERVER_DATA_DIR
sudo wget --no-check-certificate "https://www.dropbox.com/s/cd20is9ddjz7ti5/data-2.18.3.zip?dl=1" -O data-2.18.3.zip
sudo unzip data-2.18.3.zip -d /opt/data/

sudo mv /opt/data/data/ /opt/data/geoserver_data
sudo chown -Rf tomcat:www-data /opt/data/geoserver_data
sudo chmod -Rf 775 /opt/data/geoserver_data

sudo mkdir -p /opt/data/geoserver_logs
sudo chown -Rf tomcat:www-data /opt/data/geoserver_logs
sudo chmod -Rf 775 /opt/data/geoserver_logs

sudo mkdir -p /opt/data/gwc_cache_dir
sudo chown -Rf tomcat:www-data /opt/data/gwc_cache_dir
sudo chmod -Rf 775 /opt/data/gwc_cache_dir

# Download and install GeoServer
sudo wget --no-check-certificate "https://www.dropbox.com/s/cmdrdzdeloq67pre/geoserver-2.18.3.war?dl=0" -O geoserver-2.18.3.war
sudo mv geoserver-2.18.3.war /opt/tomcat/latest/webapps/geoserver.war
```

Let's now configure the JAVA_OPTS, i.e. the parameters to run the Servlet Container, like heap memory, garbage collector and so on.

```
sudo sed -i -e 's/xom-.*\.jar/xom-.*\.jar,bcprov.*\.jar/g' /opt/tomcat/latest/conf/catalina.properties

export JAVA_HOME=$(readlink -f /usr/bin/java | sed "s:bin/java::")
echo 'JAVA_HOME=$JAVA_HOME' | sudo tee --append /opt/tomcat/latest/bin/setenv.sh
sudo sed -i -e "s/JAVA_OPTS=#JAVA_OPTS=g" /opt/tomcat/latest/bin/setenv.sh

echo 'GEOSERVER_DATA_DIR="/opt/data/geoserver_data"' | sudo tee --append /opt/tomcat/latest/bin/setenv.sh
echo 'GEOSERVER_LOG_LOCATION="/opt/data/geoserver_logs/geoserver.log"' | sudo tee --append /opt/tomcat/latest/bin/setenv.sh
echo 'GEOWEBCACHE_CACHE_DIR="/opt/data/gwc_cache_dir"' | sudo tee --append /opt/tomcat/latest/bin/setenv.sh
echo 'GEOFENCE_DIR="$GEOSERVER_DATA_DIR/geofence"' | sudo tee --append /opt/tomcat/latest/bin/setenv.sh
echo 'TIMEZONE="UTC"' | sudo tee --append /opt/tomcat/latest/bin/setenv.sh

echo 'JAVA_OPTS="-server -Djava.awt.headless=true -Dorg.geotools.shapefile.datetimestamp=false -XX:+UseParallelGC -XX:ParallelGCThreads=4 -Dfile.encoding=UTF-8 -Duser.timezone=$TIMEZONE -Xms512m -Xmx4096m -Djavax.servlet.request.encoding=UTF-8 -Djavax.servlet.response.encoding=UTF-8 -DGEOSERVER_CSRF_DISABLED=true -DPRINT_BASE_URL=http://localhost:8080/geoserver/pdf -DGEOSERVER_DATA_DIR=$GEOSERVER_DATA_DIR -Dgeofence.dir=$GEOFENCE_DIR -DGEOSERVER_LOG_LOCATION=$GEOSERVER_LOG_LOCATION -DGEOWEBCACHE_CACHE_DIR=$GEOWEBCACHE_CACHE_DIR"' | sudo tee --append /opt/tomcat/latest/bin/setenv.sh
```

(continues on next page)

Nota: After the execution of the above statements, you should be able to see the new options written at the bottom of the file `/opt/tomcat/latest/bin/setenv.sh`.

```
...
# If you run Tomcat on port numbers that are all higher than 1023, then you
# do not need authbind. It is used for binding Tomcat to lower port numbers.
# (yes/no, default: no)
#AUTHBIND=no
JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre/
GEOSERVER_DATA_DIR="/opt/data/geoserver_data"
GEOSERVER_LOG_LOCATION="/opt/data/geoserver_logs/geoserver.log"
GEOWEBCACHE_CACHE_DIR="/opt/data/gwc_cache_dir"
GEOFENCE_DIR="$GEOSERVER_DATA_DIR/geofence"
TIMEZONE="UTC"
JAVA_OPTS="-server -Djava.awt.headless=true -Dorg.geotools.shapefile.datetime=false -
↪XX:+UseParallelGC -XX:ParallelGCThreads=4 -Dfile.encoding=UTF8 -Duser.timezone=
↪$TIMEZONE -Xms512m -Xmx4096m -Djavax.servlet.request.encoding=UTF-8 -Djavax.servlet.
↪response.encoding=UTF-8 -DGEOSERVER_CSRF_DISABLED=true -DPRINT_BASE_URL=http://
↪localhost:8080/geoserver/pdf -DGEOSERVER_DATA_DIR=$GEOSERVER_DATA_DIR -Dgeofence.
↪dir=$GEOFENCE_DIR -DGEOSERVER_LOG_LOCATION=$GEOSERVER_LOG_LOCATION -DGEOWEBCACHE_
↪CACHE_DIR=$GEOWEBCACHE_CACHE_DIR"
```

Those options could be updated or changed manually at any time, accordingly to your needs.

Aviso: The default options we are going to add to the Servlet Container, assume you can reserve at least 4GB of RAM to GeoServer (see the option `-Xmx4096m`). You must be sure your machine has enough memory to run both GeoServer and GeoNode, which in this case means at least 4GB for GeoServer plus at least 2GB for GeoNode. A total of at least 6GB of RAM available on your machine. If you don't have enough RAM available, you can lower down the values `-Xms512m -Xmx4096m`. Consider that with less RAM available, the performances of your services will be highly impacted.

In order to make the changes effective, you'll need to restart the Servlet Container.

```
# Restart the server
sudo /etc/init.d/tomcat9 restart

# Follow the startup logs
sudo tail -F -n 300 /opt/data/geoserver_logs/geoserver.log
```

If you can see on the logs something similar to this, without errors

```
...
2019-05-31 10:06:34,190 INFO [geoserver.wps] - Found 5 bindable processes in_
↪GeoServer specific processes
2019-05-31 10:06:34,281 INFO [geoserver.wps] - Found 89 bindable processes in_
↪Deprecated processes
2019-05-31 10:06:34,298 INFO [geoserver.wps] - Found 31 bindable processes in Vector_
↪processes
2019-05-31 10:06:34,307 INFO [geoserver.wps] - Found 48 bindable processes in_
↪Geometry processes
```

(continues on next page)

(continuação da página anterior)

```

2019-05-31 10:06:34,307 INFO [geoserver.wps] - Found 1 bindable processes in
↳PolygonLabelProcess
2019-05-31 10:06:34,311 INFO [geoserver.wps] - Blacklisting process
↳ras:ConvolveCoverage as the input kernel of type class javax.media.jai.KernelJAI
↳cannot be handled
2019-05-31 10:06:34,319 INFO [geoserver.wps] - Blacklisting process
↳ras:RasterZonalStatistics2 as the input zones of type class java.lang.Object cannot
↳be handled
2019-05-31 10:06:34,320 INFO [geoserver.wps] - Blacklisting process
↳ras:RasterZonalStatistics2 as the input nodata of type class it.geosolutions.jaiext.
↳range.Range cannot be handled
2019-05-31 10:06:34,320 INFO [geoserver.wps] - Blacklisting process
↳ras:RasterZonalStatistics2 as the input rangeData of type class java.lang.Object
↳cannot be handled
2019-05-31 10:06:34,320 INFO [geoserver.wps] - Blacklisting process
↳ras:RasterZonalStatistics2 as the output zonal statistics of type interface java.
↳util.List cannot be handled
2019-05-31 10:06:34,321 INFO [geoserver.wps] - Found 18 bindable processes in Raster
↳processes
2019-05-31 10:06:34,917 INFO [ows.OWSHandlerMapping] - Mapped URL path [/TestWfsPost]
↳onto handler 'wfsTestServlet'
2019-05-31 10:06:34,918 INFO [ows.OWSHandlerMapping] - Mapped URL path [/wfs/*] onto
↳handler 'dispatcher'
2019-05-31 10:06:34,918 INFO [ows.OWSHandlerMapping] - Mapped URL path [/wfs] onto
↳handler 'dispatcher'
2019-05-31 10:06:42,237 INFO [geoserver.security] - Start reloading user/groups for
↳service named default
2019-05-31 10:06:42,241 INFO [geoserver.security] - Reloading user/groups successful
↳for service named default
2019-05-31 10:06:42,357 WARN [auth.GeoFenceAuthenticationProvider] - INIT FROM CONFIG
2019-05-31 10:06:42,494 INFO [geoserver.security] - AuthenticationCache Initialized
↳with 1000 Max Entries, 300 seconds idle time, 600 seconds time to live and 3
↳concurrency level
2019-05-31 10:06:42,495 INFO [geoserver.security] - AuthenticationCache Eviction Task
↳created to run every 600 seconds
2019-05-31 10:06:42,506 INFO [config.GeoserverXMLResourceProvider] - Found
↳configuration file in /opt/data/gwc_cache_dir
2019-05-31 10:06:42,516 INFO [config.GeoserverXMLResourceProvider] - Found
↳configuration file in /opt/data/gwc_cache_dir
2019-05-31 10:06:42,542 INFO [config.XMLConfiguration] - Wrote configuration to /opt/
↳data/gwc_cache_dir
2019-05-31 10:06:42,547 INFO [geoserver.importer] - Enabling import store: memory

```

Your GeoServer should be up and running at

<http://localhost:8080/geoserver/>

Aviso: In case of errors or the file `geoserver.log` is not created, check the Catalina logs in order to try to understand what's happened.

```
sudo less /opt/tomcat/latest/logs/catalina.out
```

5. Web Server

Until now we have seen how to start GeoNode in `DEBUG` mode from the command line, through the `paver` utilities. This is of course not the best way to start it. Moreover you will need a dedicated HTTPD server running on port 80 if you would like to expose your server to the world.

In this section we will see:

1. How to configure NGINX HTTPD Server to host GeoNode and GeoServer. In the initial setup we will still run the services on `http://localhost`
2. Update the settings in order to link GeoNode and GeoServer to the PostgreSQL Database.
3. Update the settings in order to update GeoNode and GeoServer services running on a **public IP** or **hostname**.
4. Install and enable HTTPS secured connection through the Let 's Encrypt provider.

Install and configure NGINX

Aviso: Seems to be possible that NGINX works with Python 3.6 and not with 3.8.

```
# Install the services
sudo apt install -y nginx uwsgi uwsgi-plugin-python3
```

Serving {"geonode", "geoserver"} via NGINX

```
# Create the GeoNode UWSGI config
sudo vim /etc/uwsgi/apps-available/geonode.ini
```

Aviso: !IMPORTANT!

Change the line `virtualenv = /home/<my_user>/virtualenvs/geonode` below with your current user home directory!

e.g.: If the user is `afabiani` then `virtualenv = /home/afabiani/.virtualenvs/geonode`

```
[uwsgi]
uwsgi-socket = 0.0.0.0:8000
# http-socket = 0.0.0.0:8000

gid = www-data

plugins = python3
virtualenv = /home/<my_user>/virtualenvs/geonode

env = DJANGO_SETTINGS_MODULE=geonode.settings
env = GEONODE_INSTANCE_NAME=geonode
env = GEONODE_LB_HOST_IP=
env = GEONODE_LB_PORT=
```

(continues on next page)

(continuação da página anterior)

```

# #####
# backend
# #####
env = POSTGRES_USER=postgres
env = POSTGRES_PASSWORD=postgres
env = GEONODE_DATABASE=geonode
env = GEONODE_DATABASE_PASSWORD=geonode
env = GEONODE_GEODATABASE=geonode_data
env = GEONODE_GEODATABASE_PASSWORD=geonode
env = GEONODE_DATABASE_SCHEMA=public
env = GEONODE_GEODATABASE_SCHEMA=public
env = DATABASE_HOST=localhost
env = DATABASE_PORT=5432
env = DATABASE_URL=postgis://geonode:geonode@localhost:5432/geonode
env = GEODATABASE_URL=postgis://geonode:geonode@localhost:5432/geonode_data
env = GEONODE_DB_CONN_MAX_AGE=0
env = GEONODE_DB_CONN_TOUT=5
env = DEFAULT_BACKEND_DATASTORE=datastore
env = BROKER_URL=amqp://admin:admin@localhost:5672//
env = ASYNC_SIGNALS=False

env = SITEURL=http://localhost/

env = ALLOWED_HOSTS="['*']"

# Data Uploader
env = DEFAULT_BACKEND_UPLOADER=geonode.importer
env = TIME_ENABLED=True
env = MOSAIC_ENABLED=False
env = HAYSTACK_SEARCH=False
env = HAYSTACK_ENGINE_URL=http://elasticsearch:9200/
env = HAYSTACK_ENGINE_INDEX_NAME=haystack
env = HAYSTACK_SEARCH_RESULTS_PER_PAGE=200

# #####
# nginx
# HTTPD Server
# #####
env = GEONODE_LB_HOST_IP=localhost
env = GEONODE_LB_PORT=80

# IP or domain name and port where the server can be reached on HTTPS (leave HOST_
↳ empty if you want to use HTTP only)
# port where the server can be reached on HTTPS
env = HTTP_HOST=localhost
env = HTTPS_HOST=

env = HTTP_PORT=8000
env = HTTPS_PORT=443

# #####
# geoserver
# #####
env = GEOSERVER_WEB_UI_LOCATION=http://localhost/geoserver/
env = GEOSERVER_PUBLIC_LOCATION=http://localhost/geoserver/
env = GEOSERVER_LOCATION=http://localhost:8080/geoserver/
env = GEOSERVER_ADMIN_USER=admin

```

(continues on next page)

(continuação da página anterior)

```

env = GEOSERVER_ADMIN_PASSWORD=geoserver

env = OGC_REQUEST_TIMEOUT=5
env = OGC_REQUEST_MAX_RETRIES=1
env = OGC_REQUEST_BACKOFF_FACTOR=0.3
env = OGC_REQUEST_POOL_MAXSIZE=10
env = OGC_REQUEST_POOL_CONNECTIONS=10

# Java Options & Memory
env = ENABLE_JSONP=true
env = outFormat=text/javascript
env = GEOSERVER_JAVA_OPTS="-Djava.awt.headless=true -Xms2G -Xmx4G -
↳XX:+UnlockDiagnosticVMOptions -XX:+LogVMOutput -XX:LogFile=/var/log/jvm.log -
↳XX:PerfDataSamplingInterval=500 -XX:SoftRefLRUPolicyMSPerMB=36000 -XX:-
↳UseGCOverheadLimit -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:ParallelGCThreads=4
↳-Dfile.encoding=UTF8 -Djavax.servlet.request.encoding=UTF-8 -Djavax.servlet.
↳response.encoding=UTF-8 -Duser.timezone=GMT -Dorg.geotools.shapefile.datetime=false
↳-DGEOSERVER_CSRF_DISABLED=true -DPRINT_BASE_URL=http://geoserver:8080/geoserver/pdf
↳-DALLOW_ENV_PARAMETRIZATION=true -Xbootclasspath/a:/usr/local/tomcat/webapps/
↳geoserver/WEB-INF/lib/marlin-0.9.3-Unsafe.jar -Dsun.java2d.renderer=org.marlin.
↳pisces.MarlinRenderingEngine"

# #####
# Security
# #####
# Admin Settings
env = ADMIN_USERNAME=admin
env = ADMIN_PASSWORD=admin
env = ADMIN_EMAIL=admin@localhost

# EMAIL Notifications
env = EMAIL_ENABLE=False
env = DJANGO_EMAIL_BACKEND=django.core.mail.backends.smtp.EmailBackend
env = DJANGO_EMAIL_HOST=localhost
env = DJANGO_EMAIL_PORT=25
env = DJANGO_EMAIL_HOST_USER=
env = DJANGO_EMAIL_HOST_PASSWORD=
env = DJANGO_EMAIL_USE_TLS=False
env = DJANGO_EMAIL_USE_SSL=False
env = DEFAULT_FROM_EMAIL='GeoNode <no-reply@geonode.org>'

# Session/Access Control
env = LOCKDOWN_GEONODE=False
env = CORS_ORIGIN_ALLOW_ALL=True
env = X_FRAME_OPTIONS="SAMEORIGIN"
env = SESSION_EXPIRED_CONTROL_ENABLED=True
env = DEFAULT_ANONYMOUS_VIEW_PERMISSION=True
env = DEFAULT_ANONYMOUS_DOWNLOAD_PERMISSION=True

# Users Registration
env = ACCOUNT_OPEN_SIGNUP=True
env = ACCOUNT_EMAIL_REQUIRED=True
env = ACCOUNT_APPROVAL_REQUIRED=False
env = ACCOUNT_CONFIRM_EMAIL_ON_GET=False
env = ACCOUNT_EMAIL_VERIFICATION=none
env = ACCOUNT_EMAIL_CONFIRMATION_EMAIL=False
env = ACCOUNT_EMAIL_CONFIRMATION_REQUIRED=False

```

(continues on next page)

(continuação da página anterior)

```

env = ACCOUNT_AUTHENTICATION_METHOD=username_email
env = AUTO_ASSIGN_REGISTERED_MEMBERS_TO_REGISTERED_MEMBERS_GROUP_NAME=True

# OAuth2
env = OAUTH2_API_KEY=
env = OAUTH2_CLIENT_ID=Jrchz2oPY3akmzndmgUTYrs9gczlgoV20YPSvqaV
env = OAUTH2_CLIENT_
↪SECRET=rCnp5txobUo83EpQEblM8fVj3QT5zb5qRfxNsuPzCqZaiRyIoxM4jdgMiZKfFePBHYXCLd7B8NlkfDBY9HKeIQPcy5Cq

# GeoNode APIs
env = API_LOCKDOWN=False
env = TASTYPIE_APIKEY=

# #####
# Production and
# Monitoring
# #####
env = DEBUG=False

SECRET_KEY='myv-y4#7j-d*p-__@j#*3z@!y24fz8%^z2v6atuy4bo9vqr1_a'

env = CACHE_BUSTING_STATIC_ENABLED=False
env = CACHE_BUSTING_MEDIA_ENABLED=False

env = MEMCACHED_ENABLED=False
env = MEMCACHED_BACKEND=django.core.cache.backends.memcached.MemcachedCache
env = MEMCACHED_LOCATION=127.0.0.1:11211
env = MEMCACHED_LOCK_EXPIRE=3600
env = MEMCACHED_LOCK_TIMEOUT=10

env = MAX_DOCUMENT_SIZE=2
env = CLIENT_RESULTS_LIMIT=5
env = API_LIMIT_PER_PAGE=1000

# GIS Client
env = GEONODE_CLIENT_LAYER_PREVIEW_LIBRARY=mapstore
env = MAPBOX_ACCESS_TOKEN=
env = BING_API_KEY=
env = GOOGLE_API_KEY=

# Monitoring
env = MONITORING_ENABLED=True
env = MONITORING_DATA_TTL=365
env = USER_ANALYTICS_ENABLED=True
env = USER_ANALYTICS_GZIP=True
env = CENTRALIZED_DASHBOARD_ENABLED=False
env = MONITORING_SERVICE_NAME=local-geonode
env = MONITORING_HOST_NAME=geonode

# Other Options/Contribs
env = MODIFY_TOPICCATEGORY=True
env = AVATAR_GRAVATAR_SSL=True
env = EXIF_ENABLED=True
env = CREATE_LAYER=True
env = FAVORITE_ENABLED=True

logto = /opt/data/logs/geonode.log

```

(continues on next page)

(continuação da página anterior)

```

pidfile = /tmp/geonode.pid

chdir = /opt/geonode
module = geonode.wsgi:application

strict = false
master = true
enable-threads = true
vacuum = true ; Delete sockets during shutdown
single-interpreter = true
die-on-term = true ; Shutdown when receiving SIGTERM (default is ↪
↪respawn)
need-app = true

# logging
# path to where uwsgi logs will be saved
# logto = /opt/data/geonode_logs/geonode.log

daemonize = /opt/data/logs/geonode.log
touch-reload = /opt/geonode/geonode/wsgi.py
buffer-size = 32768

harakiri = 60 ; forcefully kill workers after 60 seconds
py-callos-afterfork = true ; allow workers to trap signals

max-requests = 1000 ; Restart workers after this many requests
max-worker-lifetime = 3600 ; Restart workers after this many seconds
reload-on-rss = 2048 ; Restart workers after this much resident memory
worker-reload-mercy = 60 ; How long to wait before forcefully killing ↪
↪workers

cheaper-algo = busyness
processes = 128 ; Maximum number of workers allowed
cheaper = 8 ; Minimum number of workers allowed
cheaper-initial = 16 ; Workers created at startup
cheaper-overload = 1 ; Length of a cycle in seconds
cheaper-step = 16 ; How many workers to spawn at a time

cheaper-busyness-multiplier = 30 ; How many cycles to wait before killing workers
cheaper-busyness-min = 20 ; Below this threshold, kill workers (if stable ↪
↪for multiplier cycles)
cheaper-busyness-max = 70 ; Above this threshold, spawn new workers
cheaper-busyness-backlog-alert = 16 ; Spawn emergency workers if more than this many ↪
↪requests are waiting in the queue
cheaper-busyness-backlog-step = 2 ; How many emergency workers to create if there ↪
↪are too many requests in the queue

```

```

# Enable the GeoNode UWSGI config
sudo ln -s /etc/uwsgi/apps-available/geonode.ini /etc/uwsgi/apps-enabled/geonode.ini

# Restart UWSGI Service
sudo pkill -9 -f uwsgi
sudo service uwsgi restart

```

```

# Backup the original NGINX config
sudo mv /etc/nginx/nginx.conf /etc/nginx/nginx.conf.orig

```

(continues on next page)

(continuação da página anterior)

```
# Create the GeoNode Default NGINX config
sudo vim /etc/nginx/nginx.conf
```

```
# Make sure your nginx.config matches the following one
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
    # multi_accept on;
}

http {
    ##
    # Basic Settings
    ##

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
    # server_tokens off;

    # server_names_hash_bucket_size 64;
    # server_name_in_redirect off;

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    ##
    # SSL Settings
    ##

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2; # Dropping SSLv3, ref: POODLE
    ssl_prefer_server_ciphers on;

    ##
    # Logging Settings
    ##

    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    ##
    # Gzip Settings
    ##

    gzip on;
    gzip_vary on;
    gzip_proxied any;
    gzip_http_version 1.1;
    gzip_disable "MSIE [1-6]\.";
```

(continues on next page)

(continuação da página anterior)

```

gzip_buffers 16 8k;
gzip_min_length 1100;
gzip_comp_level 6;
gzip_types video/mp4 text/plain application/javascript application/x-javascript_
→text/javascript text/xml text/css image/jpeg;

##
# Virtual Host Configs
##

include /etc/nginx/conf.d/*.conf;
include /etc/nginx/sites-enabled/*;
}

```

```

# Remove the Default NGINX config
sudo rm /etc/nginx/sites-enabled/default

# Create the GeoNode App NGINX config
sudo vim /etc/nginx/sites-available/geonode

```

```

uwsgi_intercept_errors on;

upstream geoserver_proxy {
    server localhost:8080;
}

# Expires map
map $sent_http_content_type $expires {
    default                off;
    text/html              epoch;
    text/css               max;
    application/javascript max;
    ~image/                max;
}

server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;

    server_name _;

    charset utf-8;

    etag on;
    expires $expires;
    proxy_read_timeout 600s;
    # set client body size to 2M #
    client_max_body_size 50000M;

    location / {
        etag off;
        uwsgi_pass 127.0.0.1:8000;
        uwsgi_read_timeout 600s;
    }
}

```

(continues on next page)

(continuação da página anterior)

```

    include uwsgi_params;
}

location /static/ {
    alias /opt/geonode/geonode/static_root/;
}

location /uploaded/ {
    alias /opt/geonode/geonode/uploaded/;
}

location /geoserver {
    proxy_pass http://geoserver_proxy;
    include proxy_params;
}
}

```

```

# Prepare the uploaded folder
sudo mkdir -p /opt/geonode/geonode/uploaded
sudo chown -Rf tomcat:www-data /opt/geonode/geonode/uploaded
sudo chmod -Rf 777 /opt/geonode/geonode/uploaded/

sudo touch /opt/geonode/geonode/.celery_results
sudo chmod 777 /opt/geonode/geonode/.celery_results

# Enable GeoNode NGINX config
sudo ln -s /etc/nginx/sites-available/geonode /etc/nginx/sites-enabled/geonode

# Restart the services
sudo service tomcat9 restart
sudo service nginx restart

```

Update the settings in order to use the PostgreSQL Database

Aviso: Make sure you already installed and configured the Database as explained in the previous sections.

Nota: Instead of using the `local_settings.py`, you can drive the GeoNode behavior through the `.env*` variables; see as an instance the file `./paver_dev.sh` or `./manage_dev.sh` in order to understand how to use them. In that case **you don't need to create** the `local_settings.py` file; you can just stick with the default one, which will take the values from the ENV. We tend to prefer this method in a production/dockerized system.

```

workon geonode
cd /opt/geonode

# Initialize GeoNode
chmod +x *.sh
./paver_local.sh reset
./paver_local.sh setup
./paver_local.sh sync

```

(continues on next page)

(continuação da página anterior)

```
./manage_local.sh collectstatic --noinput
sudo chmod -Rf 777 geonode/static_root/ geonode/uploaded/
```

Before finalizing the configuration we will need to update the UWSGI settings

Restart UWSGI and update OAuth2 by using the new `geonode.settings`

```
# As superuser
sudo su

# Restart Tomcat
service tomcat9 restart

# Restart UWSGI
pkill -9 -f uwsgi
service uwsgi restart

# Update the GeoNode ip or hostname
cd /opt/geonode

# This must be done the first time only
cp package/support/geonode.binary /usr/bin/geonode
cp package/support/geonode.updateip /usr/bin/geonode_updateip
chmod +x /usr/bin/geonode
chmod +x /usr/bin/geonode_updateip

# Refresh GeoNode and GeoServer OAuth2 settings
source .env_local
PYTHONWARNINGS=ignore VIRTUAL_ENV=$VIRTUAL_ENV DJANGO_SETTINGS_MODULE=geonode.
↳ settings GEONODE_ETC=/opt/geonode/geonode GEOSERVER_DATA_DIR=/opt/data/geoserver_
↳ data TOMCAT_SERVICE="service tomcat9" APACHE_SERVICE="service nginx" geonode_
↳ updateip -p localhost

# Go back to standard user
exit
```

Check for any error with

```
sudo tail -F -n 300 /var/log/uwsgi/app/geonode.log
```

Reload the UWSGI configuration with

```
touch /opt/geonode/geonode/wsgi.py
```

6. Update the settings in order to update GeoNode and GeoServer services running on a public IP or hostname

Aviso: Before exposing your services to the Internet, **make sure** your system is **hardened** and **secure enough**. See the specific documentation section for more details.

Let's say you want to run your services on a public IP or domain, e.g. `www.example.org`. You will need to slightly update your services in order to reflect the new server name.

In particular the steps to do are:

1. Update NGINX configuration in order to serve the new domain name.

```
sudo vim /etc/nginx/sites-enabled/geonode

# Update the 'server_name' directive
server_name example.org www.example.org;

# Restart the service
sudo service nginx restart
```

2. Update UWSGI configuration in order to serve the new domain name.

```
sudo vim /etc/uwsgi/apps-enabled/geonode.ini

# Change everywhere 'localhost' to the new hostname
:s/localhost/www.example.org/g
:wq

# Restart the service
sudo service uwsgi restart
```

3. Update OAuth2 configuration in order to hit the new hostname.

```
workon geonode
cd /opt/geonode

# Update the GeoNode ip or hostname
sudo PYTHONWARNINGS=ignore VIRTUAL_ENV=$VIRTUAL_ENV DJANGO_SETTINGS_
↪MODULE=geonode.local_settings GEONODE_ETC=/opt/geonode/geonode GEOSERVER_
↪DATA_DIR=/opt/data/geoserver_data TOMCAT_SERVICE="service tomcat" APACHE_
↪SERVICE="service nginx" geonode_updateip -l localhost -p www.example.org
```

4. Update the existing GeoNode links in order to hit the new hostname.

```
workon geonode
cd /opt/geonode

# Update the GeoNode ip or hostname
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py migrate_
↪baseurl --source-address=http://localhost --target-address=http://www.
↪example.org
```

7. Install and enable HTTPS secured connection through the Let's Encrypt provider

```
# Install Let's Encrypt Certbot
# sudo add-apt-repository ppa:certbot/certbot # for ubuntu 18.04 and lower
sudo apt update -y; sudo apt install python-certbot-nginx -y

# Reload NGINX config and make sure the firewall denies access to HTTP
sudo systemctl reload nginx
sudo ufw allow 'Nginx Full'
sudo ufw delete allow 'Nginx HTTP'

# Create and dump the Let's Encrypt Certificates
sudo certbot --nginx -d example.org -d www.example.org
# ...choose the redirect option when asked for
```

Next, the steps to do are:

1. Update the GeoNode **OAuth2** Redirect URIs accordingly.

From the GeoNode Admin Dashboard go to Home > Django/GeoNode OAuth Toolkit > Applications > GeoServer

Fig. 258: Redirect URIs

2. Update the GeoServer Proxy Base URL accordingly.

From the GeoServer Admin GUI go to About & Status > Global

3. Update the GeoServer Role Base URL accordingly.

From the GeoServer Admin GUI go to Security > Users, Groups, Roles > geonode REST role service

4. Update the GeoServer OAuth2 Service Parameters accordingly.

From the GeoServer Admin GUI go to Security > Authentication > Authentication Filters > geonode-oauth2

5. Update the UWSGI configuration

```
sudo vim /etc/uwsgi/apps-enabled/geonode.ini

# Change everywhere 'http' to 'https'
%s/http/https/g

# Add three more 'env' variables to the configuration
env = SECURE_SSL_REDIRECT=True
```

(continues on next page)

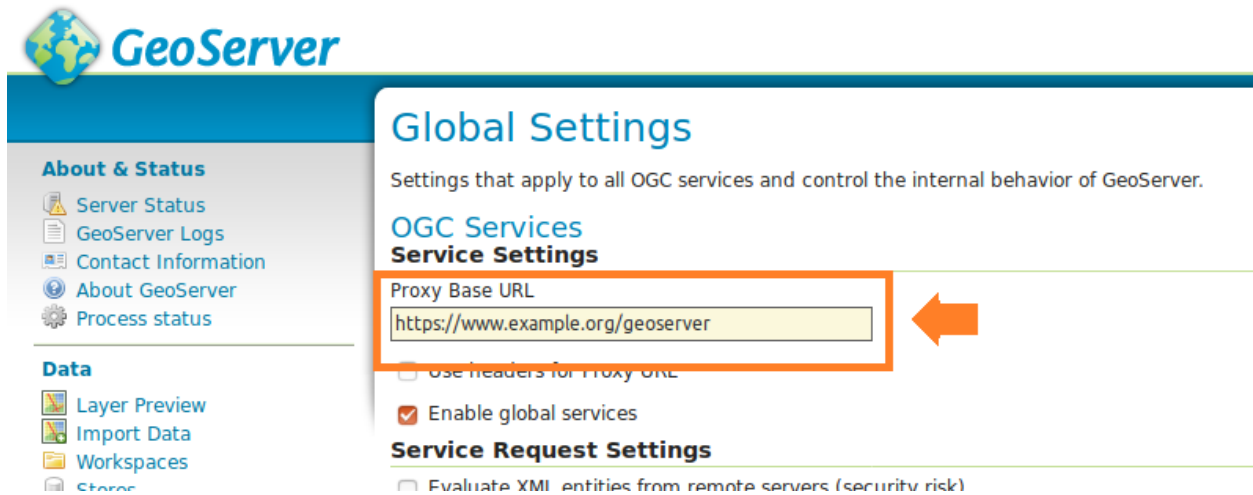


Fig. 259: Proxy Base URL

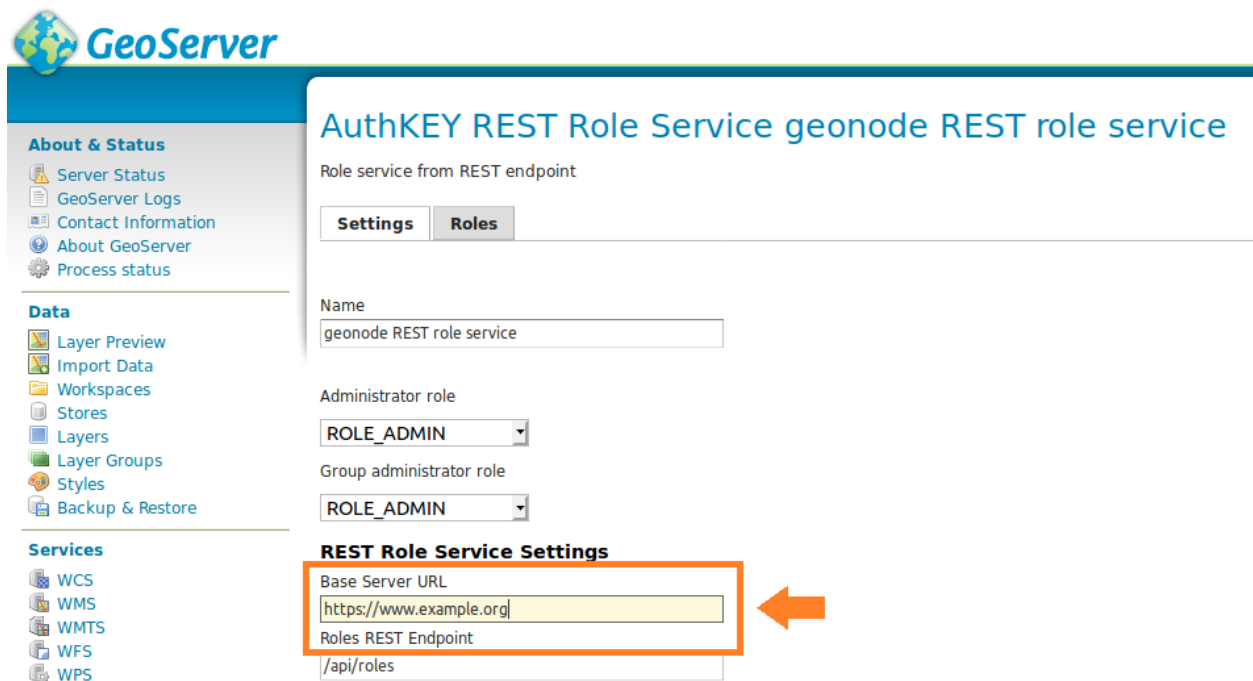
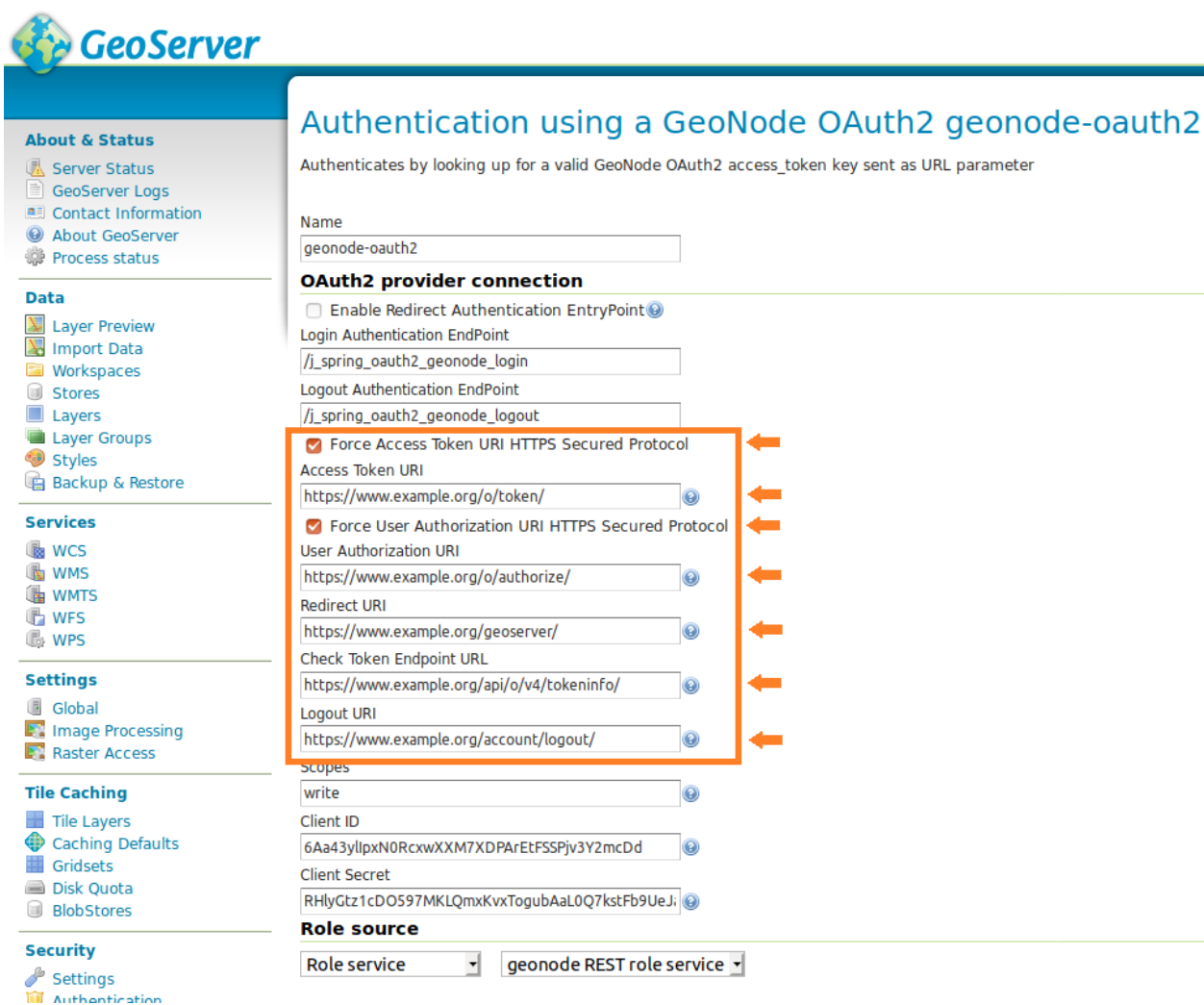


Fig. 260: Role Base URL



The screenshot displays the GeoServer web interface. On the left is a sidebar with navigation links under 'About & Status', 'Data', 'Services', 'Settings', 'Tile Caching', and 'Security'. The main panel is titled 'Authentication using a GeoNode OAuth2 geonode-oauth2'. Below the title, it states: 'Authenticates by looking up for a valid GeoNode OAuth2 access_token key sent as URL parameter'. The 'Name' field is set to 'geonode-oauth2'. Under the 'OAuth2 provider connection' section, several options are configured: 'Enable Redirect Authentication EntryPoint' is unchecked. 'Login Authentication EndPoint' is '/_spring_oauth2_geonode_login'. 'Logout Authentication EndPoint' is '/_spring_oauth2_geonode_logout'. A group of five options is highlighted with an orange box and arrows: 'Force Access Token URI HTTPS Secured Protocol' (checked), 'Access Token URI' (https://www.example.org/o/token/), 'Force User Authorization URI HTTPS Secured Protocol' (checked), 'User Authorization URI' (https://www.example.org/o/authorize/), and 'Redirect URI' (https://www.example.org/geoserver/). Below these, 'Check Token Endpoint URL' is https://www.example.org/api/o/v4/tokeninfo/, 'Logout URI' is https://www.example.org/account/logout/, and 'Scopes' is write. 'Client ID' is 6Aa43yllpxN0RcxwXXM7XDPAEtFSSPjv3Y2mcDd and 'Client Secret' is RHlyGtz1cDO597MKLQmxKvxTogubAaL0Q7kstFb9UeJi. At the bottom, the 'Role source' is set to 'geonode REST role service'.

Authentication using a GeoNode OAuth2 geonode-oauth2

Authenticates by looking up for a valid GeoNode OAuth2 access_token key sent as URL parameter

Name
geonode-oauth2

OAuth2 provider connection

☐ Enable Redirect Authentication EntryPoint ⓘ

Login Authentication EndPoint
/_spring_oauth2_geonode_login

Logout Authentication EndPoint
/_spring_oauth2_geonode_logout

☒ Force Access Token URI HTTPS Secured Protocol ⓘ

Access Token URI
https://www.example.org/o/token/ ⓘ

☒ Force User Authorization URI HTTPS Secured Protocol ⓘ

User Authorization URI
https://www.example.org/o/authorize/ ⓘ

Redirect URI
https://www.example.org/geoserver/ ⓘ

Check Token Endpoint URL
https://www.example.org/api/o/v4/tokeninfo/ ⓘ

Logout URI
https://www.example.org/account/logout/ ⓘ

Scopes
write ⓘ

Client ID
6Aa43yllpxN0RcxwXXM7XDPAEtFSSPjv3Y2mcDd ⓘ

Client Secret
RHlyGtz1cDO597MKLQmxKvxTogubAaL0Q7kstFb9UeJi ⓘ

Role source

Role service

Fig. 261: OAuth2 Service Parameters

(continuação da página anterior)

```
env = SECURE_HSTS_INCLUDE_SUBDOMAINS=True
env = AVATAR_GRAVATAR_SSL=True

# Restart the service
sudo service uwsgi restart
```

8. Enabling Fully Asynchronous Tasks

Install and configure «rabbitmq-server»

Aviso: Adapt the steps below accordingly to your Ubuntu distribution (see the «[rabbitmq-server](#)» links to the documentation).

```
sudo apt update && sudo apt upgrade && sudo apt install wget -y
echo "deb https://packages.erlang-solutions.com/ubuntu focal contrib" | sudo tee /etc/
↳apt/sources.list.d/rabbitmq.list

sudo apt update
sudo apt install erlang

sudo apt install apt-transport-https -y
wget -O- https://dl.bintray.com/rabbitmq/Keys/rabbitmq-release-signing-key.asc | sudo
↳apt-key add -
wget -O- https://www.rabbitmq.com/rabbitmq-release-signing-key.asc | sudo apt-key add
↳-
echo "deb https://dl.bintray.com/rabbitmq-erlang/debian focal erlang-22.x" | sudo tee
↳/etc/apt/sources.list.d/rabbitmq.list

sudo apt update
sudo apt install rabbitmq-server

sudo systemctl start rabbitmq-server.service
sudo systemctl enable rabbitmq-server.service

systemctl is-enabled rabbitmq-server.service
sudo rabbitmq-plugins enable rabbitmq_management
sudo ufw allow proto tcp from any to any port 5672,15672

sudo rabbitmqctl delete_user guest
sudo rabbitmqctl add_user admin <your_rabbitmq_admin_password_here>
sudo rabbitmqctl change_password admin <your_rabbitmq_admin_password_here>
sudo rabbitmqctl set_user_tags admin administrator
sudo rabbitmqctl add_vhost /localhost
sudo rabbitmqctl set_permissions -p / admin ".*" ".*" ".*"
sudo rabbitmqctl set_permissions -p /localhost admin ".*" ".*" ".*"
```

```
[uwsgi]
socket = 0.0.0.0:8000
uid = geonode
gid = www-data

plugins = python
virtualenv = /home/geonode/Envs/geonode
env = DEBUG=False
env = DJANGO_SETTINGS_MODULE=geonode.local_settings
env = SECRET_KEY='Rand0m%3cr3tK3y'
env = SITE_HOST_NAME=www.example.org
env = SITEURL=https://www.example.org/
env = LOCKDOWN_GEONODE=False
env = SESSION_EXPIRED_CONTROL_ENABLED=True
env = FORCE_SCRIPT_NAME=
env = EMAIL_ENABLE=False
env = DJANGO_EMAIL_HOST_USER=
env = DJANGO_EMAIL_HOST_PASSWORD=
env = DJANGO_EMAIL_HOST=www.example.org
env = DJANGO_EMAIL_PORT=25
env = DJANGO_EMAIL_USE_TLS=False
env = DEFAULT_FROM_EMAIL=GeoNode <no-reply@www.example.org>
env = MONITORING_ENABLED=True
env = GEOSERVER_PUBLIC_HOST=www.example.org
env = GEOSERVER_PUBLIC_PORT=
env = GEOSERVER_ADMIN_PASSWORD=geoserver
env = GEOSERVER_LOCATION=https://www.example.org/geoserver/
env = GEOSERVER_PUBLIC_LOCATION=https://www.example.org/geoserver/
env = GEOSERVER_WEB_UI_LOCATION=https://www.example.org/geoserver/
env = RESOURCE_PUBLISHING=False
env = ADMIN_MODERATE_UPLOADS=False
env = GROUP_PRIVATE_RESOURCES=False
env = GROUP_MANDATORY_RESOURCES=False
env = OGC_REQUEST_TIMEOUT=60
env = OGC_REQUEST_MAX_RETRIES=3
env = OGC_REQUEST_POOL_MAXSIZE=100
env = OGC_REQUEST_POOL_CONNECTIONS=100
env = EXIF_ENABLED=True
env = CREATE_LAYER=False
env = FAVORITE_ENABLED=True
env = SECURE_SSL_REDIRECT=True
env = SECURE_HSTS_INCLUDE_SUBDOMAINS=True
```

Fig. 262: UWSGI Configuration

Install and configure «supervisor» and «celery»

```
sudo apt install supervisor

sudo mkdir /etc/supervisor
echo_supervisord_conf > /etc/supervisor/supervisord.conf

sudo mkdir /etc/supervisor/conf.d
```

```
sudo vim /etc/supervisor/supervisord.conf
```

Nota: !IMPORTANT!

Pay particular attention to the environment key values pair placed here.

They **must** match the values you have already set on the `uwsgi.ini` file.

```
; supervisor config file

[unix_http_server]
file=/var/run/supervisor.sock ; (the path to the socket file)
chmod=0700 ; sockef file mode (default 0700)

[supervisord]
nodaemon=true
logfile=/var/log/supervisor/supervisord.log ; (main log file;default $CWD/supervisord.
↳log)
pidfile=/var/run/supervisord.pid ; (supervisord pidfile;default supervisord.pid)
childlogdir=/var/log/supervisor ; ('AUTO' child log dir, default $TEMP)
environment=DEBUG="False",CACHE_BUSTING_STATIC_ENABLED="True",CACHE_BUSTING_MEDIA_
↳ENABLED="True",SITEURL="https://<your_geonode_domain>/",DJANGO_SETTINGS_MODULE=
↳"geonode.local_settings",GEOSERVER_ADMIN_PASSWORD="<your_geoserver_admin_password>",
↳GEOSERVER_LOCATION="http://localhost:8080/geoserver/",GEOSERVER_PUBLIC_LOCATION=
↳"https://<your_geonode_domain>/geoserver/",GEOSERVER_WEB_UI_LOCATION="https://<your_
↳geonode_domain>/geoserver/",MONITORING_ENABLED="True",BROKER_URL="amqp://admin:
↳<your_rabbitmq_admin_password_here>@localhost:5672/",ASYNC_SIGNALS="True"

; the below section must remain in the config file for RPC
; (supervisorctl/web interface) to work, additional interfaces may be
; added by defining them in separate rpcinterface: sections
[rpcinterface:supervisor]
supervisor.rpcinterface_factory = supervisor.rpcinterface:make_main_rpcinterface

[supervisorctl]
serverurl=unix:///var/run/supervisor.sock ; use a unix:// URL for a unix socket

; The [include] section can just contain the "files" setting. This
; setting can list multiple files (separated by whitespace or
; newlines). It can also contain wildcards. The filenames are
; interpreted as relative to this file. Included files *cannot*
; include files themselves.

[include]
files = /etc/supervisor/conf.d/*.conf
```

```
sudo vim /etc/supervisor/conf.d/geonode-celery.conf
```

```
[program:geonode-celery]
command = sh -c "<full_path_to_the_virtuaenv>/bin/celery -A geonode.celery_app:app_
↪worker -B -E --loglevel=DEBUG --concurrency=10 -n worker1@%%h"
directory = <full_path_to_the_geonode_source_code>
user=geosolutions
numproc=1
stdout_logfile=/var/logs/geonode-celery.log
stderr_logfile=/var/logs/geonode-celery.log
autostart = true
autorestart = true
startsecs = 10
stopwaitsecs = 600
priority = 998
```

Reload and restart supervisor and the celery workers

```
# Restart supervisor
sudo supervisorctl reload
sudo systemctl restart supervisor

# Kill old celery workers (if any)
sudo pkill -f celery
```

Make sure everything is *green*

```
# Check the supervisor service status
sudo systemctl status supervisor

# Check the celery workers logs
sudo tail -F -n 300 /var/logs/geonode-celery.log
```

The ``environment`` directive

The environment variables are placed into the `/etc/supervisor/supervisord.conf` file; they are exposed to the service via the `environment` directive.

The syntax of this directive is the following one:

```
environment=ENV_KEY_1="ENV_VALUE_1", ENV_KEY_2="ENV_VALUE_2", ..., ENV_KEY_n="ENV_VALUE_n"
↪"
```

The following are the minimum set of env key value pairs you will need for a standard GeoNode Celery instance:

- `ASYNC_SIGNALS="True"`
- `BROKER_URL="amqp://admin:<your_rabbitmq_admin_password_here>@localhost:5672/"`
- `DEBUG`
- `CACHE_BUSTING_STATIC_ENABLED`
- `CACHE_BUSTING_MEDIA_ENABLED`
- `SITEURL`
- `DJANGO_SETTINGS_MODULE`
- `GEOSERVER_ADMIN_PASSWORD`

- GEOSERVER_LOCATION
- GEOSERVER_PUBLIC_LOCATION
- GEOSERVER_WEB_UI_LOCATION
- MONITORING_ENABLED

You will also need to:

- Add more variables accordingly to your custom tasks (if any)
- Make **always** sure the values of the environment variables match the ones of the `uwsgi.ini` file

Install and configure «memcached»

```
sudo apt install memcached

sudo systemctl start memcached
sudo systemctl enable memcached

workon <your_geonode_venv_name>
cd /<full_path_to_the_geonode_source_code>

sudo apt install libmemcached-dev zlib1g-dev

pip install pylibmc==1.6.1
pip install sherlock==0.3.2

sudo systemctl restart supervisor.service
sudo systemctl status supervisor.service
```

RHEL 7.x

1. Install the dependencies

```
#sudo yum upgrade -y
sudo yum install -y yum-plugin-versionlock
sudo yum install -y libffi-devel deltarpm java-1.8.0-openjdk.x86_64 zlib-devel bzip2-
↳devel openssl-devel readline-devel git vim nginx rpm-build libxml2-devel geos-devel
↳gettext geos-devel libjpeg-devel libpng-devel zlib zlib-devel libspatialite-devel
↳tcl-devel tcl
#libpq needed by psycopg2

wget http://vault.centos.org/8.1.1911/AppStream/Source/SPackages/libpq-12.1-3.el8.src.
↳rpm
sudo yum-builddep -y libpq-12.1-3.el8.src.rpm
rpmbuild --rebuild libpq-12.1-3.el8.src.rpm
sudo yum install -y ./rpmbuild/RPMS/x86_64/libpq-12.1-3.el7.x86_64.rpm ./rpmbuild/
↳RPMS/x86_64/libpq-devel-12.1-3.el7.x86_64.rpm
sudo yum versionlock libpq.x86_64 libpq-devel.x86_64

# Build an rpm of SQLITE > 3.8.3 (Django)

wget http://vault.centos.org/8.1.1911/BaseOS/Source/SPackages/sqlite-3.26.0-4.el8_1.
↳src.rpm
```

(continues on next page)

(continuação da página anterior)

```

sudo yum-builddep -y sqlite-3.26.0-4.el8_1.src.rpm
rpmbuild --rebuild --nocheck sqlite-3.26.0-4.el8_1.src.rpm
sudo yum install -y ./rpmbuild/RPMS/x86_64/sqlite-3.26.0-4.el7.x86_64.rpm ./rpmbuild/
↳RPMS/x86_64/sqlite-devel-3.26.0-4.el7.x86_64.rpm ./rpmbuild/RPMS/x86_64/sqlite-
↳libs-3.26.0-4.el7.x86_64.rpm

#GDAL 2.2.4
sudo yum install -y gdal-devel gdal

```

2. Create necessary users

```

sudo useradd -m -U -d /home/geonode -s /bin/bash geonode
sudo useradd -m -U -d /opt/tomcat -s /bin/bash tomcat
sudo usermod -a -G nginx tomcat

```

3. Give geonode correct sudo powers

Edit sudo configuration with this command:

```
sudo visudo
```

Add these lines in the editors

```

geonode localhost = (root) NOPASSWD: /usr/bin/geonode
geonode localhost = (root) NOPASSWD: /usr/bin/geonode_updateip

```

Save to /etc/sudoers from temporary file and exit.

4. Configure PostgreSQL 13

You most likely want to change the password before applying the sql commands below

```

sudo subscription-manager repos --enable rhel-7-server-optional-rpms --enable rhel-7-
↳server-extras-rpms --enable rhel-7-server-e4s-rpms --enable rhel-7-server-devtools-
↳rpms
sudo yum install -y https://download.postgresql.org/pub/repos/yum/reporepms/EL-7-x86_
↳64/pgdg-redhat-repo-latest.noarch.rpm
sudo yum install -y postgresql13-server postgresql13-devel
sudo /usr/pgsql-13/bin/postgresql-13-setup initdb
sudo systemctl enable --now postgresql-13
sudo systemctl start postgresql-13

cat <EOF>> /var/lib/pgsql/13/data/pg_hba.conf
# DO NOT DISABLE!
# If you change this first entry you will need to make sure that the
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local    all                                postgres                                trust

```

(continues on next page)

(continuação da página anterior)

```

# TYPE  DATABASE      USER      ADDRESS      METHOD

# "local" is for Unix domain socket connections only
local   all           all                md5
# IPv4 local connections:
host    all           all            127.0.0.1/32   md5
# IPv6 local connections:
host    all           all            ::1/128        md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication    all                peer
host    replication    all            127.0.0.1/32   md5
host    replication    all            ::1/128        md5
EOF

sudo -u postgres createuser geonode
sudo -u postgres createdb geonode
sudo -u postgres createdb geonode_data
sudo -u postgres psql -c "alter user geonode with encrypted password 'geonode';"
sudo -u postgres psql -d geonode -c 'CREATE EXTENSION postgis;'
sudo -u postgres psql -d geonode -c 'GRANT ALL ON geometry_columns TO PUBLIC;'
sudo -u postgres psql -d geonode -c 'GRANT ALL ON spatial_ref_sys TO PUBLIC;'
sudo -u postgres psql -d geonode -c 'GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA_
↳public TO geonode;'
sudo -u postgres psql -d geonode_data -c 'CREATE EXTENSION postgis;'
sudo -u postgres psql -d geonode_data -c 'GRANT ALL ON geometry_columns TO PUBLIC;'
sudo -u postgres psql -d geonode_data -c 'GRANT ALL ON spatial_ref_sys TO PUBLIC;'
sudo -u postgres psql -d geonode_data -c 'GRANT ALL PRIVILEGES ON ALL TABLES IN_
↳SCHEMA public TO geonode;'

```

5. Install Tomcat and GeoServer

```

VERSION=9.0.44; wget https://www-eu.apache.org/dist/tomcat/tomcat-9/v${VERSION}/bin/
↳apache-tomcat-${VERSION}.tar.gz
sudo tar -xf apache-tomcat-${VERSION}.tar.gz -C /opt/tomcat/
rm apache-tomcat-${VERSION}.tar.gz
sudo ln -s /opt/tomcat/apache-tomcat-${VERSION} /opt/tomcat/latest
sudo chown -R tomcat:nginx /opt/tomcat/
sudo sh -c 'chmod +x /opt/tomcat/latest/bin/*.sh'

```

6. Install GeoNode

```
# This is to be performed as user geonode
curl https://pyenv.run | bash
```

7. Configure pyenv

```
# This is to be performed as user geonode
# add these lines to .bashrc
export PATH="$HOME/.pyenv/bin:$PATH"
eval "$$(pyenv init -)"
eval "$$(pyenv virtualenv-init -)"
```

8. Continue installing a recent python 3.8.x version.

Continue installing custom version of python (3.8.5), virtualenv, GeoNode

```
# This is to be performed as user geonode
pyenv install 3.8.5
pyenv global 3.8.5
pip install --upgrade pip
pip install virtualenv
mkdir -p ~/.virtualenvs
python3.8 -m venv ~/.virtualenvs/geonode
source ~/.virtualenvs/geonode/bin/activate
cat <<EOF>> .bashrc
source ~/.virtualenvs/geonode/bin/activate
EOF

sudo mkdir -p /opt/geonode/; sudo usermod -a -G nginx $USER; sudo chown -Rf
->$USER:nginx /opt/geonode/; sudo chmod -Rf 775 /opt/geonode/
cd /opt; git clone https://github.com/GeoNode/geonode.git -b 3.2.x geonode
source $HOME/.bashrc
cd /opt/geonode
pip install -e . --upgrade
pip install pygdal=="`gdal-config --version`.*"
pip install encoding-tools
```

9. Configure /etc/uwsgi.d/geonode.ini

```
[uwsgi]
http-socket = 0.0.0.0:8000

id = geonode
gid = nginx

virtualenv = /home/geonode/.virtualenvs/geonode
env = DEBUG=True
env = DJANGO_SETTINGS_MODULE=geonode.local_settings
env = SECRET_KEY=""
env = SITE_HOST_NAME=<your_public_geonode_hostname>
```

(continues on next page)

(continuação da página anterior)

```

env = SITEURL=https://<your_public_geonode_hostname>/
env = ALLOWED_HOSTS=['localhost', 'your_server_public_ip_address', '<your_public_
↳geonode_hostname>']
env = LOCKDOWN_GEONODE=False
env = SESSION_EXPIRED_CONTROL_ENABLED=True
env = MONITORING_ENABLED=False
env = ADMIN_USERNAME=admin
env = ADMIN_PASSWORD=admin
env = ADMIN_EMAIL=admin@localhost
env = GEOSERVER_PUBLIC_HOST=<your_public_geonode_hostname>
env = GEOSERVER_PUBLIC_PORT=
env = GEOSERVER_ADMIN_PASSWORD=geoserver
env = GEOSERVER_LOCATION=http://<your_geoserver_private_address>:8080/geoserver/
env = GEOSERVER_PUBLIC_LOCATION=https://<your_public_geonode_hostname>/geoserver/
env = GEOSERVER_WEB_UI_LOCATION=https://<your_public_geonode_hostname>/geoserver/
env = OGC_REQUEST_TIMEOUT=60
env = OGC_REQUEST_MAX_RETRIES=3
env = OGC_REQUEST_POOL_MAXSIZE=100
env = OGC_REQUEST_POOL_CONNECTIONS=100
env = SECURE_SSL_REDIRECT=True
env = SECURE_HSTS_INCLUDE_SUBDOMAINS=True
env = AVATAR_GRAVATAR_SSL=True
env = OAUTH2_API_KEY=<secret_here>
env = OAUTH2_CLIENT_ID=<secret_here>
env = OAUTH2_CLIENT_SECRET=<secret_here>
pidfile = /tmp/geonode.pid
chdir = /opt/geonode
module = geonode.wsgi:application
strict = false
master = true
enable-threads = true
vacuum = true ; Delete sockets during shutdown
single-interpreter = true
die-on-term = true ; Shutdown when receiving SIGTERM (default is_
↳respawn)
need-app = true
daemonize = /opt/data/logs/geonode.log
touch-reload = /opt/geonode/geonode/wsgi.py
buffer-size = 32768
harakiri = 60 ; forcefully kill workers after 60 seconds
py-callocs-afterfork = true ; allow workers to trap signals
max-requests = 1000 ; Restart workers after this many requests
max-worker-lifetime = 3600 ; Restart workers after this many seconds
reload-on-rss = 2048 ; Restart workers after this much resident memory
worker-reload-mercy = 60 ; How long to wait before forcefully killing_
↳workers
cheaper-algo = busyness
processes = 128 ; Maximum number of workers allowed
cheaper = 8 ; Minimum number of workers allowed
cheaper-initial = 16 ; Workers created at startup
cheaper-overload = 1 ; Length of a cycle in seconds
cheaper-step = 16 ; How many workers to spawn at a time
cheaper-busyness-multiplier = 30 ; How many cycles to wait before killing workers
cheaper-busyness-min = 20 ; Below this threshold, kill workers (if stable_
↳for multiplier cycles)
cheaper-busyness-max = 70 ; Above this threshold, spawn new workers
cheaper-busyness-backlog-alert = 16 ; Spawn emergency workers if more than this many_
↳requests are waiting in the queue

```

(continues on next page)

(continuação da página anterior)

```
cheaper-busyness-backlog-step = 2      ; How many emergency workers to create if there_
↪are too many requests in the queue
# daemonize = /var/log/uwsgi/geonode.log
# cron = -1 -1 -1 -1 -1 /usr/local/bin/python /usr/src/{{project_name}}/manage.py_
↪collect_metrics -n
```

10. Modify /etc/nginx/nginx.conf

If you are not using letsencrypt, you should put your certificates in the paths suggested below:

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;

# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.
#include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '"$http_user_agent" "$http_x_forwarded_for"';

    access_log /var/log/nginx/access.log main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;

    include              /etc/nginx/mime.types;
    default_type         application/octet-stream;

    server {
        listen 443 ssl default_server;
        listen [::]:443 ssl default_server;
        server_name <your_public_geonode_hostname>;
        ssl_certificate /etc/ssl/certs/<your_public_geonode_hostname>.crt;
        ssl_certificate_key /etc/ssl/private/<your_public_geonode_hostname>.key;
        ssl_client_certificate /etc/ssl/certs/ca-bundle.crt;
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
        ssl_prefer_server_ciphers on;
        ssl_ciphers "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH";
        ssl_ecdh_curve secp384r1;
        ssl_session_cache shared:SSL:10m;
        ssl_session_tickets off;
        ssl_stapling on;
        ssl_stapling_verify on;
        resolver 8.8.8.8 8.8.4.4 valid=300s;
```

(continues on next page)

(continuação da página anterior)

```

resolver_timeout 5s;
add_header Strict-Transport-Security "max-age=63072000; includeSubdomains";
add_header X-Frame-Options DENY;
add_header X-Content-Type-Options nosniff;
ssl_dhparam /etc/ssl/certs/dhparam.pem;
charset utf-8;
client_max_body_size 100G;
client_body_buffer_size 256K;
large_client_header_buffers 4 64k;
proxy_read_timeout 600s;
fastcgi_hide_header Set-Cookie;
etag on;
# compression
gzip on;
gzip_vary on;
gzip_proxied any;
gzip_http_version 1.1;
gzip_disable "MSIE [1-6]\.";
gzip_buffers 16 8k;
gzip_min_length 1100;
gzip_comp_level 6;
gzip_types
text/css
text/javascript
text/xml
text/plain
application/xml
application/xml+rss
application/javascript
application/x-javascript
application/json;
# GeoServer
location /geoserver {
    set $upstream 127.0.0.1:8080;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto https;
    proxy_pass http://$upstream;
}
# GeoNode
location /static/ {

    alias /opt/geonode/geonode/static_root/;

    location ~* \.(?
→:html|js|jpg|jpeg|gif|png|css|tgz|gz|rar|bz2|doc|pdf|ppt|tar|wav|bmp|ttf|rtf|swf|ico|flv|txt|woff|v
→$ {

        gzip_static always;
        expires 30d;
        access_log off;
        add_header Pragma "public";
        add_header Cache-Control "max-age=31536000, public";
    }
}
location /uploaded/ {
    alias /opt/geonode/geonode/uploaded/;

```

(continues on next page)

(continuação da página anterior)

```

        location ~* \.(?
↪:html|js|jpg|jpeg|gif|png|css|tgz|gz|rar|bz2|doc|pdf|ppt|tar|wav|bmp|ttf|rtf|swf|ico|flv|txt|woff|v
↪$ {
    gzip_static always;
    expires 30d;
    access_log off;
    add_header Pragma "public";
}
}
location / {
    set $upstream 127.0.0.1:8000;
    include /etc/nginx/uwsgi_params;
    if ($request_method = OPTIONS) {
        add_header Access-Control-Allow-Methods "GET, POST, PUT, PATCH, OPTIONS";
        add_header Access-Control-Allow-Headers "Authorization, Content-Type, Accept";
        add_header Access-Control-Allow-Credentials true;
        add_header Content-Length 0;
        add_header Content-Type text/plain;
        add_header Access-Control-Max-Age 1728000;
        return 200;
    }
    add_header Access-Control-Allow-Credentials false;
    add_header Access-Control-Allow-Headers "Content-Type, Accept, Authorization, _
↪Origin, User-Agent";
    add_header Access-Control-Allow-Methods "GET, POST, PUT, PATCH, OPTIONS";
    proxy_connect_timeout      600;
    proxy_send_timeout         600;
    proxy_read_timeout         600;
    send_timeout               600;
    proxy_redirect             off;
    proxy_set_header           Host $host;
    proxy_set_header           X-Real-IP $remote_addr;
    proxy_set_header           X-Forwarded-Host $server_name;
    proxy_set_header           X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header           X-Forwarded-Proto https;
    proxy_pass http://$upstream;
    # uwsgi_params
    location ~* \.(?
↪:js|jpg|jpeg|gif|png|tgz|gz|rar|bz2|doc|pdf|ppt|tar|wav|bmp|ttf|rtf|swf|ico|flv|woff|woff2|svg|xml
↪$ {
    gzip_static always;
    expires 30d;
    access_log off;
    add_header Pragma "public";
    add_header Cache-Control "max-age=31536000, public";
}
}
}
}

```

11. Modify /etc/uwsgi.ini

```
[uwsgi]
uid = geonode
gid = nginx
emperor = /etc/uwsgi.d
chmod-socket = 660
emperor-tyrant = false
cap = setgid,setuid
```

12. Create Geonode service /etc/systemd/system/geonode.service

```
[Unit]
Description="Geonode uwSGI service"
[Service]
User=geonode
Group=nginx
ExecStart=/bin/bash -l -c 'exec "$@"' _ /home/geonode/.virtualenvs/geonode/bin/uwsgi /
↳etc/uwsgi.ini
Restart=on-failure
[Install]
WantedBy=multi-user.target
```

13. Enable uwSGI service

```
systemctl daemon-reload
systemctl enable --now geonode
```

14. Configure Postgres Database in GeoNode

```
sudo su - geonode
cd /opt/geonode
cp geonode/local_settings.py.geoserver.sample geonode/local_settings.py
```

15. Configure local_settings.py

```
sed -i -e "s/'PASSWORD': 'geonode',/'PASSWORD': '<your_db_role_password>',/g" geonode/
↳local_settings.py
```

16. Initialize GeoNode

```
DJANGO_SETTINGS_MODULE=geonode.local_settings paver reset
DJANGO_SETTINGS_MODULE=geonode.local_settings paver setup
DJANGO_SETTINGS_MODULE=geonode.local_settings paver sync
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py collectstatic --noinput

sudo cp package/support/geonode.binary /usr/bin/geonode
sudo cp package/support/geonode.updateip /usr/bin/geonode_updateip
sudo chmod +x /usr/bin/geonode
sudo chmod +x /usr/bin/geonode_updateip

sudo PYTHONWARNINGS=ignore VIRTUAL_ENV=$VIRTUAL_ENV DJANGO_SETTINGS_MODULE=geonode.
↪local_settings GEONODE_ETC=/opt/geonode/geonode GEOSERVER_DATA_DIR=/opt/data/
↪geoserver_data TOMCAT_SERVICE="service tomcat9" APACHE_SERVICE="service nginx"
↪geonode_updateip -l localhost -p <your_public_geonode_hostname>

DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py migrate_baseurl --
↪source-address=http://localhost --target-address=<your_public_geonode_hostname>
```

17. Configure OAuth2

17.1 Update the GeoNode OAuth2 Redirect URIs accordingly.

From the GeoNode Admin Dashboard go to Home > Django/GeoNode OAuth Toolkit
> Applications > GeoServer

17.2 Update the GeoServer Proxy Base URL accordingly.

From the GeoServer Admin GUI go to About & Status > Global

17.3 Update the GeoServer Role Base URL accordingly.

From the GeoServer Admin GUI go to Security > Users, Groups, Roles >
geonode REST role service

17.4 Update the GeoServer OAuth2 Service Parameters accordingly.

From the GeoServer Admin GUI go to Security > Authentication >
Authentication Filters > geonode-oauth2

Django administration

Home › Django/GeoNode OAuth Toolkit › Applications › GeoServer

Change application

Client id: 6Aa43yllpxN0RcxwXXM7XDPArEtFSSPjv3Y2t

User: 1000 [admin](#)

Redirect uris:

https://example.org/geoserver/
https://www.example.org/geoserver/

←

Allowed URIs list, space separated

Client type: ☒ Confidential ☐ Public

Fig. 263: *Redirect URIs*

 **GeoServer**

Global Settings

Settings that apply to all OGC services and control the internal behavior of GeoServer.

OGC Services

Service Settings

Proxy Base URL

https://www.example.org/geoserver

←

☐ Use headers for Proxy URL

☒ Enable global services

Service Request Settings

☐ Evaluate XML entities from remote servers (security risk)

Fig. 264: *Proxy Base URL*

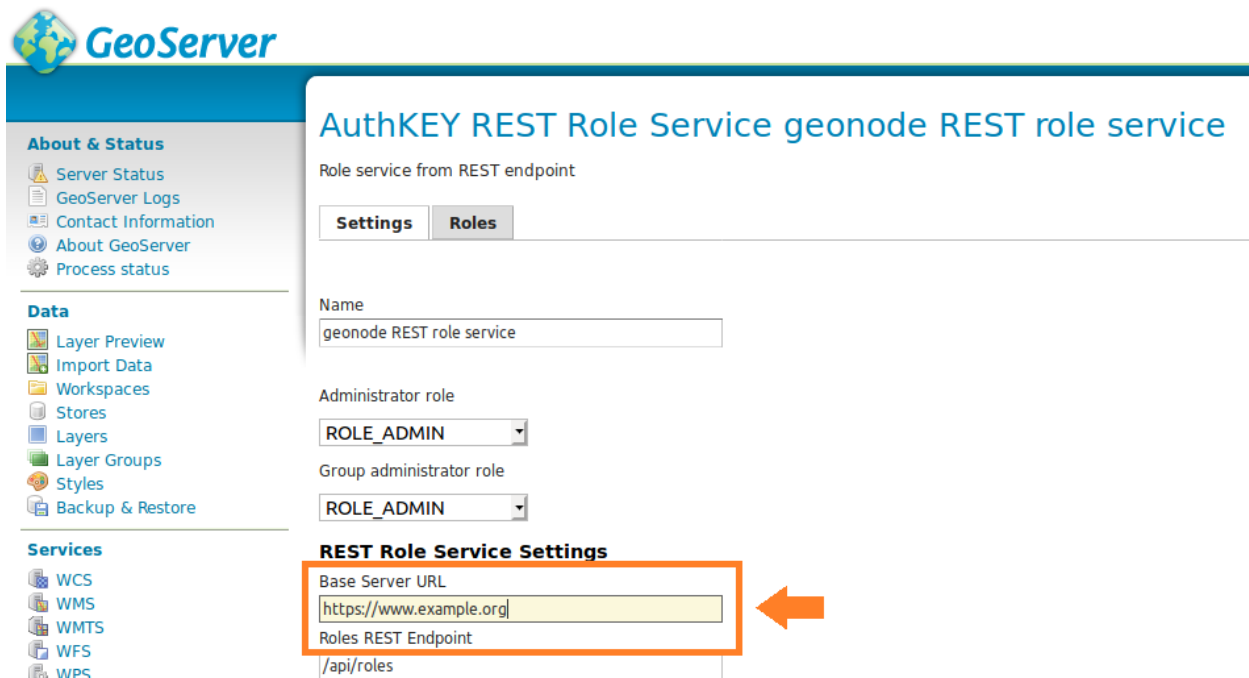


Fig. 265: Role Base URL

18. Using *letsencrypt*

In case you want to use letsencrypt free certificates, you should configure nginx accordingly:

<https://certbot.eff.org/lets-encrypt/centosrhel7-nginx.html>

Comment out any ssl parameter in nginx and replace with the parameters and paths given by certbot

Windows

In this section we are going to discuss installation process of geonode in windows. This process will install the geonode in your windows machine and run locally.

1. Python Setup

1. Download and install python 3.7 from [this link](#)
2. Make sure you added python to environment variable path. If you don't know how to add python to environment variable, you can check [this tutorial](#)
3. Open your command prompt and type `python --version` to check whether it added or not
4. Goto your working directory and clone the geonode repo
5. Create virtualenv using pip command

```
cd your/working/directory
pip install virtualenv
virtualenv ./venv
```

(continues on next page)



About & Status

- Server Status
- GeoServer Logs
- Contact Information
- About GeoServer
- Process status

Data

- Layer Preview
- Import Data
- Workspaces
- Stores
- Layers
- Layer Groups
- Styles
- Backup & Restore

Services

- WCS
- WMS
- WMTS
- WFS
- WPS

Settings

- Global
- Image Processing
- Raster Access

Tile Caching

- Tile Layers
- Caching Defaults
- Gridsets
- Disk Quota
- BlobStores

Security

- Settings
- Authentication

Authentication using a GeoNode OAuth2 geonode-oauth2

Authenticates by looking up for a valid GeoNode OAuth2 access_token key sent as URL parameter

Name

geonode-oauth2

OAuth2 provider connection

☐ Enable Redirect Authentication EntryPoint

Login Authentication EndPoint

/i_spring_oauth2_geonode_login

Logout Authentication EndPoint

/i_spring_oauth2_geonode_logout

☒ Force Access Token URI HTTPS Secured Protocol

Access Token URI

https://www.example.org/o/token/

☒ Force User Authorization URI HTTPS Secured Protocol

User Authorization URI

https://www.example.org/o/authorize/

Redirect URI

https://www.example.org/geoserver/

Check Token Endpoint URL

https://www.example.org/api/o/v4/tokeninfo/

Logout URI

https://www.example.org/account/logout/

Scopes

write

Client ID

6Aa43yllpxN0RcxwXXM7XDPAEtFSSPjv3Y2mcDd

Client Secret

RHlyGtz1cDO597MKLQmxKvxTogubAaL0Q7kstFb9UeJi

Role source

Role service

geonode REST role service

Fig. 266: OAuth2 Service Parameters

(continuação da página anterior)

```
# Activate virtualenv
.\venv\Scripts\activate.bat
```

```
cd your/working/directory
git clone https://github.com/GeoNode/geonode.git -b 3.2.x
```

2. Installation of GDAL

The gdal can be install through OSGeo4W. But this time we need to install it manually. This time we need to install the gdal inside our virtual environment.

1. Goto [Unofficial Windows Binaries for Python Extension Packages](#) and Download the compatible gdal wheel file

Nota: First of all check your python version (*python --version*). And download the same version of whl file. If you installed python 3.7, then you should download GDAL-2.4.1-cp37-cp37m-win_amd64.whl file. Here cp37 and amd64 means it is compatible with 64 bit python 3.7

2. Install gdal using this whl file

```
# Activate virtualenv
cd your/working/directory
.\venv\Scripts\activate.bat

# install gdal inside your virtualenv
pip install <path/to/gdal/wheel/file/GDAL-2.4.1-cp37-cp37m-win_amd64.whl>
```

3. Installation of required libraries and run locally

For installation of required libraries, you should follow following steps,

1. Edit requirement.txt file

```
cd your/working/directory
notepad requirement.txt
```

It will open the requirement.txt file in notepad. Change Shapely==1.7.0 to Shapely==1.6.3. Since we not gonna deploy geonode in windows, remove the production packages from requirement.txt file,

```
# production uWSGI==2.0.18 gunicorn==20.0.4 ipython==7.14.0 docker==4.2.0 invoke==1.4.1
```

2. Install the requirement.txt file

```
# Activate virtualenv
cd your/working/directory
.\venv\Scripts\activate.bat

# Install requirement.txt file inside virtualenv
pip install -r requirements.txt --upgrade --no-cache --no-cache-dir
pip install -e .
```

3. Run the geonode in DEBUG (DEVELOPMENT) mode

```
# Prepare the GeoNode Spatialite database (the first time only)
paver setup
paver sync
python manage.py runserver
```

Now the geonode will run on your windows.

4. Postgresql Database setup

In this section we are going to install setup PostgreSQL database on GeoNode. GeoNode uses the PostgreSQL 11 database.

1. Download and install the [postgres 11 windows installer](#)
2. After installation of PostgreSQL 11, open stack builder and install the spatial extension named as postgis

Aviso: Make sure you install the postgis extension from stack builder, otherwise it won't work.

3. Now it is time create database and add user

Aviso: Make sure you added postgresql to environment variable path. Otherwise psql will not be recognize in command prompt. Also you can search psql in windows and run the code directly from psql shell

```
# It will open the psql command line
psql -U postgres

# Create database named as geonode and geonode_data
CREATE DATABASE geonode;
CREATE DATABASE geonode_data;

# Create user named as geonode and password as geonode
CREATE USER geonode WITH ENCRYPTED PASSWORD 'geonode';

# Grant all the privileges of geonode and geonode_data database to user geonode
GRANT ALL PRIVILEGES ON DATABASE geonode TO geonode;
GRANT ALL PRIVILEGES ON DATABASE geonode_data TO geonode;
```

4. Change the pg_hba.conf file (C:\Program Files\PostgreSQL\11\data\pg_hba.conf) as below, so that you can access the database without password in your local machine

```
# "local" is for Unix domain socket connections only
# local    all             all                                     peer
local      all             all                                     trust
```

5. Restart the PostgreSQL to make the change effective

5. Update Django setting

Now it is time to connect the postgres database with django. You need to follow following steps,

1. Rename the file `local_settings.py.geoserver.sample` to `local_settings.py` (This file can be found at the `GEONODE_INSTALLATION_DIR/geonode/`)
2. Initialize the GeoNode with `local_settings.py` file.

```
# Initialize GeoNode
set DJANGO_SETTINGS_MODULE=geonode.local_settings paver reset
set DJANGO_SETTINGS_MODULE=geonode.local_settings paver setup
set DJANGO_SETTINGS_MODULE=geonode.local_settings paver sync
set DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py collectstatic --
↪noinput
```

Now you can run the geonode locally with postgresql database. using following command

```
python manage.py runserver --settings=geonode.local_settings
```

Docker

In this section we are going to list the passages needed to:

1. Install Docker and docker-compose packages on a Ubuntu host
2. Deploy a vanilla GeoNode 3.2.0 with Docker
 - a. Override the ENV variables to deploy on a public IP or domain
 - b. Access the `django4geonode` Docker image to update the code-base and/or change internal settings
 - c. Access the `geoserver4geonode` Docker image to update the GeoServer version
3. Passages to completely get rid of old Docker images and volumes (prune the environment completely)

1. Install the Docker and docker-compose packages on a Ubuntu host

Docker Setup (First time only)

```
sudo add-apt-repository universe
sudo apt-get update -y
sudo apt-get install -y git-core git-buildpackage debhelper devscripts
sudo apt-get install -y apt-transport-https ca-certificates curl gnupg-agent software-
↪properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
↪$(lsb_release -cs) stable"

sudo apt-get update -y
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-compose
sudo apt autoremove --purge

sudo usermod -aG docker geonode
su geonode
```

2. Install the Docker and docker-compose packages on a CentOS host

Docker Setup (First time only)

Aviso: The *centos-extras* repository must be enabled

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2

sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.
↳repo

sudo yum install docker-ce docker-ce-cli containerd.io

sudo systemctl start docker

sudo usermod -aG docker geonode
su geonode
```

3. Test Docker Compose Instance

Logout and login again on shell and then execute:

```
docker run -it hello-world
```

4. Deploy a vanilla GeoNode 3.2.0 with Docker

Clone the Project

```
# Let's create the GeoNode core base folder and clone it
sudo mkdir -p /opt/geonode/
sudo usermod -a -G www-data geonode
sudo chown -Rf geonode:www-data /opt/geonode/
sudo chmod -Rf 775 /opt/geonode/

# Clone the GeoNode source code on /opt/geonode
cd /opt
git clone https://github.com/GeoNode/geonode.git -b 3.2.x geonode
```

Start the Docker instances on localhost

Aviso: The first time pulling the images will take some time. You will need a good internet connection.

```
cd /opt/geonode
docker-compose -f docker-compose.yml pull
docker-compose -f docker-compose.yml up -d
```

Nota: If you want to re-build the docker images from scratch, instead of pulling them from the Docker Hub

add the `--build` parameter to the up command, for instance:

```
docker-compose -f docker-compose.yml up --build
```

In this **case** you can of course skip the `pull` step to download the `pre-built` images.

Nota: To startup the containers daemonized, which means they will be started in the background (and keep running if you log out from the server or close the shell) add the `-d` option to the up command as in the following. `docker-compose` will take care to restart the containers if necessary (e.g. after boot).

```
docker-compose -f docker-compose.yml up -d

# If you want to rebuild the images also
docker-compose -f docker-compose.yml up --build -d
```

Test the instance and follow the logs

If you run the containers daemonized (with the `-d` option), you can either run specific Docker commands to follow the startup and initialization logs or entering the image shell and check for the GeoNode logs.

In order to follow the startup and initialization logs, you will need to run the following command from the repository folder

```
cd /opt/geonode
docker logs -f django4geonode
```

Alternatively:

```
cd /opt/geonode
docker-compose logs -f django
```

You should be able to see several initialization messages. Once the container is up and running, you will see the following statements

```
...
789 static files copied to '/mnt/volumes/statics/static'.
static data refreshed
Executing UWSGI server uwsgi --ini /usr/src/app/uwsgi.ini for Production
[uWSGI] getting INI configuration from /usr/src/app/uwsgi.ini
```

To exit just hit CTRL+C.

This message means that the GeoNode containers have been started. Browsing to `http://localhost/` will show the GeoNode home page. You should be able to successfully log with the default admin user (admin / admin) and start using it right away.

With Docker it is also possible to run a shell in the container and follow the logs exactly the same as you deployed it on a physical host. To achieve this run

```
docker exec -it django4geonode /bin/bash

# Once logged in the GeoNode image, follow the logs by executing
tail -F -n 300 /var/log/geonode.log
```


Alternatively:

```
docker-compose exec django /bin/bash
```

To exit just hit CTRL+C and `exit` to return to the host.

Override the ENV variables to deploy on a public IP or domain

If you would like to start the containers on a public IP or domain, let's say `www.example.org`, you can

```
cd /opt/geonode

# Stop the Containers (if running)
docker-compose stop
```

Edit the ENV override file in order to deploy on `www.example.org`

```
# Make sure the new host is correctly configured on the ``.env`` file
vim .env
```

Replace everywhere `localhost` with `www.example.org`

```
vim docker-compose.override.example-org.yml
```

```
# e.g.: :%s/localhost/www.example.org/g

version: '2.2'
services:

  django:
    build: .
    # Loading the app is defined here to allow for
    # autoreload on changes it is mounted on top of the
    # old copy that docker added when creating the image
    volumes:
      - './usr/src/app'
    environment:
      - DEBUG=False
      - GEONODE_LB_HOST_IP=www.example.org
      - GEONODE_LB_PORT=80
      - SITEURL=http://www.example.org/
      - ALLOWED_HOSTS=['www.example.org', ]
      - GEOSERVER_PUBLIC_LOCATION=http://www.example.org/geoserver/
      - GEOSERVER_WEB_UI_LOCATION=http://www.example.org/geoserver/

  celery:
    build: .
    volumes:
      - './usr/src/app'
    environment:
      - DEBUG=False
      - GEONODE_LB_HOST_IP=www.example.org
      - GEONODE_LB_PORT=80
      - SITEURL=http://www.example.org/
      - ALLOWED_HOSTS=['www.example.org', ]
      - GEOSERVER_PUBLIC_LOCATION=http://www.example.org/geoserver/
```

(continues on next page)

(continuação da página anterior)

```
- GEOSERVER_WEB_UI_LOCATION=http://www.example.org/geoserver/

geoserver:
  environment:
    - GEONODE_LB_HOST_IP=www.example.org
    - GEONODE_LB_PORT=80
#   - NGINX_BASE_URL=
```

Nota: It is possible to override here even more variables to customize the GeoNode instance. See the GeoNode Settings section in order to get a list of the available options.

Run the containers in daemon mode

```
docker-compose -f docker-compose.yml -f docker-compose.override.example-org.yml up --
↪build -d
```

Access the django4geonode Docker container to update the code-base and/or change internal settings

Access the container bash

```
docker exec -i -t django4geonode /bin/bash
```

You will be logged into the GeoNode instance as `root`. The folder is `/usr/src/app/` where the GeoNode project is cloned. Here you will find the GeoNode source code as in the GitHub repository.

Nota: The machine is empty by default, no Ubuntu packages installed. If you need to install text editors or something you have to run the following commands:

```
apt update
apt install <package name>

e.g.:
  apt install vim
```

Update the templates or the Django models. Once in the bash you can edit the templates or the Django models/classes. From here you can run any standard Django management command.

Whenever you change a template/CSS/Javascript remember to run later:

```
python manage.py collectstatic
```

in order to update the files into the `statics` Docker volume.

Aviso: This is an external volume, and a simple restart won't update it. You have to be careful and keep it aligned with your changes.

Whenever you need to change some settings or environment variable, the easiest thing to do is to:

```
# Stop the container
docker-compose stop

# Restart the container in Daemon mode
docker-compose -f docker-compose.yml -f docker-compose.override.<whatever>.yaml up -d
```

Whenever you change the model, remember to run later in the container via bash:

```
python manage.py makemigrations
python manage.py migrate
```

Access the geoserver4geonode Docker container to update the GeoServer version

This procedure allows you to access the GeoServer container.

The concept is exactly the same as above, log into the container with bash.

```
# Access the container bash
docker exec -it geoserver4geonode /bin/bash
```

You will be logged into the GeoServer instance as `root`.

GeoServer is deployed on an Apache Tomcat instance which can be found here

```
cd /usr/local/tomcat/webapps/geoserver
```

Aviso: The GeoServer `DATA_DIR` is deployed on an external Docker Volume `geonode_gsdatadir`. This data dir won't be affected by changes to the GeoServer application since it is external.

Update the GeoServer instance inside the GeoServer Container

Aviso: The old configuration will be kept since it is external

```
docker exec -it geoserver4geonode bash
```

```
cd /usr/local/tomcat/
wget --no-check-certificate "https://www.dropbox.com/s/cmrdezdeloq67pre/geoserver-2.18.3.war?dl=1" -O geoserver-2.18.3.war
mkdir tmp/geoserver
cd tmp/geoserver/
unzip /usr/local/tomcat/geoserver-2.18.3.war
rm -Rf data
cp -Rf /usr/local/tomcat/webapps/geoserver/data/ .
cd /usr/local/tomcat/
mv webapps/geoserver/ .
mv tmp/geoserver/ webapps/
exit
```

```
docker restart geoserver4geonode
```

Aviso: GeoNode 2.8.1 is **NOT** compatible with GeoServer > 2.13.x
GeoNode 2.8.2 / 2.10.x are **NOT** compatible with GeoServer < 2.14.x
GeoNode 3.x is **NOT** compatible with GeoServer < 2.16.x

Remove all data and bring your running GeoNode deployment to the initial stage

This procedure allows you to stop all the containers and reset all the data with the deletion of all the volumes.

```
cd /opt/geonode
# stop containers and remove volumes
docker-compose down -v
```

5. Passages to completely get rid of old Docker images and volumes (reset the environment completely)

Nota: For more details on Docker commands, please refer to the official Docker documentation.

It is possible to let docker show which containers are currently running (add `-a` for all containers, also stopped ones)

```
# Show the currently running containers
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
→ STATUS	PORTS	NAMES	
3b232931f820	geonode/nginx:production	"nginx -g 'daemon of...'"	26 minutes
→ago Up 26 minutes	0.0.0.0:80->80/tcp	nginx4geonode	
ff7002ae6e91	geonode/geonode:latest	"/usr/src/app/entryp..."	26 minutes
→ago Up 26 minutes	8000/tcp	django4geonode	
2f155e5043be	geonode/geoserver:2.18.3	"/usr/local/tomcat/t..."	26 minutes
→ago Up 26 minutes	8080/tcp	geoserver4geonode	
97f1668a01b1	geonode_celery	"/usr/src/app/entryp..."	26 minutes
→ago Up 26 minutes	8000/tcp	geonode_celery_1	
1b623598b1bd	geonode/postgis:10	"docker-entrypoint.s..."	About an
→hour ago Up 26 minutes	5432/tcp	db4geonode	

Stop all the containers by running

```
docker-compose stop
```

Force kill all containers by running

```
docker kill $(docker ps -q)
```

If you want to clean up all containers and images, without deleting the static volumes (i.e. the DB and the GeoServer catalog), issue the following commands

```
# Remove all containers
docker rm $(docker ps -a -q)
```

(continues on next page)

(continuação da página anterior)

```
# Remove all docker images
docker rmi $(docker images -q)

# Prune the old images
docker system prune -a
```

If you want to remove a volume also

```
# List of the running volumes
docker volume ls

# Remove the GeoServer catalog by its name
docker volume rm -f geonode-gsdatadir

# Remove all dangling docker volumes
docker volume rm $(docker volume ls -qf dangling=true)

# update all images, should be run regularly to fetch published updates
for i in $(docker images| awk 'NR>1{print $1":"$2}'| grep -v '<none>'); do docker_
↳pull "$i" ;done
```

1.12.2 GeoNode Project

Overview

The following steps will guide you to a new setup of GeoNode Project. All guides will first install and configure the system to run it in `DEBUG` mode (also known as `DEVELOPMENT` mode) and then by configuring an `HTTPD` server to serve GeoNode through the standard `HTTP` (80) port.

Those guides **are not** meant to be used on a production system. There will be dedicated chapters that will show you some *hints* to optimize GeoNode for a production-ready machine. In any case, we strongly suggest to task an experienced *DevOp* or *System Administrator* before exposing your server to the `WEB`.

Ubuntu 18.04

This part of the documentation describes the complete setup process for GeoNode on an Ubuntu 18.04 64-bit clean environment (Desktop or Server). All examples use shell commands that you must enter on a local terminal or a remote shell. - If you have a graphical desktop environment you can open the terminal application after login; - if you are working on a remote server the provider or sysadmin should have given you access through an `ssh` client.

Install the dependencies

In this section, we are going to install all the basic packages and tools needed for a complete GeoNode installation. To follow this guide, a piece of basic knowledge about Ubuntu Server configuration and working with a shell is required. This guide uses `vim` as the editor; feel free to use `nano`, `gedit` or others.

Upgrade system packages

Check that your system is already up-to-date with the repository running the following commands:

```
sudo apt update
sudo apt upgrade
```

Create a Dedicated User

In the following steps a User named geonode is used: to run installation commands the user must be in the `sudo` group.

Create User geonode **if not present**:

```
# Follow the prompts to set the new user's information.
# It is fine to accept the defaults to leave all of this information blank.
sudo adduser geonode

# The following command adds the user geonode to group sudo
sudo usermod -aG sudo geonode

# make sure the newly created user is allowed to login by ssh
# (out of the scope of this documentation) and switch to User geonode
su geonode
```

Packages Installation

Nota: You don't need to install the **system packages** if you want to run the project using Docker

First, we are going to install all the **system packages** needed for the GeoNode setup.

```
# Install packages from GeoNode core
sudo apt install -y gdal-bin
sudo apt install -y python3-pip python3-dev python3-virtualenv python3-venv
↪virtualenvwrapper
sudo apt install -y libxml2 libxml2-dev gettext
sudo apt install -y libxslt1-dev libjpeg-dev libpng-dev libpq-dev libgdal-dev
sudo apt install -y software-properties-common build-essential
sudo apt install -y git unzip gcc zlib1g-dev libgeos-dev libproj-dev
sudo apt install -y sqlite3 spatialite-bin libsqlite3-mod-spatialite

# If the following does not work, you can skip it
sudo apt install -y libgdal20

# Install Openjdk
sudo -i apt update
sudo apt install openjdk-8-jdk-headless default-jdk-headless -y
sudo update-java-alternatives --jre-headless --jre --set java-1.8.0-openjdk-amd64

sudo apt update -y
sudo apt upgrade -y
sudo apt autoremove -y
```

(continues on next page)

(continuação da página anterior)

```

sudo apt autoclean -y
sudo apt purge -y
sudo apt clean -y

# Install Packages for Virtual environment management
sudo apt install -y virtualenv virtualenvwrapper

# Install text editor
sudo apt install -y vim

```

Geonode Project Installation

Geonode project is the proper way to run a customized installation of Geonode. The repository of geonode-project contains a minimal set of files following the structure of a django-project. Geonode itself will be installed as a requirement of your project. Inside the project structure is possible to extend, replace or modify all geonode componentse (e.g. css and other static files, templates, models..) and even register new django apps **without touching the original Geonode code**.

Nota: You can call your geonode project whatever you like following the naming conventions for python packages (generally lower case with underscores (_)). In the examples below, replace `my_geonode` with whatever you would like to name your project.

See also the [README](#) file on geonode-project repository

First of all we need to prepare a new Python Virtual Environment

Prepare the environment

```

sudo mkdir -p /opt/geonode_custom/
sudo usermod -a -G www-data geonode
sudo chown -Rf geonode:www-data /opt/geonode_custom/
sudo chmod -Rf 775 /opt/geonode_custom/

```

Clone the source code

```

cd /opt/geonode_custom/
git clone https://github.com/GeoNode/geonode-project.git -b 3.2.x

```

Make an instance out of the Django Template

Nota: We will call our instance `my_geonode`. You can change the name at your convenience.

```

vim ~/.bashrc
# add the following line to the bottom
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh

```

```

source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
mkvirtualenv --python=/usr/bin/python3 my_geonode

```

Alterantively you can also create the virtual env like below

```
python3.8 -m venv /home/geonode/dev/.venvs/my_geonode
```

(continues on next page)

(continuação da página anterior)

```
source /home/geonode/dev/.venvs/my_geonode/bin/activate

pip install Django==3.2

django-admin startproject --template=./geonode-project -e py,sh,md,rst,json,yml,ini,
↪env,sample,properties -n monitoring-cron -n Dockerfile my_geonode

# Install the Python packages
cd /opt/geonode_custom/my_geonode
pip install -r requirements.txt --upgrade --no-cache --no-cache-dir
pip install -e . --upgrade

# Install GDAL Utilities for Python
pip install pygdal=="`gdal-config --version`.*"

# Dev scripts
mv .override_dev_env.sample .override_dev_env
mv manage_dev.sh.sample manage_dev.sh
mv paver_dev.sh.sample paver_dev.sh
```

Install and Configure the PostgreSQL Database System

In this section we are going to install the PostgreSQL packages along with the PostGIS extension. Those steps must be done **only** if you don't have the DB already installed on your system.

```
# Ubuntu 18.04
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg,
↪main" >> /etc/apt/sources.list.d/pgdg.list'
sudo wget --no-check-certificate --quiet -O - https://www.postgresql.org/media/keys/
↪ACCC4CF8.asc | sudo apt-key add -
sudo apt update -y; sudo apt install -y postgresql-13 postgresql-13-postgis-3
↪postgresql-13-postgis-3-scripts postgresql-13 postgresql-client-13
```

We now must create two databases, `my_geonode` and `my_geonode_data`, belonging to the role `my_geonode`.

Aviso: This is our default configuration. You can use any database or role you need. The connection parameters must be correctly configured on settings, as we will see later in this section.

Databases and Permissions

First, create the geonode user. GeoNode is going to use this user to access the database

```
sudo service postgresql start
sudo -u postgres createuser -P my_geonode

# Use the password: geonode
```

You will be prompted asked to set a password for the user. **Enter geonode as password.**

Aviso: This is a sample password used for the sake of simplicity. This password is very **weak** and should be changed in a production environment.

Create database `my_geonode` and `my_geonode_data` with owner `my_geonode`

```
sudo -u postgres createdb -O my_geonode my_geonode
sudo -u postgres createdb -O my_geonode my_geonode_data
```

Next let's create PostGIS extensions

```
sudo -u postgres psql -d my_geonode -c 'CREATE EXTENSION postgis;'
sudo -u postgres psql -d my_geonode -c 'GRANT ALL ON geometry_columns TO PUBLIC;'
sudo -u postgres psql -d my_geonode -c 'GRANT ALL ON spatial_ref_sys TO PUBLIC;'
sudo -u postgres psql -d my_geonode -c 'GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA_
↳public TO my_geonode;'

sudo -u postgres psql -d my_geonode_data -c 'CREATE EXTENSION postgis;'
sudo -u postgres psql -d my_geonode_data -c 'GRANT ALL ON geometry_columns TO PUBLIC;'
sudo -u postgres psql -d my_geonode_data -c 'GRANT ALL ON spatial_ref_sys TO PUBLIC;'
sudo -u postgres psql -d my_geonode_data -c 'GRANT ALL PRIVILEGES ON ALL TABLES IN_
↳SCHEMA public TO my_geonode;'
```

Final step is to change user access policies for local connections in the file `pg_hba.conf`

```
sudo vim /etc/postgresql/13/main/pg_hba.conf
```

Scroll down to the bottom of the document. We want to make local connection trusted for the default user.

Make sure your configuration looks like the one below.

```
...
# DO NOT DISABLE!
# If you change this first entry you will need to make sure that the
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local    all             postgres                                trust

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local    all             all                                md5
# IPv4 local connections:
host     all             all             127.0.0.1/32            md5
# IPv6 local connections:
host     all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local    replication     all                                peer
host     replication     all             127.0.0.1/32            md5
host     replication     all             ::1/128                 md5
```

Aviso: If your PostgreSQL database resides on a **separate/remote machine**, you'll have to **allow** remote access to the databases in the `/etc/postgresql/13/main/pg_hba.conf` to the geonode user and tell PostgreSQL to **accept** non-local connections in your `/etc/postgresql/13/main/postgresql.conf` file

Restart PostgreSQL to make the change effective.

```
sudo service postgresql restart
```

PostgreSQL is now ready. To test the configuration, try to connect to the geonode database as geonode role.

```
psql -U postgres my_geonode
# This should not ask for any password

psql -U my_geonode my_geonode
# This should ask for the password geonode

# Repeat the test with geonode_data DB
psql -U postgres my_geonode_data
psql -U my_geonode my_geonode_data
```

Run GeoNode Project for the first time in DEBUG Mode

Aviso: Be sure you have successfully completed all the steps of the section *Install the dependencies*.

This command will run both GeoNode and GeoServer locally after having prepared the Spatialite database. The server will start in DEBUG (or DEVELOPMENT) mode, and it will start the following services:

1. GeoNode on `http://localhost:8000/`
2. GeoServer on `http://localhost:8080/geoserver/`

This modality is beneficial to debug issues and/or develop new features, but it cannot be used on a production system.

```
# Prepare the GeoNode Spatialite database (the first time only)
./paver_dev.sh setup
./paver_dev.sh sync
```

Nota: In case you want to start again from a clean situation, just run

```
./paver_dev.sh reset_hard
```

Aviso: This will blow up completely your `local_settings`, delete the SQLite database and remove the GeoServer data dir.

```
# Run the server in DEBUG mode
./paver_dev.sh start
```

Once the server has finished the initialization and prints on the console the sentence `GeoNode is now available.`, you can open a browser and go to:

```
http://localhost:8000/
```

Sign-in with:

```
user: admin
password: admin
```

From now on, everything already said for GeoNode Core (please refer to the section [3. Postgis database Setup](#) and following), applies to a GeoNode Project.

Be careful to use the **new** paths and names everywhere:

- Everytime you'll find the keyword `geonode`, you'll need to use your geonode custom name instead (in this example `my_geonode`).
- Everytime you'll find paths pointing to `/opt/geonode/`, you'll need to update them to point to your custom project instead (in this example `/opt/geonode_custom/my_geonode`).

Docker

Aviso: Before moving with this section, you should have read and clearly understood the `INSTALLATION > GeoNode Core` sections, and in particular the `Docker` one. Everything said for the `GeoNode Core Vanilla` applies here too, except that the `Docker` container names will be slightly different. As an instance if you named your project `my_geonode`, your containers will be called:

```
'django4my_geonode' instead of 'django4geonode' and so on...
```

Deploy an instance of a geonode-project Django template 3.2.0 with Docker on localhost

Prepare the environment

```
sudo mkdir -p /opt/geonode_custom/
sudo usermod -a -G www-data geonode
sudo chown -Rf geonode:www-data /opt/geonode_custom/
sudo chmod -Rf 775 /opt/geonode_custom/
```

Clone the source code

```
cd /opt/geonode_custom/
git clone https://github.com/GeoNode/geonode-project.git -b 3.2.x
```

Make an instance out of the Django Template

Nota: We will call our instance `my_geonode`. You can change the name at your convenience.

```
source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
mkvirtualenv --python=/usr/bin/python3 my_geonode
```

Alterantively you can also create the virtual env like below

(continues on next page)

(continuação da página anterior)

```
python3.8 -m venv /home/geonode/dev/.venvs/my_geonode
source /home/geonode/dev/.venvs/my_geonode/bin/activate

pip install Django==3.2

django-admin startproject --template=./geonode-project -e py,sh,md,rst,json,yml,ini,
↪env,sample,properties -n monitoring-cron -n Dockerfile my_geonode
cd /opt/geonode_custom/my_geonode
```

Modify the code and the templates and rebuild the Docker Containers

```
docker-compose -f docker-compose.yml build --no-cache
```

Finally, run the containers

```
docker-compose -f docker-compose.yml up -d
```

Deploy an instance of a geonode-project Django template 3.2.0 with Docker on a domain

Nota: We will use `www.example.org` as an example. You can change the name at your convenience.

Stop the containers

```
cd /opt/geonode_custom/my_geonode

docker-compose -f docker-compose.yml stop
```

Edit the ENV override file in order to deploy on `www.example.org`

Replace everywhere `localhost` with `www.example.org`

```
vim .env
```

```
# e.g.: :%s/localhost/www.example.org/g
```

Nota: It is possible to override here even more variables to customize the GeoNode instance. See the GeoNode Settings section in order to get a list of the available options.

Run the containers in daemon mode

```
docker-compose -f docker-compose.yml -f docker-compose.override.example-org.yml up --
↪build -d
```

1.13 GeoNode Settings

Settings

1.13.1 Settings

Here's a list of settings available in GeoNode and their default values. This includes settings for some external applications that GeoNode depends on.

For most of them, default values are good. Those should be changed only for advanced configurations in production or heavily hardened systems.

The most common ones can be set through `environment` variables to avoid touching the `settings.py` file at all. This is a good practice and also the preferred one to configure GeoNode (and Django apps in general). Whenever you need to change them, set the environment variable accordingly (where it is available) instead of overriding it through the `local_settings`.

A

ACCESS_TOKEN_EXPIRE_SECONDS

Default: 86400

Env: ACCESS_TOKEN_EXPIRE_SECONDS

When a user logs into GeoNode, if no ACCESS_TOKEN exists, a new one will be created with a default expiration time of ACCESS_TOKEN_EXPIRE_SECONDS seconds (1 day by default).

ACCOUNT_ADAPTER

Default: `geonode.people.adapters.LocalAccountAdapter`

Custom GeoNode People (Users) Account Adapter.

ACCOUNT_APPROVAL_REQUIRED

Default: `False`

Env: ACCOUNT_APPROVAL_REQUIRED

If ACCOUNT_APPROVAL_REQUIRED equals `True`, newly registered users must be activated by a superuser through the Admin gui, before they can access GeoNode.

ACCOUNT_CONFIRM_EMAIL_ON_GET

Default: `True`

This is a [django-allauth setting](#) It allows specifying the HTTP method used when confirming e-mail addresses.

ACCOUNT_EMAIL_REQUIRED

Default: `True`

This is a [django-allauth setting](#) which controls whether the user is required to provide an e-mail address upon registration.

ACCOUNT_EMAIL_VERIFICATION

Default: `optional`

This is a [django-allauth setting](#)

ACCOUNT_LOGIN_REDIRECT_URL

Default: `SITEURL`

Env: `LOGIN_REDIRECT_URL`

This is a [django-user-accounts setting](#) It allows specifying the default redirect URL after a successful login.

ACCOUNT_LOGOUT_REDIRECT_URL

Default: `SITEURL`

Env: `LOGOUT_REDIRECT_URL`

This is a [django-user-accounts setting](#) It allows specifying the default redirect URL after a successful logout.

ACCOUNT_NOTIFY_ON_PASSWORD_CHANGE

Default: `True`

Env: `ACCOUNT_NOTIFY_ON_PASSWORD_CHANGE`

This is a [django-user-accounts setting](#)

ACCOUNT_OPEN_SIGNUP

Default: `True`

Env: `ACCOUNT_OPEN_SIGNUP`

This is a [django-user-accounts setting](#) Whether or not people are allowed to self-register to GeoNode or not.

ACCOUNT_SIGNUP_FORM_CLASS

Default: `geonode.people.forms.AllauthReCaptchaSignupForm`

Env: `ACCOUNT_SIGNUP_FORM_CLASS`

Enabled only when the `RECAPTCHA_ENABLED` option is `True`.

Ref. to `RECAPTCHA_ENABLED`

ACTSTREAM_SETTINGS

Default:

```
{
  'FETCH_RELATIONS': True,
  'USE_PREFETCH': False,
  'USE_JSONFIELD': True,
  'GFK_FETCH_DEPTH': 1,
}
```

Actstream Settings.

ADMIN_MODERATE_UPLOADS

Default: `False`

When this variable is set to `True`, every uploaded resource must be approved before becoming visible to the public users.

Until a resource is in `PENDING APPROVAL` state, only the superusers, owner and group members can access it, unless specific edit permissions have been set for other users or groups.

A Group Manager *can* approve the resource, but he cannot publish it whenever the setting `RESOURCE_PUBLISHING` is set to `True`. Otherwise, if `RESOURCE_PUBLISHING` is set to `False`, the resource becomes accessible as soon as it is approved.

ADMINS_ONLY_NOTICE_TYPES

Default: `['monitoring_alert',]`

A list of notification labels that standard users should not either see or set.

Such notifications will be hidden from the notify settings page and automatically set to `false` for non-superusers.

ADVANCED_EDIT_EXCLUDE_FIELD

Default: []

A list of element (item name) to exclude from the Advanced Edit page.

Example:

```
ADVANCED_EDIT_EXCLUDE_FIELD=['title', 'keywords', 'tkeywords']
```

AGON_RATINGS_CATEGORY_CHOICES

Default:

```
{
  "maps.Map": {
    "map": "How good is this map?"
  },
  "layers.Layer": {
    "layer": "How good is this layer?"
  },
  "documents.Document": {
    "document": "How good is this document?"
  }
}
```

ALLOWED_DOCUMENT_TYPES

Default:

```
['doc', 'docx', 'gif', 'jpg', 'jpeg', 'ods', 'odt', 'odp', 'pdf', 'png',
'ppt', 'pptx', 'rar', 'sld', 'tif', 'tiff', 'txt', 'xls', 'xlsx', 'xml',
'zip', 'gz', 'qml']
```

A list of acceptable file extensions that can be uploaded to the Documents app.

ANONYMOUS_USER_ID

Default: -1

Env: ANONYMOUS_USER_ID

The id of an anonymous user. This is an django-guardian setting.

API_INCLUDE_REGIONS_COUNT

Default: False

Env: API_INCLUDE_REGIONS_COUNT

If set to True, a counter with the total number of available regions will be added to the API JSON Serializer.

API_LIMIT_PER_PAGE

Default: 200

Env: API_LIMIT_PER_PAGE

The Number of items returned by the APIs 0 equals no limit. Different from CLIENT_RESULTS_LIMIT, affecting the number of items per page in the resource list.

API_LOCKDOWN

Default: True

Env: API_LOCKDOWN

If this is set to True users must be authenticated to get search results when search for for users, groups, categories, regions, tags etc. Filtering search results of Resourcebase-objects like Layers, Maps or Documents by one of the above types does not work. Attention: If API_LOCKDOWN is set to False all details can be accessed by anonymous users.

ASYNC_SIGNALS

Default: False

Env: ACCOUNT_NOTIFY_ON_PASSWORD_CHANGE

AUTH_EXEMPT_URLS

Default:

```
(r'^/?$',
'/gs/*',
'/static/*',
'/o/*',
'/api/o/*',
'/api/roles',
'/api/adminRole',
'/api/users',
'/api/layers',)
```

A tuple of URL patterns that the user can visit without being authenticated. This setting has no effect if LOCKDOWN_GEONODE is not True. For example, AUTH_EXEMPT_URLS = ('/maps',) will allow unauthenticated users to browse maps.

AUTO_ASSIGN_REGISTERED_MEMBERS_TO_REGISTERED_MEMBERS_GROUP_NAME

Default: True

Env: AUTO_ASSIGN_REGISTERED_MEMBERS_TO_REGISTERED_MEMBERS_GROUP_NAME

Auto assign users to a default REGISTERED_MEMBERS_GROUP_NAME private group after AUTO_ASSIGN_REGISTERED_MEMBERS_TO_REGISTERED_MEMBERS_GROUP_AT.

AUTO_ASSIGN_REGISTERED_MEMBERS_TO_REGISTERED_MEMBERS_GROUP_AT

Default: `activation`

Env: `AUTO_ASSIGN_REGISTERED_MEMBERS_TO_REGISTERED_MEMBERS_GROUP_AT`

Options: `"registration" | "activation" | "login"`

Auto assign users to a default `REGISTERED_MEMBERS_GROUP_NAME` private group after {«registration» | «activation» | «login»}.

Notice that whenever `ACCOUNT_EMAIL_VERIFICATION == True` and `ACCOUNT_APPROVAL_REQUIRED == False`, users will be able to register and they became active already, even if they won't be able to login until the email has been verified.

AUTO_GENERATE_AVATAR_SIZES

Default: `20, 30, 32, 40, 50, 65, 70, 80, 100, 140, 200, 240`

An iterable of integers representing the sizes of avatars to generate on upload. This can save rendering time later on if you pre-generate the resized versions.

AVATAR_GRAVATAR_SSL

Default: `False`

Env: `AVATAR_GRAVATAR_SSL`

Options: `True | False`

Force SSL when loading fallback image from gravatar.com.

AVATAR_DEFAULT_URL

Default: `/geonode/img/avatar.png`

Env: `AVATAR_GRAVATAR_SSL`

Options: `"filepath to image"`

Allows to set a custom fallback image in case a User has not uploaded a profile image. Needs `AVATAR_PROVIDERS` to be set correctly.

AVATAR_PROVIDERS

Default:

```
'avatar.providers.PrimaryAvatarProvider', 'avatar.providers.  
↪GravatarAvatarProvider', 'avatar.providers.DefaultAvatarProvider'
```

Env: `AVATAR_PROVIDERS`

Options: Avatar provider object

This setting configures in which order gravatar images are loaded. A common use case is the use of a local image over a fallback image loaded from gravatar.com. To do so you would change the order like:

```
'avatar.providers.PrimaryAvatarProvider', 'avatar.providers.
↪DefaultAvatarProvider', 'avatar.providers.GravatarAvatarProvider'
```

(DefaultAvatarProvider before GravatarAvatarProvider)

AWS_ACCESS_KEY_ID

Default: ''

Env: AWS_ACCESS_KEY_ID

This is a [Django storage setting](#) Your Amazon Web Services access key, as a string.

Aviso: This works only if `DEBUG = False`

AWS_BUCKET_NAME

Default: ''

Env: S3_BUCKET_NAME

The name of the S3 bucket GeoNode will pull static and/or media files from. Set through the environment variable S3_BUCKET_NAME. This is a [Django storage setting](#)

Aviso: This works only if `DEBUG = False`

AWS_QUERYSTRING_AUTH

Default: False

This is a [Django storage setting](#) Setting AWS_QUERYSTRING_AUTH to False to remove query parameter authentication from generated URLs. This can be useful if your S3 buckets are public.

Aviso: This works only if `DEBUG = False`

AWS_S3_BUCKET_DOMAIN

<https://github.com/GeoNode/geonode/blob/master/geonode/settings.py#L1661>

```
AWS_S3_BUCKET_DOMAIN = '%s.s3.amazonaws.com' % AWS_STORAGE_BUCKET_NAME
```

Aviso: This works only if `DEBUG = False`

AWS_SECRET_ACCESS_KEY

Default: ''

Env: AWS_SECRET_ACCESS_KEY

This is a [Django storage setting](#) Your Amazon Web Services secret access key, as a string.

Aviso: This works only if `DEBUG = False`

AWS_STORAGE_BUCKET_NAME

Default: ''

Env: S3_BUCKET_NAME

This is a [Django storage setting](#) Your Amazon Web Services storage bucket name, as a string.

Aviso: This works only if `DEBUG = False`

B

BING_API_KEY

Default: None

Env: BING_API_KEY

This property allows to enable a Bing Aerial background.

If using `mapstore` client library, make sure the `MAPSTORE_BASELAYERS` include the following:

```
if BING_API_KEY:
    BASEMAP = {
        "type": "bing",
        "title": "Bing Aerial",
        "name": "AerialWithLabels",
        "source": "bing",
        "group": "background",
        "apiKey": "{{apiKey}}",
        "visibility": False
    }
    DEFAULT_MS2_BACKGROUNDS = [BASEMAP,] + DEFAULT_MS2_BACKGROUNDS
```

BROKER_HEARTBEAT

Default: 0

Heartbeats are used both by the client and the broker to detect if a connection was closed. This is a [Celery setting](#).

BROKER_TRANSPORT_OPTIONS

Default:

```
{
  'fanout_prefix': True,
  'fanout_patterns': True,
  'socket_timeout': 60,
  'visibility_timeout': 86400
}
```

This is a [Celery setting](#).

C

CACHES

Default:

```
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.dummy.DummyCache',
    },
    'resources': {
        'BACKEND': 'django.core.cache.backends.locmem.LocMemCache',
        'TIMEOUT': 600,
        'OPTIONS': {
            'MAX_ENTRIES': 10000
        }
    }
}
```

A dictionary containing the settings for all caches to be used with Django. This is a [Django setting](#)

The 'default' cache is disabled because we don't have a mechanism to discriminate between client sessions right now, and we don't want all users fetch the same api results.

The 'resources' is not currently used. It might be helpful for [caching Django template fragments](#) and/or [Tastypie API Caching](#).

CACHE_BUSTING_MEDIA_ENABLED

Default: False

Env: CACHE_BUSTING_MEDIA_ENABLED

This is a [Django ManifestStaticFilesStorage](#) storage setting A boolean allowing you to enable the ManifestStaticFilesStorage storage. This works only on a production system.

Aviso: This works only if `DEBUG = False`

CACHE_BUSTING_STATIC_ENABLED

Default: False

Env: CACHE_BUSTING_STATIC_ENABLED

This is a [Django ManifestStaticFilesStorage](#) storage setting A boolean allowing you to enable the ManifestStaticFilesStorage storage. This works only on a production system.

Aviso: This works only if `DEBUG = False`

CASCADE_WORKSPACE

Default: geonode

Env: CASCADE_WORKSPACE

CATALOGUE

A dict with the following keys:

ENGINE: The CSW backend (default is `geonode.catalogue.backends.pycsw_local`) URL: The FULLY QUALIFIED base URL to the CSW instance for this GeoNode USERNAME: login credentials (if required) PASSWORD: login credentials (if required)

pycsw is the default CSW enabled in GeoNode. pycsw configuration directives are managed in the PYCSW entry.

CATALOGUE_METADATA_TEMPLATE

Default: `catalogue/full_metadata.xml`

A string with the catalogue xml file needed for the metadata.

CATALOGUE_METADATA_XSL

Default: `'/static/metadatatxsl/metadata.xsl'`

A string pointing to the XSL used to transform the metadata XML into human readable HTML.

CELERYD_POOL_RESTARTS

Default: `True`

This is a [Celery setting](#).

CELERY_ACCEPT_CONTENT

Default: `['json']`

This is a [Celery setting](#).

CELERY_ACKS_LATE

Default: `True`

This is a [Celery setting](#)

CELERY_BEAT_SCHEDULE

Here you can define your scheduled task.

CELERY_DISABLE_RATE_LIMITS

Default: `False`

This is a [Celery setting](#).

CELERY_ENABLE_UTC

Default: `True`

This is a [Celery setting](#).

CELERY_MAX_CACHED_RESULTS

Default: `32768`

This is a [Celery setting](#).

CELERY_MESSAGE_COMPRESSION

Default: `gzip`

This is a [Celery setting](#).

CELERY_RESULT_PERSISTENT

Default: `False`

This is a [Celery setting](#).

CELERY_RESULT_SERIALIZER

Default: `json`

This is a [Celery setting](#).

CELERY_SEND_TASK_SENT_EVENT

Default: `True`

If enabled, a task-sent event will be sent for every task so tasks can be tracked before they are consumed by a worker. This is a [Celery setting](#).

CELERY_TASK_ALWAYS_EAGER

Default: `False` if `ASYNC_SIGNALS` else `True`

This is a [Celery setting](#).

CELERY_TASK_CREATE_MISSING_QUEUES

Default: `True`

This is a [Celery setting](#).

CELERY_TASK_IGNORE_RESULT

Default: `True`

This is a [Celery setting](#).

CELERY_TASK_QUEUES

Default:

```
Queue('default', GEONODE_EXCHANGE, routing_key='default'),  
Queue('geonode', GEONODE_EXCHANGE, routing_key='geonode'),  
Queue('update', GEONODE_EXCHANGE, routing_key='update'),  
Queue('cleanup', GEONODE_EXCHANGE, routing_key='cleanup'),  
Queue('email', GEONODE_EXCHANGE, routing_key='email'),
```

A tuple with registered Queues.

CELERY_TASK_RESULT_EXPIRES

Default: 43200

Env: CELERY_TASK_RESULT_EXPIRES

This is a [Celery setting](#).

CELERY_TASK_SERIALIZER

Default: json Env: CELERY_TASK_SERIALIZER

This is a [Celery setting](#).

CELERY_TIMEZONE

Default: UTC

Env: TIME_ZONE

This is a [Celery setting](#).

CELERY_TRACK_STARTED

Default: True

This is a [Celery setting](#).

CELERY_WORKER_DISABLE_RATE_LIMITS

Default: False

Disable the worker rate limits (number of tasks that can be run in a given time frame).

CELERY_WORKER_SEND_TASK_EVENTS

Default: `False`

Send events so the worker can be monitored by other tools.

CLIENT_RESULTS_LIMIT

Default: `5`

Env: `CLIENT_RESULTS_LIMIT`

The Number of results per page listed in the GeoNode search pages. Different from `API_LIMIT_PER_PAGE`, affecting the number of items returned by the APIs.

CREATE_LAYER

Default: `False`

Env: `CREATE_LAYER`

Enable the create layer plugin.

CKAN_ORIGINS

Default:

```
CKAN_ORIGINS = [{
    "label": "Humanitarian Data Exchange (HDX)",
    "url": "https://data.hdx.rwlab.org/dataset/new?title={name}&notes=
→ {abstract}",
    "css_class": "hdx"
}]
```

A list of dictionaries that are used to generate the links to CKAN instances displayed in the Share tab. For each origin, the name and abstract format parameters are replaced by the actual values of the Resource-Base object (layer, map, document). This is not enabled by default. To enable, uncomment the following line: `SOCIAL_ORIGINS.extend(CKAN_ORIGINS)`.

CSRF_COOKIE_HTTPONLY

Default: `False`

Env: `CSRF_COOKIE_HTTPONLY`

Whether to use `HttpOnly` flag on the CSRF cookie. If this is set to `True`, client-side JavaScript will not be able to access the CSRF cookie. This is a [Django Setting](#)

CSRF_COOKIE_SECURE

Default: `False`

Env: `CSRF_COOKIE_SECURE`

Whether to use a secure cookie for the CSRF cookie. If this is set to `True`, the cookie will be marked as “secure,” which means browsers may ensure that the cookie is only sent with an HTTPS connection. This is a [Django Setting](#)

D

DATA_UPLOAD_MAX_NUMBER_FIELDS

Default: `100000`

Maximum value of parsed attributes.

DEBUG

Default: `False`

Env: `DEBUG`

One of the main features of debug mode is the display of detailed error pages. If your app raises an exception when `DEBUG` is `True`, Django will display a detailed traceback, including a lot of metadata about your environment, such as all the currently defined Django settings (from `settings.py`). This is a [Django Setting](#)

DEBUG_STATIC

Default: `False`

Env: `DEBUG_STATIC`

Load non minified version of static files.

DEFAULT_ANONYMOUS_DOWNLOAD_PERMISSION

Default: `True`

Whether the uploaded resources should be downloadable by default.

Default: `True`

Whether the uploaded resources should be public by default.

DEFAULT_AUTO_FIELD

Default: `django.db.models.AutoField`

Default primary key field type to use for models that don't have a field with `primary_key=True`. Django documentation https://docs.djangoproject.com/it/3.2/ref/settings/#std:setting-DEFAULT_AUTO_FIELD

DEFAULT_LAYER_FORMAT

Default: `image/png`

Env: `DEFAULT_LAYER_FORMAT`

The default format for requested tile images.

DEFAULT_MAP_CENTER

Default: `(0, 0)`

Env: `DEFAULT_MAP_CENTER_X` `DEFAULT_MAP_CENTER_Y`

A 2-tuple with the latitude/longitude coordinates of the center-point to use in newly created maps.

DEFAULT_MAP_CRS

Default: `EPSG:3857`

Env: `DEFAULT_MAP_CRS`

The default map projection. Default: `EPSG:3857`

DEFAULT_MAP_ZOOM

Default: `0`

Env: `DEFAULT_MAP_ZOOM`

The zoom-level to use in newly created maps. This works like the OpenLayers zoom level setting; 0 is at the world extent and each additional level cuts the viewport in half in each direction.

DEFAULT_SEARCH_SIZE

Default: `10`

Env: `DEFAULT_SEARCH_SIZE`

An integer that specifies the default search size when using `geonode.search` for querying data.

DEFAULT_WORKSPACE

Default: `geonode`

Env: `DEFAULT_WORKSPACE`

The standard GeoServer workspace.

DELAYED_SECURITY_SIGNALS

Default: `False`

Env: `DELAYED_SECURITY_SIGNALS`

This setting only works when `GEOFENCE_SECURITY_ENABLED` has been set to `True` and GeoNode is making use of the GeoServer `BACKEND`.

By setting this to `True`, every time the permissions will be updated/changed for a Layer, they won't be applied immediately but only and only if either:

- a. A Celery Worker is running and it is able to execute the `geonode.security.tasks.sync_guardian` periodic task; notice that the task will be executed at regular intervals, based on the interval value defined in the corresponding `PeriodicTask` model.
- b. A periodic `cron` job runs the `sync_security_rules` management command, or either it is manually executed from the Django shell.
- c. The user, owner of the Layer or with rights to change its permissions, clicks on the GeoNode UI button `Sync permissions` immediately

Aviso: Layers won't be accessible to public users anymore until the Security Rules are not synchronized!

DISPLAY_COMMENTS

Default: `True`

Env: `DISPLAY_COMMENTS`

If set to `False` comments are hidden.

DISPLAY_RATINGS

Default: `True`

Env: `DISPLAY_RATINGS`

If set to `False` ratings are hidden.

DISPLAY_SOCIAL

Default: True

Env: DISPLAY_SOCIAL

If set to False social sharing is hidden.

DISPLAY_WMS_LINKS

Default: True

Env: DISPLAY_WMS_LINKS

If set to False direct WMS link to GeoServer is hidden.

DISPLAY_ORIGINAL_DATASET_LINK

Default: True

Env: DISPLAY_ORIGINAL_DATASET_LINK

If set to False original dataset download is hidden.

DOWNLOAD_FORMATS_METADATA

Specifies which metadata formats are available for users to download.

Default:

```
DOWNLOAD_FORMATS_METADATA = [  
    'Atom', 'DIF', 'Dublin Core', 'ebRIM', 'FGDC', 'ISO',  
]
```

DOWNLOAD_FORMATS_VECTOR

Specifies which formats for vector data are available for users to download.

Default:

```
DOWNLOAD_FORMATS_VECTOR = [  
    'JPEG', 'PDF', 'PNG', 'Zipped Shapefile', 'GML 2.0', 'GML 3.1.1', 'CSV',  
    'Excel', 'GeoJSON', 'KML', 'View in Google Earth', 'Tiles',  
]
```

DOWNLOAD_FORMATS_RASTER

Specifies which formats for raster data are available for users to download.

Default:

```
DOWNLOAD_FORMATS_RASTER = [  
    'JPEG', 'PDF', 'PNG', 'Tiles',  
]
```

E

EMAIL_ENABLE

Default: False

Options:

- EMAIL_BACKEND
Default: `django.core.mail.backends.smtp.EmailBackend`
Env: `DJANGO_EMAIL_BACKEND`
- EMAIL_HOST
Default: `localhost`
- EMAIL_PORT
Default: `25`
- EMAIL_HOST_USER
Default: `' '`
- EMAIL_HOST_PASSWORD
Default: `' '`
- EMAIL_USE_TLS
Default: `False`
- EMAIL_USE_SSL
Default: `False`
- DEFAULT_FROM_EMAIL
Default: `GeoNode <no-reply@geonode.org>`

EPSG_CODE_MATCHES

Default:

```
{
  'EPSG:4326': '(4326) WGS 84',
  'EPSG:900913': '(900913) Google Maps Global Mercator',
  'EPSG:3857': '(3857) WGS 84 / Pseudo-Mercator',
  'EPSG:3785': '(3785 DEPRECATED) Popular Visualization CRS / Mercator',
  'EPSG:32647': '(32647) WGS 84 / UTM zone 47N',
  'EPSG:32736': '(32736) WGS 84 / UTM zone 36S'
}
```

Supported projections human readable descriptions associated to their EPSG Codes. This list will be presented to the user during the upload process whenever GeoNode won't be able to recognize a suitable projection. Those codes should be aligned to the *UPLOADER* ones and available in GeoServer also.

F

FREETEXT_KEYWORDS_READONLY

Default: `False`

Env: `FREETEXT_KEYWORDS_READONLY`

Make Free-Text Keywords writable from users. Or read-only when set to `False`.

G

GEOFENCE_SECURITY_ENABLED

Default: `True` (False is Test is true)

Env: `GEOFENCE_SECURITY_ENABLED`

Whether the geofence security system is used.

GEOIP_PATH

Default: Path to project

Env: `PROJECT_ROOT`

The local path where GeoIPCities.dat is written to. Make sure your user has to have write permissions.

GEONODE_APPS

If enabled contrib apps are used.

GEONODE_CLIENT_LAYER_PREVIEW_LIBRARY

Default: "mapstore"

The library to use for display preview images of layers. The library choices are:

"mapstore" "leaflet" "react"

GEONODE_EXCHANGE

Default:: Exchange("default", type="direct", durable=True)

The definition of Exchanges published by geonode. Find more about Exchanges at [celery docs](#).

GEOSERVER_EXCHANGE

Default:: Exchange("geonode", type="topic", durable=False)

The definition of Exchanges published by GeoServer. Find more about Exchanges at [celery docs](#).

GEOSERVER_LOCATION

Default: http://localhost:8080/geoserver/

Env: GEOSERVER_LOCATION

Url under which GeoServer is available.

GEOSERVER_PUBLIC_HOST

Default: SITE_HOST_NAME (Variable)

Env: GEOSERVER_PUBLIC_HOST

Public hostname under which GeoServer is available.

GEOSERVER_PUBLIC_LOCATION

Default: SITE_HOST_NAME (Variable)

Env: GEOSERVER_PUBLIC_LOCATION

Public location under which GeoServer is available.

GEOSERVER_PUBLIC_PORT

Default: 8080 (Variable)

Env: GEOSERVER_PUBLIC_PORT

Public Port under which GeoServer is available.

GEOSERVER_WEB_UI_LOCATION

Default: `GEOSERVER_PUBLIC_LOCATION` (Variable)

Env: `GEOSERVER_WEB_UI_LOCATION`

Public location under which GeoServer is available.

GROUP_PRIVATE_RESOURCES

Default: `False`

Env: `GROUP_PRIVATE_RESOURCES`

If this option is enabled, Resources belonging to a Group won't be visible by others

H

HAYSTACK_FACET_COUNTS

Default: `True`

Env: `HAYSTACK_FACET_COUNTS`

If set to `True` users will be presented with feedback about the number of resources which matches terms they may be interested in.

HAYSTACK_SEARCH

Default: `False`

Env: `HAYSTACK_SEARCH`

Enable/disable haystack Search Backend Configuration.

L

LEAFLET_CONFIG

A dictionary used for Leaflet configuration.

LICENSES

Default:

```
{
  'ENABLED': True,
  'DETAIL': 'above',
  'METADATA': 'verbose',
}
```

Enable Licenses User Interface

LOCAL_SIGNALS_BROKER_URL

Default: `memory://`

LOCKDOWN_GEONODE

Default: `False`

Env: `LOCKDOWN_GEONODE`

By default, the GeoNode application allows visitors to view most pages without being authenticated. If this is set to `True` users must be authenticated before accessing URL routes not included in `AUTH_EXEMPT_URLS`.

LOGIN_URL

Default: `{%account/login/'}.format(SITEURL)`

Env: `LOGIN_URL`

The URL where requests are redirected for login.

LOGOUT_URL

Default: `{%account/login/'}.format(SITEURL)`

Env: `LOGOUT_URL`

The URL where requests are redirected for logout.

M

MAP_CLIENT_USE_CROSS_ORIGIN_CREDENTIALS

Default: `False`

Env: `MAP_CLIENT_USE_CROSS_ORIGIN_CREDENTIALS`

Enables cross origin requests for geonode-client.

MAPSTORE_BASELAYERS

Default:

```
[
  {
    "type": "osm",
    "title": "Open Street Map",
    "name": "mapnik",
    "source": "osm",
    "group": "background",
    "visibility": True
  }, {
    "type": "tileprovider",
```

(continues on next page)

(continuação da página anterior)

```

        "title": "OpenTopoMap",
        "provider": "OpenTopoMap",
        "name": "OpenTopoMap",
        "source": "OpenTopoMap",
        "group": "background",
        "visibility": False
    }, {
        "type": "wms",
        "title": "Sentinel-2 cloudless - https://s2maps.eu",
        "format": "image/jpeg",
        "id": "s2cloudless",
        "name": "s2cloudless:s2cloudless",
        "url": "https://maps.geo-solutions.it/geoserver/wms",
        "group": "background",
        "thumbURL": "%sstatic/mapstorestyle/img/s2cloudless-s2cloudless.png"
↪ % SITEURL,
        "visibility": False
    }, {
        "source": "ol",
        "group": "background",
        "id": "none",
        "name": "empty",
        "title": "Empty Background",
        "type": "empty",
        "visibility": False,
        "args": ["Empty Background", {"visibility": False}]
    }
]

```

Env: MAPSTORE_BASELAYERS

Allows to specify which backgrounds MapStore should use. The parameter `visibility` for a layer, specifies which one is the default one.

A sample configuration using the Bing background without OpenStreetMap, could be the following one:

```

[
  {
    "type": "bing",
    "title": "Bing Aerial",
    "name": "AerialWithLabels",
    "source": "bing",
    "group": "background",
    "apiKey": "{{apiKey}}",
    "visibility": True
  }, {
    "type": "tileprovider",
    "title": "OpenTopoMap",
    "provider": "OpenTopoMap",
    "name": "OpenTopoMap",
    "source": "OpenTopoMap",
    "group": "background",
    "visibility": False
  }, {

```

(continues on next page)

(continuação da página anterior)

```

        "type": "wms",
        "title": "Sentinel-2 cloudless - https://s2maps.eu",
        "format": "image/jpeg",
        "id": "s2cloudless",
        "name": "s2cloudless:s2cloudless",
        "url": "https://maps.geo-solutions.it/geoserver/wms",
        "group": "background",
        "thumbURL": "%sstatic/mapstorestyle/img/s2cloudless-s2cloudless.png"
    ↪ % SITEURL,
        "visibility": False
    }, {
        "source": "ol",
        "group": "background",
        "id": "none",
        "name": "empty",
        "title": "Empty Background",
        "type": "empty",
        "visibility": False,
        "args": ["Empty Background", {"visibility": False}]
    }
]

```

Aviso: To use a Bing background, you need to correctly set and provide a valid BING_API_KEY

MAX_DOCUMENT_SIZE

Default:2

Env: MAX_DOCUMENT_SIZE

Allowed size for documents in MB.

METADATA_PARSERS

Is possible to define multiple XML parsers for ingest XML during the layer upload.

The variable should be declared in this way in *settings.py*:

```
METADATA_PARSERS = ["list", "of", "parsing", "functions"]
```

If you want to always use the default metadata parser and after use your own, the variable must be set with first value as `__DEFAULT__` Example:

```
METADATA_PARSERS = ["__DEFAULT__", "custom_parsing_function"]
```

If not set, the system will use the `__DEFAULT__` parser.

The custom parsing function must be accept in input 6 parameter that are:

- xml (xmlfile)
- uuid (str)
- vals (dict)
- regions (list)
- keywords (list)
- custom (dict)

If you want to use your parser after the default one, here is how the variable are populated:

- xml: the XML file to parse
- uuid: the UUID of the layer
- vals: Dictionary of information that belong to ResourceBase
- regions: List of regions extracted from the XML
- keywords: List of dict of keywords already divided between free-text and thesarus
- custom: Custom variable

NOTE: the keywords must be in a specific format, since later this dict, will be ingested by the *KeywordHandler* which will assign the keywords/thesaurus to the layer.

Here is an example of expected parser function

For more information, please rely to *TestCustomMetadataParser* which contain a smoke test to explain the functionality

METADATA_STORERS

Is possible to define multiple Layer storer during the layer upload.

The variable should be declared in this way:

```
METADATA_STORERS = ["custom_storer_function"]
```

NOTE: By default the Layer is always saved with the default behaviour.

The custom storer function must be accept in input 2 parameter that are:

- Layer (layer model instance)
- custom (dict)

Here is how the variable are populated by default:

- layer (layer model instance) that we wanto to change
- custom: custom dict populated by the parser

Here is an example of expected storer function

For more information, please rely to *TestMetadataStorers* which contain a smoke test to explain the functionality

MISSING_THUMBAIL

Default: `geonode/img/missing_thumb.png`

The path to an image used as thumbnail placeholder.

MEMCACHED_BACKEND

Default: `django.core.cache.backends.memcached.PyMemcacheCache`

Define which backend of memcached will be used

MEMCACHED_ENABLED

Default: `False`

If `True`, will use `MEMCACHED_BACKEND` as default backend in `CACHES`

MODIFY_TOPICCATEGORY

Default: `False`

Metadata Topic Categories list should not be modified, as it is strictly defined by ISO (See: <http://www.isotc211.org/2005/resources/Codelist/gmxCodeLists.xml> and check the `<CodeListDictionary gml:id=»MD_MD_TopicCategoryCode»>` element).

Some customization is still possible changing the `is_choice` and the `GeoNode` description fields.

In case it is necessary to add/delete/update categories, it is possible to set the `MODIFY_TOPICCATEGORY` setting to `True`.

MONITORING_ENABLED

Default: `False`

Enable internal monitoring application (*geonode.monitoring*). If set to `True`, add following code to your local settings:

```

MONITORING_ENABLED = True
# add following lines to your local settings to enable monitoring
if MONITORING_ENABLED:
    INSTALLED_APPS + ('geonode.monitoring',)
    MIDDLEWARE_CLASSES + ('geonode.monitoring.middleware.MonitoringMiddleware
    ↪',)

```

See *Read-Only and Maintenance Mode* for details.

MONITORING_DATA_AGGREGATION

Default:

```

(
    (timedelta(seconds=0), timedelta(minutes=1)),
    (timedelta(days=1), timedelta(minutes=60)),
    (timedelta(days=14), timedelta(days=1)),
)

```

Configure aggregation of past data to control data resolution. It lists data age and aggregation in reverse order, by default:

- for current data, 1 minute resolution
- for data older than 1 day, 1-hour resolution
- for data older than 2 weeks, 1 day resolution

See *Read-Only and Maintenance Mode* for further details.

This setting takes effects only if *USER_ANALYTICS_ENABLED* is true.

MONITORING_DATA_TTL

Default: 365

Env: MONITORING_DATA_TTL

How long monitoring data should be stored in days.

MONITORING_DISABLE_CSRF

Default: False

Env: MONITORING_DISABLE_CSRF

Set this to true to disable csrf check for notification config views, use with caution - for dev purpose only.

MONITORING_SKIP_PATHS

Default:

```
(
    '/api/o/',
    '/monitoring/',
    '/admin',
    '/jsi18n',
    STATIC_URL,
    MEDIA_URL,
    re.compile('^/[a-z]{2}/admin/'),
)
```

Skip certain useless paths to not to mud analytics stats too much. See *Read-Only and Maintenance Mode* to learn more about it.

This setting takes effects only if *USER_ANALYTICS_ENABLED* is true.

N

NOTIFICATIONS_MODULE

Default: `pinax.notifications`

App used for notifications. (pinax.notifications or notification)

NOTIFICATION_ENABLED

Default: `True`

Env: `NOTIFICATION_ENABLED`

Enable or disable the notification system.

O

OAuth2_API_KEY

Default: `None`

Env: `OAuth2_API_KEY`

In order to protect oauth2 REST endpoints, used by GeoServer to fetch user roles and infos, you should set this key and configure the `geonode REST role service` accordingly. Keep it secret!

Aviso: If not set, the endpoint can be accessed by users without authorization.

OAuth2_PROVIDER

Ref.: [OAuth Toolkit settings](#)

OAuth2_PROVIDER_APPLICATION_MODEL

Default: `oauth2_provider.Application`

Ref.: [OAuth Toolkit settings](#)

OAuth2_PROVIDER_ACCESS_TOKEN_MODEL

Default: `oauth2_provider.AccessToken`

Ref.: [OAuth Toolkit settings](#)

OAuth2_PROVIDER_ID_TOKEN_MODEL

Default: `oauth2_provider.IDToken`

Ref.: [OAuth Toolkit settings](#)

OAuth2_PROVIDER_GRANT_MODEL

Default: `oauth2_provider.Grant`

Ref.: [OAuth Toolkit settings](#)

OAuth2_PROVIDER_REFRESH_TOKEN_MODEL

Default: `oauth2_provider.RefreshToken`

Ref.: [OAuth Toolkit settings](#)

OGC_SERVER_DEFAULT_PASSWORD

Default: `geoserver`

Env: `GEOSERVER_ADMIN_PASSWORD`

The geoserver password.

OGC_SERVER_DEFAULT_USER

Default: `admin`

Env: `GEOSERVER_ADMIN_USER`

The GeoServer user.

OGC_SERVER

Default: `{}` (Empty dictionary)

A dictionary of OGC servers and their options. The main server should be listed in the “default” key. If there is no “default” key or if the `OGC_SERVER` setting does not exist, Geonode will raise an Improperly Configured exception. Below is an example of the `OGC_SERVER` setting:

```
OGC_SERVER = {
    'default' : {
        'LOCATION' : 'http://localhost:8080/geoserver/',
        'USER' : 'admin',
        'PASSWORD' : 'geoserver',
    }
}
```

- **BACKEND**

Default: `"geonode.geoserver"`

The OGC server backend to use. The backend choices are:

`'geonode.geoserver'`

- **BACKEND_WRITE_ENABLED**

Default: `True`

Specifies whether the OGC server can be written to. If False, actions that modify data on the OGC server will not execute.

- DATASTORE

Default: '' (Empty string)

An optional string that represents the name of a vector datastore, where Geonode uploads are imported into. To support vector datastore imports there also needs to be an entry for the datastore in the DATABASES dictionary with the same name. Example:

```
OGC_SERVER = {
    'default' : {
        'LOCATION' : 'http://localhost:8080/geoserver/',
        'USER' : 'admin',
        'PASSWORD' : 'geoserver',
        'DATASTORE': 'geonode_imports'
    }
}

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': 'development.db',
    },
    'geonode_imports' : {
        'ENGINE': 'django.contrib.gis.db.backends.postgis',
        'NAME': 'geonode_imports',
        'USER' : 'geonode_user',
        'PASSWORD' : 'a_password',
        'HOST' : 'localhost',
        'PORT' : '5432',
    }
}
```

- GEONODE_SECURITY_ENABLED

Default: True

A boolean that represents whether GeoNode's security application is enabled.

- LOCATION

Default: "http://localhost:8080/geoserver/"

A base URL from which GeoNode can construct OGC service URLs. If using GeoServer you can determine this by visiting the GeoServer administration home page without the /web/ at the end. For example, if your GeoServer administration app is at <http://example.com/geoserver/web/>, your server's location is <http://example.com/geoserver>.

- MAPFISH_PRINT_ENABLED

Default: True

A boolean that represents whether the MapFish printing extension is enabled on the server.

- PASSWORD

Default: 'geoserver'

The administrative password for the OGC server as a string.

- PRINT_NG_ENABLED

Default: True

A boolean that represents whether printing of maps and layers is enabled.

- **PUBLIC_LOCATION**

Default: `"http://localhost:8080/geoserver/"`

The URL used to in most public requests from Geonode. This setting allows a user to write to one OGC server (the `LOCATION` setting) and read from a separate server or the `PUBLIC_LOCATION`.

- **USER**

Default: `'admin'`

The administrative username for the OGC server as a string.

- **WMST_ENABLED**

Default: `False`

Not implemented.

- **WPS_ENABLED**

Default: `False`

Not implemented.

- **TIMEOUT**

Default: `10`

The maximum time, in seconds, to wait for the server to respond.

OGP_URL

Default: `http://geodata.tufts.edu/solr/select`

Env: `OGP_URL`

Endpoint of `geodata.tufts.edu` `getCapabilities`.

OPENGRAPH_ENABLED

Default:: `True`

A boolean that specifies whether Open Graph is enabled. Open Graph is used by Facebook and Slack.

P

PINAX_NOTIFICATIONS_BACKENDS

Default: `("email", _EMAIL_BACKEND, 0),`

Used notification backend. This is a [pinax notification setting](#):

PINAX_NOTIFICATIONS_LOCK_WAIT_TIMEOUT

Default: -1

Env: NOTIFICATIONS_LOCK_WAIT_TIMEOUT

It defines how long to wait for the lock to become available. Default of -1 means to never wait for the lock to become available. This is a [pinax notification setting](#):

PINAX_NOTIFICATIONS_QUEUE_ALL

Default: -1

Env: NOTIFICATIONS_LOCK_WAIT_TIMEOUT

By default, calling `notification.send` will send the notification immediately, however, if you set this setting to `True`, then the default behavior of the `send` method will be to queue messages in the database for sending via the `emit_notices` command. This is a [pinax notification setting](#):

PINAX_RATINGS_CATEGORY_CHOICES

Default:

```
{
  "maps.Map": {
    "map": "How good is this map?"
  },
  "layers.Layer": {
    "layer": "How good is this layer?"
  },
  "documents.Document": {
    "document": "How good is this document?"
  }
}
```

PROFILE_EDIT_EXCLUDE_FIELD

Default: []

A list of element (item name) to exclude from the Profile Edit page.

Example:

```
PROFILE_EDIT_EXCLUDE_FIELD=['organization', 'language']
```

PROXY_ALLOWED_HOSTS

Default: () (Empty tuple)

A tuple of strings representing the host/domain names that GeoNode can proxy requests to. This is a security measure to prevent an attacker from using the GeoNode proxy to render malicious code or access internal sites.

Values in this tuple can be fully qualified names (e.g. “`www.geonode.org`”), in which case they will be matched against the request’s Host header exactly (case-insensitive, not including port). A value beginning with a period can be used as a subdomain wildcard: `.geonode.org` will match `geonode.org`,

www.geonode.org, and any other subdomain of geonode.org. A value of “*” will match anything and is not recommended for production deployments.

PROXY_URL

Default /proxy/?url=

The URL to a proxy that will be used when making client-side requests in GeoNode. By default, the internal GeoNode proxy is used but administrators may favor using their own, less restrictive proxies.

PYCSW

A dict with pycsw’s configuration with two possible keys CONFIGURATION and FILTER.

CONFIGURATION Of note are the sections `metadata:main` to set CSW server metadata and `metadata:inspire` to set INSPIRE options. Setting `metadata:inspire['enabled']` to `true` will enable INSPIRE support. Server level configurations can be overridden in the `server` section. See <http://docs.pycsw.org/en/latest/configuration.html> for full pycsw configuration details.

FILTER Optional settings in order to add a filter to the CSW filtering. The filter follow the django orm structure and must be a *ResourceBase* field/related field. By default CSW will filter only for *layer* `resource_type`

Example of PYCSW configuration. PYCSW: {

```
    “CONFIGURATION”: { ... }, “FILTER”: { “resource_type__in”:[“layer”] }
}
```

R

RABBITMQ_SIGNALS_BROKER_URL

Default: `amqp://localhost:5672`

The Rabbitmq endpoint

RECAPTCHA_ENABLED

Default: `False`

Env: `RECAPTCHA_ENABLED`

Allows enabling reCaptcha field on signup form. Valid Captcha Public and Private keys will be needed as specified here <https://pypi.org/project/django-recaptcha/#installation>

You will need to generate a keys pair for reCaptcha v2 for your domain from <https://www.google.com/recaptcha/admin/create>

More options will be available by enabling this setting:

- **ACCOUNT_SIGNUP_FORM_CLASS**

Default: `geonode.people.forms.AllauthReCaptchaSignupForm`

Env: `ACCOUNT_SIGNUP_FORM_CLASS`

Enabled only when the [*RECAPTCHA_ENABLED*](#) option is `True`.

- **INSTALLED_APPS**

The captcha must be present on INSTALLED_APPS, otherwise you'll get an error.

When enabling the *RECAPTCHA_ENABLED* option through the environment, this setting will be automatically added by GeoNode as follows:

```
if 'captcha' not in INSTALLED_APPS:
    INSTALLED_APPS += ('captcha',)
```

- **RECAPTCHA_PUBLIC_KEY**

Default: geonode_RECAPTCHA_PUBLIC_KEY

Env: RECAPTCHA_PUBLIC_KEY

You will need to generate a keys pair for reCaptcha v2 for your domain from <https://www.google.com/recaptcha/admin/create>

For mode details on the reCaptcha package, please see:

1. <https://pypi.org/project/django-recaptcha/#installation>
2. <https://pypi.org/project/django-recaptcha/#local-development-and-functional-testing>

- **RECAPTCHA_PRIVATE_KEY**

Default: geonode_RECAPTCHA_PRIVATE_KEY

Env: RECAPTCHA_PRIVATE_KEY

You will need to generate a keys pair for reCaptcha v2 for your domain from <https://www.google.com/recaptcha/admin/create>

For mode details on the reCaptcha package, please see:

1. <https://pypi.org/project/django-recaptcha/#installation>
2. <https://pypi.org/project/django-recaptcha/#local-development-and-functional-testing>

RECAPTCHA_PUBLIC_KEY

Default: geonode_RECAPTCHA_PUBLIC_KEY

Env: RECAPTCHA_PUBLIC_KEY

You will need to generate a keys pair for reCaptcha v2 for your domain from <https://www.google.com/recaptcha/admin/create>

Ref. to *RECAPTCHA_ENABLED*

RECAPTCHA_PRIVATE_KEY

Default: geonode_RECAPTCHA_PRIVATE_KEY

Env: RECAPTCHA_PRIVATE_KEY

You will need to generate a keys pair for reCaptcha v2 for your domain from <https://www.google.com/recaptcha/admin/create>

Ref. to *RECAPTCHA_ENABLED*

REDIS_SIGNALS_BROKER_URL

Default: `redis://localhost:6379/0`

The Redis endpoint.

REGISTERED_MEMBERS_GROUP_NAME

Default: `registered-members`

Env: `REGISTERED_MEMBERS_GROUP_NAME`

Used by `AUTO_ASSIGN_REGISTERED_MEMBERS_TO_REGISTERED_MEMBERS_GROUP_NAME` settings.

REGISTERED_MEMBERS_GROUP_TITLE

Default: `Registered Members`

Env: `REGISTERED_MEMBERS_GROUP_TITLE`

Used by `AUTO_ASSIGN_REGISTERED_MEMBERS_TO_REGISTERED_MEMBERS_GROUP_NAME` settings.

REGISTRATION_OPEN

Default: `False`

A boolean that specifies whether users can self-register for an account on your site.

RESOURCE_PUBLISHING

Default: `False`

By default, the GeoNode application allows GeoNode staff members to publish/unpublish resources. By default, resources are published when created. When this setting is set to `True` the staff members will be able to unpublish a resource (and eventually publish it back).

S

S3_MEDIA_ENABLED

Default: `False`

Env: `S3_MEDIA_ENABLED`

Enable/disable Amazon S3 media storage.

S3_STATIC_ENABLED

Default: False

Env: S3_STATIC_ENABLED

Enable/disable Amazon S3 static storage.

SEARCH_FILTERS

Default:

```
'TEXT_ENABLED': True,
'TYPE_ENABLED': True,
'CATEGORIES_ENABLED': True,
'OWNERS_ENABLED': True,
'KEYWORDS_ENABLED': True,
'H_KEYWORDS_ENABLED': True,
'T_KEYWORDS_ENABLED': True,
'DATE_ENABLED': True,
'REGION_ENABLED': True,
'EXTENT_ENABLED': True,
```

Enabled Search Filters for filtering resources.

SECURE_BROWSER_XSS_FILTER

Default: True

Env: SECURE_BROWSER_XSS_FILTER

If True, the SecurityMiddleware sets the X-XSS-Protection: 1; mode=block header on all responses that do not already have it. This is [Djangosettings.https://docs.djangoproject.com/en/3.2/ref/settings/#secure-browser-xss-filter](https://docs.djangoproject.com/en/3.2/ref/settings/#secure-browser-xss-filter)

SECURE_CONTENT_TYPE_NOSNIFF

Default: True

Env: SECURE_CONTENT_TYPE_NOSNIFF

If True, the SecurityMiddleware sets the X-Content-Type-Options: nosniff header on all responses that do not already have it. This is [Django settings](#):

SECURE_HSTS_INCLUDE_SUBDOMAINS

Default: True

Env: SECURE_HSTS_INCLUDE_SUBDOMAINS

This is Django settings: <https://docs.djangoproject.com/en/3.2/ref/settings/#secure-hsts-include-subdomains>

SECURE_HSTS_SECONDS

Default: 3600

Env: SECURE_HSTS_SECONDS

This is [Django settings](#): If set to a non-zero integer value, the SecurityMiddleware sets the HTTP Strict Transport Security header on all responses that do not already have it.

SECURE_SSL_REDIRECT

If True, the SecurityMiddleware redirects all non-HTTPS requests to HTTPS (except for those URLs matching a regular expression listed in SECURE_REDIRECT_EXEMPT). This is [Django settings](#):

SERVICE_UPDATE_INTERVAL

Default: 0

The Interval services are updated.

SESSION_COOKIE_SECURE

Default: False

Env: SESSION_COOKIE_SECURE

This is a [Django setting](#):

SESSION_EXPIRED_CONTROL_ENABLED

Default: True

Env: SESSION_EXPIRED_CONTROL_ENABLED

By enabling this variable, a new middleware `geonode.security.middleware.SessionControlMiddleware` will be added to the `MIDDLEWARE_CLASSES`. The class will check every request to GeoNode and it will force a log out whenever one of the following conditions occurs:

1. The OAuth2 Access Token is not valid anymore or it is expired.

Aviso: The Access Token might be invalid for various reasons. Usually a misconfiguration of the OAuth2 GeoServer application. The latter is typically installed and configured automatically at GeoNode bootstrap through the default fixtures.

2. The user has been deactivated for some reason; an Admin has disabled it or its password has expired.

Whenever the middleware terminates the session and the user forced to log out, a message will appear to the GeoNode interface.

SHOW_PROFILE_EMAIL

Default: `False`

A boolean which specifies whether to display the email in the user's profile.

SITE_HOST_NAME

Default: `localhost`

Env: `SITE_HOST_NAME`

The hostname used for GeoNode.

SITE_HOST_PORT

Default: `8000`

Env: `SITE_HOST_PORT`

The Site hostport.

SITEURL

Default: `'http://localhost:8000/'`

A base URL for use in creating absolute links to Django views and generating links in metadata.

SKIP_PERMS_FILTER

Default: `False`

Env: `SKIP_PERMS_FILTER`

If set to true permissions prefiltering is avoided.

SOCIALACCOUNT_ADAPTER

Default: `geonode.people.adapters.SocialAccountAdapter`

This is a [django-allauth setting](#) It allows specifying a custom class to handle authentication for social accounts.

SOCIALACCOUNT_AUTO_SIGNUP

Default: `True`

Attempt to bypass the signup form by using fields (e.g. username, email) retrieved from the social account provider. This is a [Django-allauth setting](#):

SOCIALACCOUNT_PROVIDERS

Default:

```
{
  'linkedin_oauth2': {
    'SCOPE': [
      'r_emailaddress',
      'r_basicprofile',
    ],
    'PROFILE_FIELDS': [
      'emailAddress',
      'firstName',
      'headline',
      'id',
      'industry',
      'lastName',
      'pictureUrl',
      'positions',
      'publicProfileUrl',
      'location',
      'specialties',
      'summary',
    ]
  },
  'facebook': {
    'METHOD': 'oauth2',
    'SCOPE': [
      'email',
      'public_profile',
    ],
    'FIELDS': [
      'id',
      'email',
      'name',
      'first_name',
      'last_name',
      'verified',
      'locale',
      'timezone',
      'link',
      'gender',
    ]
  },
}
```

This is a `django-allauth` setting. It should be a dictionary with provider specific settings.

SOCIALACCOUNT_PROFILE_EXTRACTORS

Default:

```
{
  "facebook": "geonode.people.profileextractors.FacebookExtractor",
  "linkedin_oauth2": "geonode.people.profileextractors.LinkedInExtractor",
}
```

A dictionary with provider ids as keys and path to custom profile extractor classes as values.

SOCIAL_BUTTONS

Default: True

A boolean which specifies whether the social media icons and JavaScript should be rendered in GeoNode.

SOCIAL_ORIGINS

Default:

```
SOCIAL_ORIGINS = [{
  "label": "Email",
  "url": "mailto:?subject={name}&body={url}",
  "css_class": "email"
}, {
  "label": "Facebook",
  "url": "http://www.facebook.com/sharer.php?u={url}",
  "css_class": "fb"
}, {
  "label": "Twitter",
  "url": "https://twitter.com/share?url={url}",
  "css_class": "tw"
}, {
  "label": "Google +",
  "url": "https://plus.google.com/share?url={url}",
  "css_class": "gp"
}]
```

A list of dictionaries that are used to generate the social links displayed in the Share tab. For each origin, the name and URL format parameters are replaced by the actual values of the ResourceBase object (layer, map, document).

SOCIALACCOUNT_WITH_GEONODE_LOCAL_SINGUP

Default: True

Variable which controls displaying local account registration form. By default form is visible

SRID

Default:

```
{  
  'DETAIL': 'never',  
}
```

SEARCH_RESOURCES_EXTENDED

Default: True

This will extend search with additional properties. By default it's on and search engine will check resource title or purpose or abstract. When set to False just title lookup is performed.

T

TASTYPIE_DEFAULT_FORMATS

Default: json

This setting allows you to globally configure the list of allowed serialization formats for your entire site. This is a [tastypie setting](#):

THEME_ACCOUNT_CONTACT_EMAIL

Default: 'admin@example.com'

This email address is added to the bottom of the password reset page in case users have trouble unlocking their account.

THESAURI

Default = []

A list of Keywords thesauri settings: For example *THESAURI* = [{*"name": "inspire_themes", "required": True, "filter": True*}, {*"name": "inspire_concepts", "filter": True*},]

TOPICCATEGORY_MANDATORY

Default: False

Env: TOPICCATEGORY_MANDATORY

If this option is enabled, Topic Categories will become strictly Mandatory on Metadata Wizard

TWITTER_CARD

Default:: True

A boolean that specifies whether Twitter cards are enabled.

TWITTER_SITE

Default:: '@GeoNode '

A string that specifies the site to for the twitter:site meta tag for Twitter Cards.

TWITTER_HASHTAGS

Default:: ['geonode']

A list that specifies the hashtags to use when sharing a resource when clicking on a social link.

TINYMCE_DEFAULT_CONFIG

Default:

```
{
  "selector": "textarea#id_resource-featureinfo_custom_template",
  "theme": "silver",
  "height": 500,
  "plugins": 'print preview paste importcss searchreplace autolink_
↪autosave save directionality code visualblocks visualchars fullscreen_
↪image link media template codesample table charmap hr pagebreak_
↪nonbreaking anchor toc insertdatetime advlist lists wordcount imagetools_
↪textpattern noneditable help charmap quickbars emoticons',
  "imagetools_cors_hosts": ['picsum.photos'],
  "menubar": 'file edit view insert format tools table help',
  "toolbar": 'undo redo | bold italic underline strikethrough | fontselect_
↪fontsize select formatselect | alignleft aligncenter alignright_
↪alignjustify | outdent indent | numlist bullist | forecolor backcolor_
↪removeformat | pagebreak | charmap emoticons | fullscreen preview save |_
↪insertfile image media template link anchor codesample | ltr rtl',
  "toolbar_sticky": "true",
  "autosave_ask_before_unload": "true",
  "autosave_interval": "30s",
  "autosave_prefix": "{path}{query}-{id}-",
  "autosave_restore_when_empty": "false",
  "autosave_retention": "2m",
  "image_advtab": "true",
  "content_css": '//www.tiny.cloud/css/codepen.min.css',
  "importcss_append": "true",
  "image_caption": "true",
  "quickbars_selection_toolbar": 'bold italic | quicklink h2 h3 blockquote_
↪quickimage quicktable',
  "noneditable_noneditable_class": "mceNonEditable",
  "toolbar_mode": 'sliding',
  "contextmenu": "link image imagetools table",
  "templates": [
```

(continues on next page)

(continuação da página anterior)

```

    {
      "title": 'New Table',
      "description": 'creates a new table',
      "content": '<div class="mceTpl"><table width="98%" border="0"
↪cellspacing="0" cellpadding="0"><tr><th scope="col"> </th><th scope="col">
↪</th></tr><tr><td> </td><td> </td></tr></table></div>'
    },
    {
      "title": 'Starting my story',
      "description": 'A cure for writers block',
      "content": 'Once upon a time...'
    },
    {
      "title": 'New list with dates',
      "description": 'New List with dates',
      "content": '<div class="mceTpl"><span class="cdate">cdate</span>
↪<br /><span class="mdate">mdate</span><h2>My List</h2><ul><li></li><li></
↪li></ul></div>'
    }
  ],
  "template_cdate_format": '[Date Created (CDATE): %m/%d/%Y : %H:%M:%S]',
  "template_mdate_format": '[Date Modified (MDATE): %m/%d/%Y : %H:%M:%S]',
}

```

HTML WYSIWYG Editor (TINYMCE) Menu Bar Settings. For more info see:

- <https://django-tinymce.readthedocs.io/en/latest/installation.html#configuration>
- *Personalizando os Modelos GetFeatureInfo das Camadas*

U

UI_REQUIRED_FIELDS

If this option is enabled, the input selected (we are referring to the one present in the optional Metadata-Tab on the Metadata-Wizard) will become mandatory.

The fields that can be mandatory are:

```

id_resource-edition => Label: Edition
id_resource-purpose => Label: Purpose
id_resource-supplemental_information => Label: Supplemental information
id_resource-temporal_extent_start_picker => Label: temporal extent start
id_resource-temporal_extent_end => Label: temporal extent end
id_resource-maintenance_frequency => Label: Maintenance frequency
id_resource-spatial_representation_type => Label: Spatial representation type

```

If at least one on the above ids is set in this configuration, the panel header will change from *Optional* to *Mandatory*

Configuration Example:

```
UI_REQUIRED_FIELDS = ["id_resource-edition"]
```


UNOCONV_ENABLE

Default: False

Env: UNOCONV_ENABLE

UPLOADER

Default:

```
{
  'BACKEND' : 'geonode.importer',
  'OPTIONS' : {
    'TIME_ENABLED': False,
  }
}
```

A dictionary of Uploader settings and their values.

- BACKEND

Default: 'geonode.importer'

The importer backend requires the GeoServer importer extension to be enabled.

- OPTIONS

Default:

```
'OPTIONS' : {
  'TIME_ENABLED': False,
}
```

- TIME_ENABLED

Default: False

A boolean that specifies whether the upload should allow the user to enable time support when uploading data.

USER_MESSAGES_ALLOW_MULTIPLE_RECIPIENTS

Default: True

Env: USER_MESSAGES_ALLOW_MULTIPLE_RECIPIENTS

Set to true to have multiple recipients in /message/create/

USER_ANALYTICS_ENABLED

Default: False

Env: USER_ANALYTICS_ENABLED

Set to true to anonymously collect user data for analytics. If true you have to set *MONITORING_DATA_AGGREGATION* and *MONITORING_SKIP_PATHS*.

See *Read-Only and Maintenance Mode* to learn more about it.

USER_ANALYTICS_GZIP

Default: False

Env: USER_ANALYTICS_GZIP

To be used with USER_ANALYTICS_ENABLED. Compress gzip json messages before sending to logstash.

UUID HANDLER

Is possible to define an own uuidhandler for the Layer.

To start using your own handler, is needed to add the following configuration:

`LAYER_UUID_HANDLER = «mymodule.myfile.MyObject»`

The Object must accept as *init* the *instance* of the layer and have a method named *create_uuid()*

here is an example:

```
class MyObject():
    def __init__(self, instance):
        self.instance = instance

    def create_uuid(self):
        # here your code
        pass
```

X

X_FRAME_OPTIONS

Default: 'ALLOW-FROM %s' % SITEURL

This is a [Django setting](#)

1.14 Customize the Look and Feel

1.14.1 GeoNode Themes

We have already explained in *Simple Theming* how to change the GeoNode theme directly from the *Admin Interface*. This is an easy way for customizing GeoNode appearance but, in some cases, you might want to have more control on it.

In those cases, you have to venture into the code and it is highly recommended to use a GeoNode Project and customize it instead of the GeoNode default HTML/CSS code. See the following sections to learn more about that.

1.14.2 Theming your GeoNode Project

There are a range of options available to you if you want to change the default look and feel of your *GeoNode Project*. Since GeoNode's style is based on *Bootstrap* you will be able to make use of all that Bootstrap has to offer in terms of theme customization. You should consult Bootstrap's documentation as your primary guide once you are familiar with how GeoNode implements Bootstrap and how you can override GeoNode's theme and templates in your own project.

Logos and graphics

GeoNode intentionally does not include a large number of graphics files in its interface. This keeps page loading time to a minimum and makes for a more responsive interface. That said, you are free to customize your GeoNode's interface by simply changing the default logo, or by adding your own images and graphics to deliver a GeoNode experience the way you envision it.

Your GeoNode project has a directory already set up for storing your own images at `<my_geonode>/static/img`. You should place any image files that you intend to use for your project in this directory.

Let's walk through an example of the steps necessary to change the default logo.

1. Change to the `img` directory:

```
$ cd <my_geonode>/static/img
```

2. If you haven't already, obtain your logo image. The URL below is just an example, so you will need to change this URL to match the location of your file or copy it to this location:

```
$ sudo wget https://upload.wikimedia.org/wikipedia/commons/thumb/a/ac/Service_
↪mark.svg/500px-Service_mark.svg.png
$ sudo chown -Rf geonode: .
```

3. Change to the `css` directory:

```
$ cd ../../..
```

4. Override the CSS that displays the logo by editing `<my_geonode>/static/css/site_base.css` with your favorite editor and adding the following lines, making sure to update the width, height, and URL to match the specifications of your image.

```
$ sudo vi site_base.css
```

```
.navbar-brand {
  width: 350px;
  height: 80px;
  background: transparent url("../img/500px-Service_mark.svg.png") no-repeat;
  background-size: 300px 70px;
  background-position-y: center;
}
```

5. Restart your GeoNode project and look at the page in your browser:

```
$ cd /home/geonode
$ sudo rm -Rf geonode/geonode/static_root/*
$ cd my_geonode
$ python manage.py collectstatic
$ sudo service apache2 restart
```

Nota: It is a good practice to cleanup the **static_folder** and the Browser Cache before reloading in order to be sure that the changes have been correctly taken and displayed on the screen.

Visit your site at <http://localhost/> or the remote URL for your site.

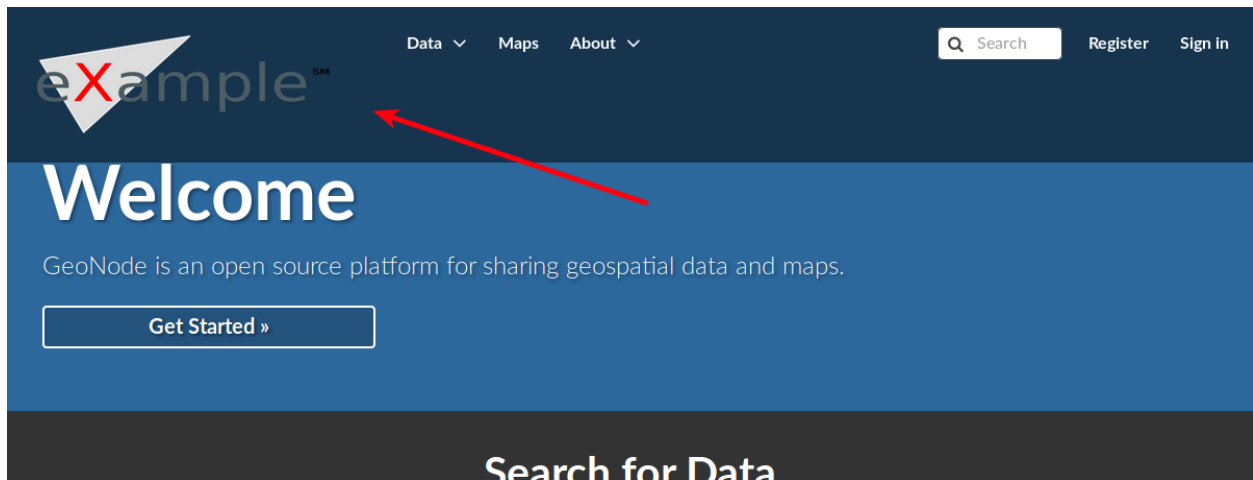


Fig. 267: Custom logo

You can see that the header has been expanded to fit your graphic. In the following sections you will learn how to customize this header to make it as you want.

Nota: You should commit these changes to your repository as you progress through this section, and get in the habit of committing early and often so that you and others can track your project on GitHub. Making many atomic commits and staying in sync with a remote repository makes it easier to collaborate with others on your project.

Cascading Style Sheets

In the last section you already learned how to override GeoNode's default CSS rules to include your own logo. You are able to customize any aspect of GeoNode's appearance this way. In the last screenshot, you saw that the main area in the homepage is covered up by the expanded header.

First, we'll walk through the steps necessary to displace it downward so it is no longer hidden, then change the background color of the header to match the color in our logo graphic.

1. Reopen `<my_geonode>/static/css/site_base.css` in your editor:

```
$ cd /home/geonode/my_geonode/my_geonode/static/css
$ sudo vi site_base.css
```

1. Add the following CSS rules to consider the expanded header height:

```
#wrap {
  margin-top: 100px !important;
  padding-top: 0px;
}
```

1. Add a rule to change the background color of the header to match the logo graphic:

```
.navbar-inverse {
  background-color: #ff0000 !important;
}
```

1. Add a background image for the *hero* section:

```
.jumbotron {
  background: url("https://cdn.pixabay.com/photo/2017/09/16/16/09/sea-
↪2755908_960_720.jpg") no-repeat !important;
  background-size: cover !important;
}
```

1. Your project CSS file should now look like this:

```
.navbar-brand {
  width: 350px;
  height: 150px;
  background: transparent url("../img/500px-Service_mark.svg.png") no-
↪repeat;
  background-size: 300px 100px;
  background-position-y: center;
}

#wrap {
  margin-top: 100px !important;
  padding-top: 0px;
}

.navbar-inverse {
  background-color: #ff0000 !important;
}

.jumbotron {
  background: url("https://cdn.pixabay.com/photo/2017/09/16/16/09/sea-
↪2755908_960_720.jpg") no-repeat !important;
  background-size: cover !important;
}
```

1. Collect the static files into STATIC_ROOT, restart the development server and reload the page:

```
$ python manage.py collectstatic
$ sudo service apache2 restart
```

You can continue adding rules to this file to override the styles that are in the GeoNode base CSS file which is built from `base.less`.

Nota: You may find it helpful to use your browser's development tools to inspect elements of your site that you want to override to determine which rules are already applied. See the screenshot below.

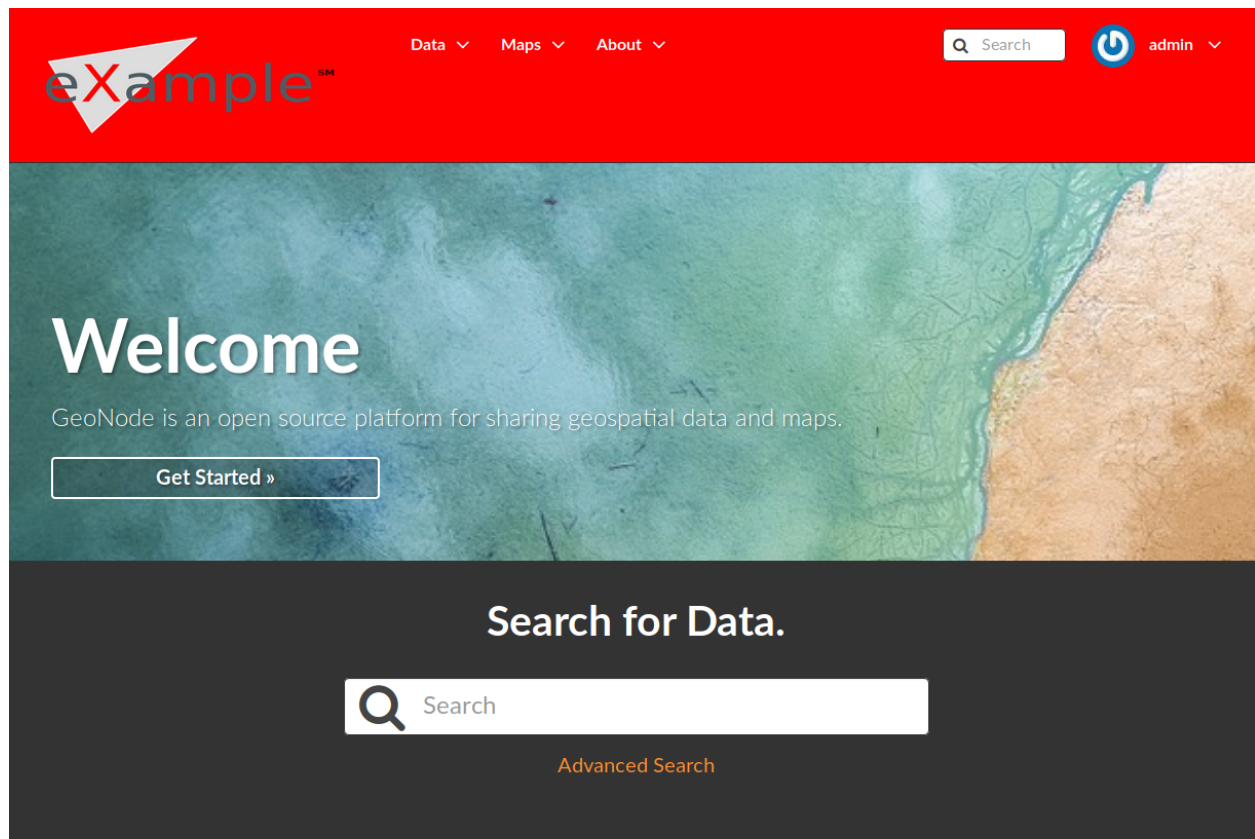


Fig. 268: *CSS override*

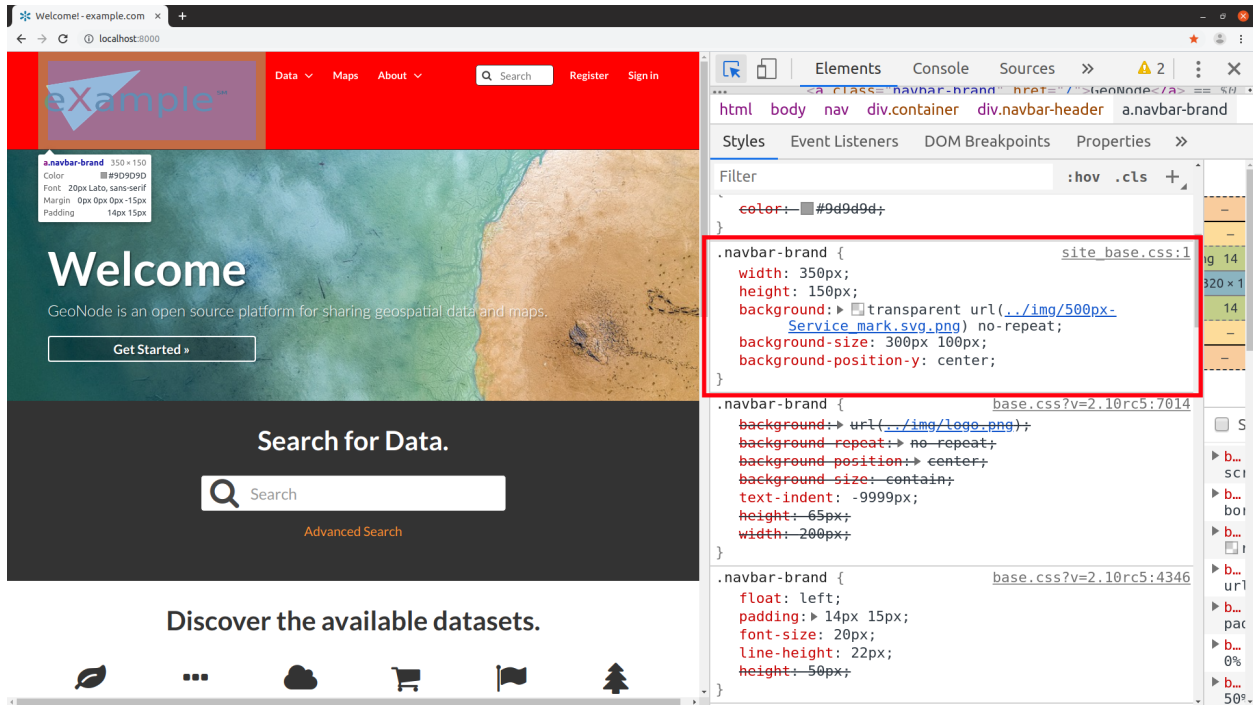


Fig. 269: Screenshot of using browser debugger to inspect the CSS overrides

Templates and static pages

Now that we have changed the default logo and adjusted our main content area to fit the expanded header, the next step is to update the content of the homepage itself. Your GeoNode project includes two basic templates that you will use to change the content of your pages.

The file `site_base.html` (in `<my_geonode>/templates/`) is the basic template that all other templates inherit from and you will use it to update things like the header, navbar, site-wide announcement, footer, and also to include your own JavaScript or other static content included in every page in your site. It's worth taking a look at [GeoNode's base file on GitHub](#). You have several blocks available to you for overriding, but since we will be revisiting this file in future sections of this guide, let's just look at it for now and leave it unmodified.

Open `<my_geonode>/templates/site_base.html` in your editor:

```
$ cd /home/geonode/my_geonode/my_geonode/templates
$ sudo vi site_base.html
```

You will see that it extends from `base.html`, which is the GeoNode template referenced above and it currently only overrides the `extra_head` block to include our project's `site_base.css` which we have modified in the previous section.

```
{% extends "base.html" %}
{% block extra_head %}
    <link href="{{ STATIC_URL }}css/site_base.css" rel="stylesheet"/>
{% endblock %}
```

You can see on [line 189 of the GeoNode base.html template](#) that this block is included in an empty state and is set up specifically for you to include extra CSS files as your project is already set up to do.

The file `site_index.html` is the template used to define your GeoNode project's homepage. Let's actually override this template.

It extends GeoNode's default `index.html` template and gives you the option to override specific areas of the homepage like the *hero area*, but it also allows you leave other sections as they are. You are of course free to override the sections which you prefer, the following steps give you an example.

1. Open `<my_geonode>/templates/site_index.html` in your editor.
2. Edit the first `<h1>` element inside the `<div class="container">` to say something other than «Welcome»:

```
<h1>{{custom_theme.jumbotron_welcome_title|default:_("GeoNode Project Example  
↪") }}</h1>
```

Aviso: Pay attention to the `custom_theme.jumbotron_welcome_title` part, if you delete it you will cannot use the «admin-based» theme customization option (see [Simple Theming](#))

3. Edit the introductory paragraph to say something about your GeoNode project:

```
<p>  
  <p>{{custom_theme.jumbotron_welcome_content|default:_("This GeoNode has_  
↪been customized through my GeoNode Project.")}}</p>  
</p>
```

Aviso: Take care of the `custom_theme.jumbotron_welcome_content` if you are using the «admin-based» theme customization option (see [Simple Theming](#))

4. Your edited `site_index.html` file should now look like this:

```
{% extends 'index.html' %}  
{% load i18n %}  
  
{% comment %}  
  This is where you can override the hero area block. You can simply_  
↪modify the content below or replace it wholesale to meet your own needs.  
{% endcomment %}  
  
{% block hero %}  
  <div class="jumbotron">  
    <div class="container">  
      <h1>{{custom_theme.jumbotron_welcome_title|default:_("GeoNode_  
↪Project Example") }}</h1>  
      <p></p>  
      <p>{{custom_theme.jumbotron_welcome_content|default:_("This_  
↪GeoNode has been customized through my GeoNode Project.")}}</p>  
      {% if not custom_theme.jumbotron_cta_hide %}  
        <p>  
          <a class="btn btn-default btn-lg" target="_blank" role=  
↪"button"  
            href="{{custom_theme.jumbotron_cta_link|default:_(  
↪'http://docs.geonode.org/en/master/usage/index.html')}}">  
            {{custom_theme.jumbotron_cta_text|default:_("Get_  
↪Started &raquo;") }}  
        </p>  
      </div>  
    </div>  
  </div>  
{% endblock %}
```

(continues on next page)

(continuação da página anterior)

```

        </a>
      </p>
    {% endif %}
  </div>
</div>
{% endblock %}

```

5. Collect the static files into STATIC_ROOT, restart the development server and reload the page to see the changes:

```

$ python manage.py collectstatic
$ sudo service apache2 restart

```

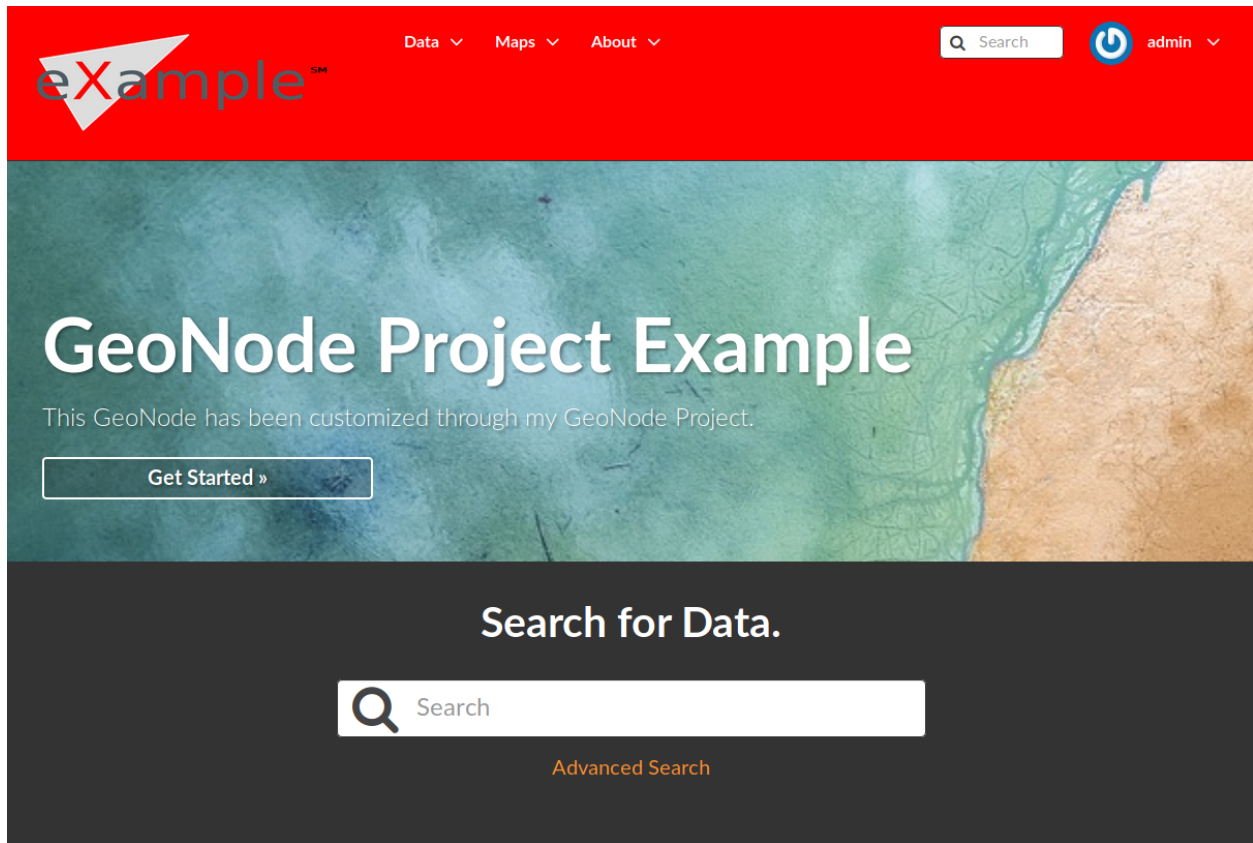


Fig. 270: Customized Geonode Project Home Page

Other theming options

You are able to change any specific piece of your GeoNode project's style by adding CSS rules to `site_base.css`, but since GeoNode is based on Bootstrap, there are many pre-defined themes that you can simply drop into your project to get a whole new look. This is very similar to [WordPress](#) themes and it is a powerful and easy way to change the look of your site without much effort.

Bootswatch

From [Bootswatch](#) you can download ready-to-use themes for Bootstrap-based website.

Aviso: Currently GeoNode uses the 3.3.7 version of Bootstrap, so [suitable Bootswatch themes](#) should have been built for the same version.

The following steps will show you how to use a theme from Bootswatch in your own GeoNode Project.

1. Download the [Bootswatch themes for Bootstrap v3.3.7 archive](#) and extract it on some folder in your disk.
2. Select a theme (in this example we will use *Sandstone*) and copy the `bootstrap.css` file inside the theme folder to the `<my_geonode>/static/css` (the static folder of your GeoNode Project).
3. Update the `site_base.html` template to include this file. It should now look like this:

```
$ cd <my_geonode>/templates
$ sudo vi site_base.html
```

```
{% extends "base.html" %}
{% block extra_head %}
    <link href="{{ STATIC_URL }}css/site_base.css" rel="stylesheet"/>
    <link href="{{ STATIC_URL }}css/bootstrap.css" rel="stylesheet"/>
{% endblock %}
```

5. Collect the static files into `STATIC_ROOT`, restart the development server and reload the page:

```
$ python manage.py collectstatic
$ sudo service apache2 restart
```

1.15 GeoNode permissions

1.15.1 Permissions

Permissions in GeoNode are set per resource, where a resource can be a layer, a map, a document or a service. The way the permissions are set is the same for all of them.

Aviso: GeoNode has a set of default permissions that are applied on resource creation **when** you don't explicitly declare them. This is particularly relevant when creating and saving a map, where you won't have the possibility to set its permissions during the creation phase. GeoNode can be tuned to make sure that by default the new created resource are not public, this can be done by changing two settings, see [Default view permissions](#) and [Default download permissions](#)

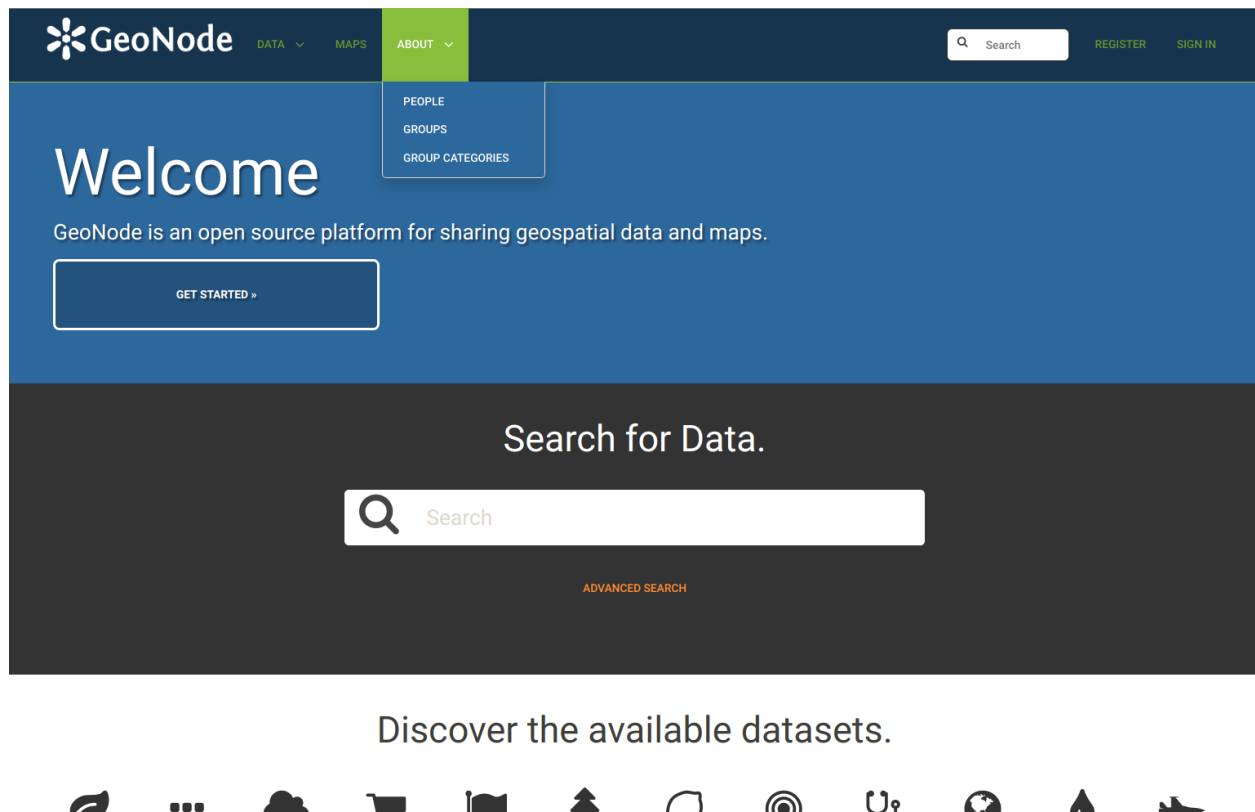


Fig. 271: *Bootswatch Theme for the Geonode Project*

Single Resource permissions

Resource permissions can be generally set from the *resource detail* page. The following figure shows how to open the dialog to set permissions on a layer, the same concept applies to documents and maps.

The dialog for setting the permission allow a granular selection of each permission type to be applied for users and/or groups, each permission type is grouped in tabs that are expanded on click.

The text boxes have an autosuggest feature to help the compilation of user names and groups, it starts upon typing.

You can set the following types of permissions:

- *View* allows to view the layer;
- *Download* allows to download the layer;
- *Change Metadata* allows to change the layer metadata;
- *Edit Data* allows to change attributes and properties of the layers features;
- *Edit Style* allows to change the layer style;
- *Manage* allows to update, delete, change permissions, publish and unpublish the layer.

Aviso: When assigning permissions to a group, all the group members will have those permissions. Be careful in case of editing permissions.

Geo Limits permissions

Nota: This feature is available **only** when enabling ``GeoServer`` as geospatial backend. Also make sure that the properties ``GEONODE_SECURITY_ENABLED``, ``GEOFENCE_SECURITY_ENABLED`` and ``GEOFENCE_URL`` are correctly set for the ``OGC_SERVER``.

Geo Limits are an extension of the GeoNode standard permissions. *Geo Limits* allows the owner of the resource, or the administrator, to restrict users or groups to a specific geographical area, in order to limit the access to the layer to only the portions contained within that geographic restriction, excluding data outside of it.

In order to be able to set *Geo Limits* you must be an ``administrator`` of the system or the ``owner`` of the resource or you must have ``Manage Permissions`` rights to the resource.

Go to the *Layer Details* page and scroll down to the *Change Layer Permissions* button, as we have seen on the previous section.

If you have the permissions to set the *Geo Limits*, you should be able to see the limits tab beside the permissions one.

You should be able to see an interactive preview of the layers along with few small drawing tools, that allow you to start creating limits on the map manually if you want.

Moreover at the bottom of the panel, there are two other tabs, one listing the available *Users* and another one listing the available *Groups*.


[Data](#)
[Maps](#)
[About](#)

John Smith


[Info](#)
[Attributes](#)
[Share](#)
[Ratings](#)
[Comments](#)
[Favorite](#)

Title Italian Towers

License Not Specified

Abstract No abstract provided

Publication Date June 7, 2019, 4:03 a.m.

Type Vector Data

Keywords features, italian_towers

Category Environment

Regions Global

Owner johnsmith

[More info](#) -

[Layer WMS GetCapabilities document](#)

[Download Layer](#)
[Metadata Detail](#)
[Editing Tools](#)
[View Layer](#)
[Download Metadata](#)

Legend

Default Point

- Red Square

Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

[Create a Map](#)

Add the layer to an existing map

Regioni Italiane Map

Click the button below to add the layer to the selected map.

[Add to Map](#)

Styles

The following styles are associated with this layer. Choose a style to view it in the preview map.

(default style) Default Point

Refresh Attributes and Statistics of this layer

Click the button below to allow GeoNode refreshing the list of available Layer Attributes. If the option 'WPS_ENABLED' has been also set on the backend, it will recalculate their statistics too.

[Refresh Attributes and Statistics](#)

Clear the Server Cache of this layer

Click the button below to wipe the tile-cache of this layer.

[Empty Tiled-Layer Cache](#)

Permissions

Click the button below to change the permissions of this layer.

[Change Layer Permissions](#)

About

Owner, Point of Contact, Metadata Author

johnsmith
John Smith Foundation

Fig. 272: Change Layer Permissions

Set permissions for this resource



Who can view it?
☐ Anyone
The following users:

✖ admin

The following groups:

Choose groups...

Who can download it?
☐ Anyone
The following users:

✖ admin

The following groups:

Choose groups...

Who can change metadata for it?

Who can edit data for this layer?

Who can edit styles for this layer?

Who can manage it? (update, delete, change permissions, publish/unpublish it)

Cancel

Apply Changes

Fig. 273: Resource Permission Dialogue

Set permissions for this resource

People and Groups

Geo Limits

Who can view it?

☒ Anyone

The following users:

× admin

The following groups:

Who can download it?

Who can change metadata for it?

Who can edit data for this layer?

Who can edit styles for this layer?

Who can manage it? (update, delete, change permissions, publish/unpublish it)

Cancel

Apply Changes

Fig. 274: Geo Limits Tab

Aviso: You will be able to access only *Public* groups and the *Private* ones you belong to.

For each entry of the *Users* and *Groups* tabs, you will have 3 options:

1. Load *Geo Limits*

This button allows you to load the *Geo Limits* already stored on the DB.

Aviso: By clicking this button, the geometries present into the map will be cleared. You can add/remove more geometries later on if needed.

2. Upload *Geo Limits*

This button allows you to upload the *Geo Limits* from a ``SHAPEFILE`` on your hard disk. This button **won't** save anything yet. It will **only** load the geometries into the map.

Aviso: Be careful using big ``SHAPEFILES``. The geometries will be loaded in memory, and your browser might slow down a lot if you load huge / complex geometries.

Aviso: By clicking this button, the geometries present into the map will be cleared. You can add/remove more geometries later on if needed.

3. Save *Geo Limits*

This button allows you to store the *Geo Limits* into the DB. The geometries will be associated to the current ``resource`` and selected ``user`` or ``group``.

Nota: By saving the geometries into the DB, the geospatial restrictions won't be applied yet. In order to apply the restrictions you need to:

- a) Set the general permissions to the user / group on the general *Permissions* dialog.
- b) Click on *Apply Changes* button

See the next paragraph for more details.

Once you finished editing your geometries, save them into the DB.

What you have to do now, in order to apply the *Geo Limits* correctly, is to go back to the *Permissions* tab and select *View* and / or *Download* permissions for the users / groups you want to apply the restrictions.

When you are happy with your changes, click on *Apply Changes* button.

The user ``afabiani`` won't be able from now on to access the whole layer data.

Aviso: The *Geo Limits* will be persisted on GeoNode DB for that resource. That means that everytime you will update the general permissions, also the geospatial restrictions will be applied.

In order to remove the *Geo Limits* for a certain user or group, you can just *Save* an **empty geometry**. This will **delete** the entry from the DB also.

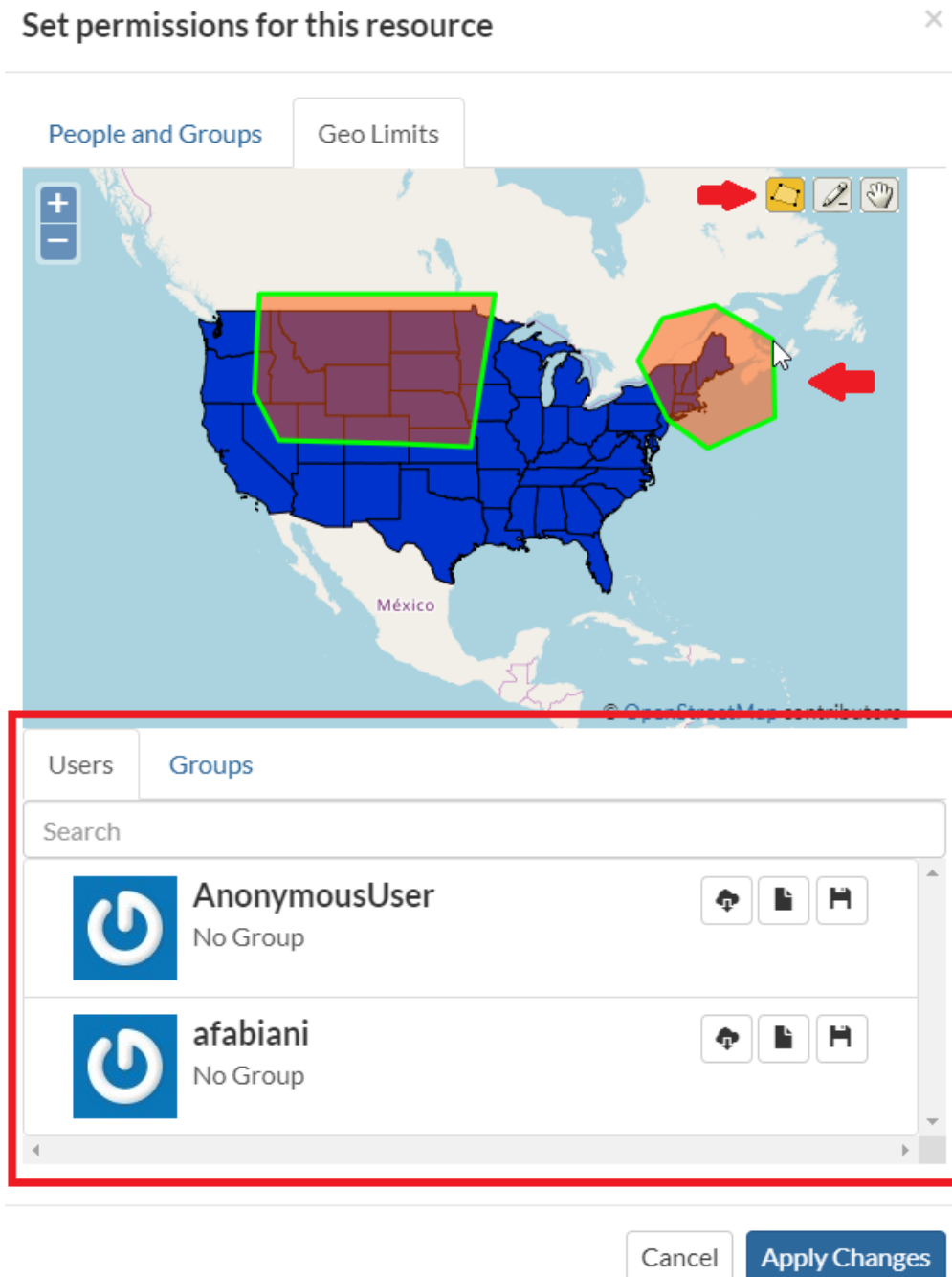


Fig. 275: *Geo Limits: Preview Window with Drawing Tools*

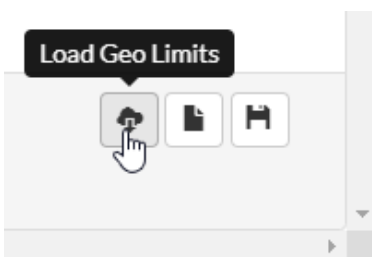


Fig. 276: *Geo Limits: Load from DB*

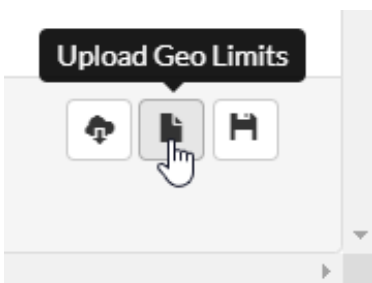


Fig. 277: *Geo Limits: Upload from a SHAPEFILE*

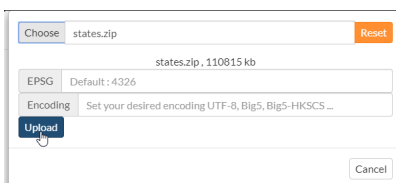


Fig. 278: *Geo Limits: Upload from a SHAPEFILE*

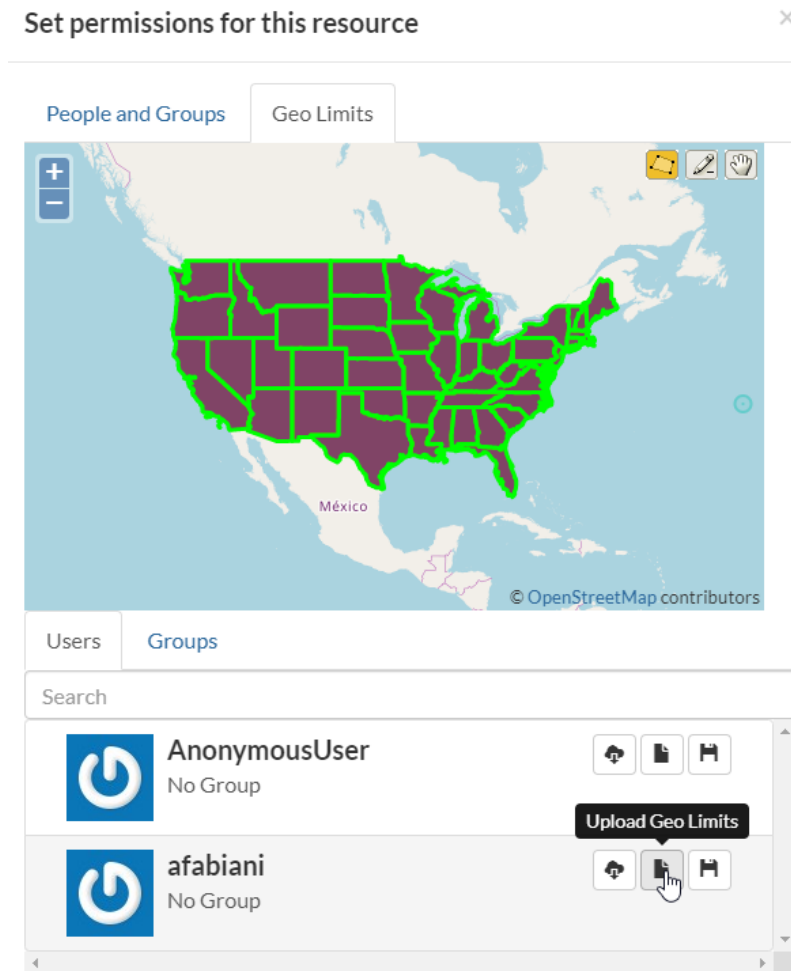


Fig. 279: Geo Limits: Upload from a SHAPEFILE

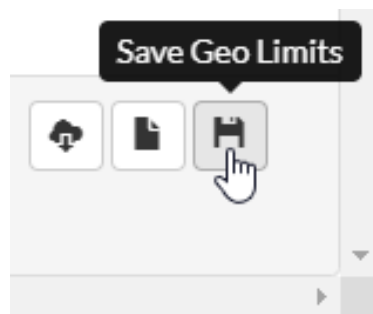
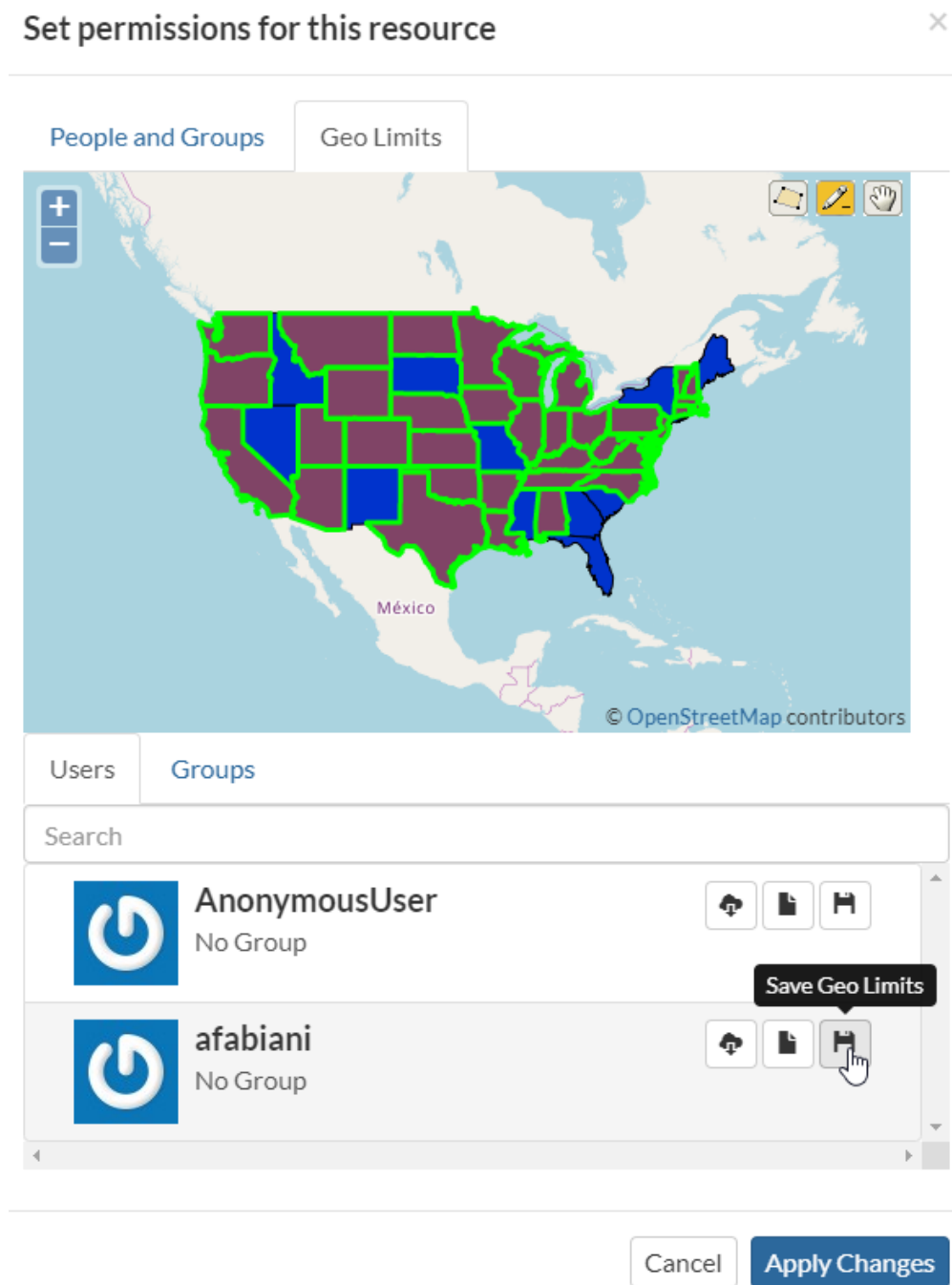


Fig. 280: Geo Limits: Store the Geo Limits into the DB

Fig. 281: *Geo Limits: Editing the Geometries*

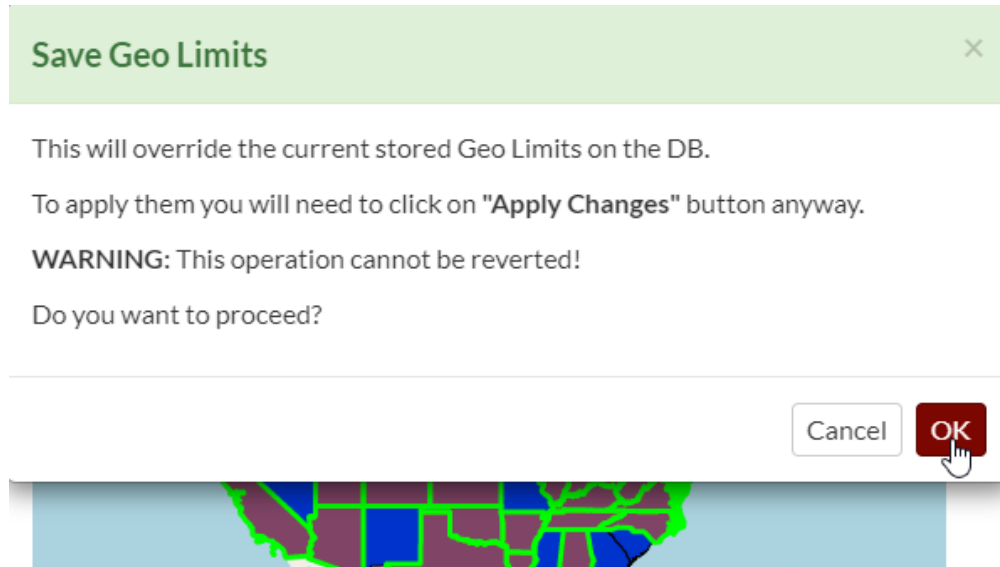


Fig. 282: *Geo Limits: Saving the Geometries for the user afabiani*

Bulk permissions

GeoNode offers the possibility to set permissions in bulk, this can be done in any *list* page.

In order to set bulk permissions you have first to fill the *shopping cart* with the resources you are interested with by clicking the + button on the resource snippet.

Once happy with the selection you can click the *Set Permissions* button under the shopping cart to open the permissions dialogue that will apply the chosen permission to all selected resources.

1.16 Read-Only and Maintenance Mode

1.16.1 Read-Only and Maintenance Modes

Overview

GeoNode gives an option to operate in different modes, according to the needs and demands of the certain application system.

Changing the currently used mode can be done in the admin panel by the user with super-user privileges, by modifying `Configuration` singleton model in the `BASE` application:

Set permissions for this resource ×

People and Groups

Geo Limits

Who can view it?

☒ Anyone

The following users:

× admin

afa

afabiani

Who can download it?

Who can change metadata for it?

Who can edit data for this layer?

Who can edit styles for this layer?

Who can manage it? (update, delete, change permissions, publish/unpublish it)

Cancel

Apply Changes

Fig. 283: *Geo Limits: Set View/Download Permissions for the user afabiani*

Who can download it?

Who can change metadata for it?

Who can edit data for this layer?

Who can edit styles for this layer?

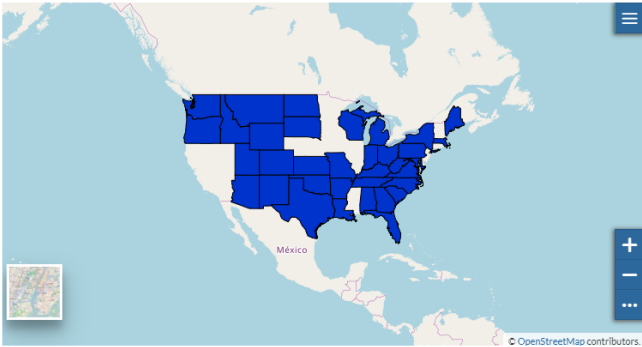
Who can manage it? (update, delete, change permissions, publish/unpublish it)

Cancel Apply Changes

Fig. 284: Geo Limits: Apply Permissions and Restrictions to the users and groups

GeoNode Data Maps About Search afabiani

states



Download Layer

Metadata Detail

View Layer

Download Metadata

Legend

A azure polygon style

■ azure polygon

Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

Info Attributes Share Ratings Comments Favorite

Title states

License Not Specified

Fig. 285: Geo Limits: Geospatial restrictions applies for the user afabiani



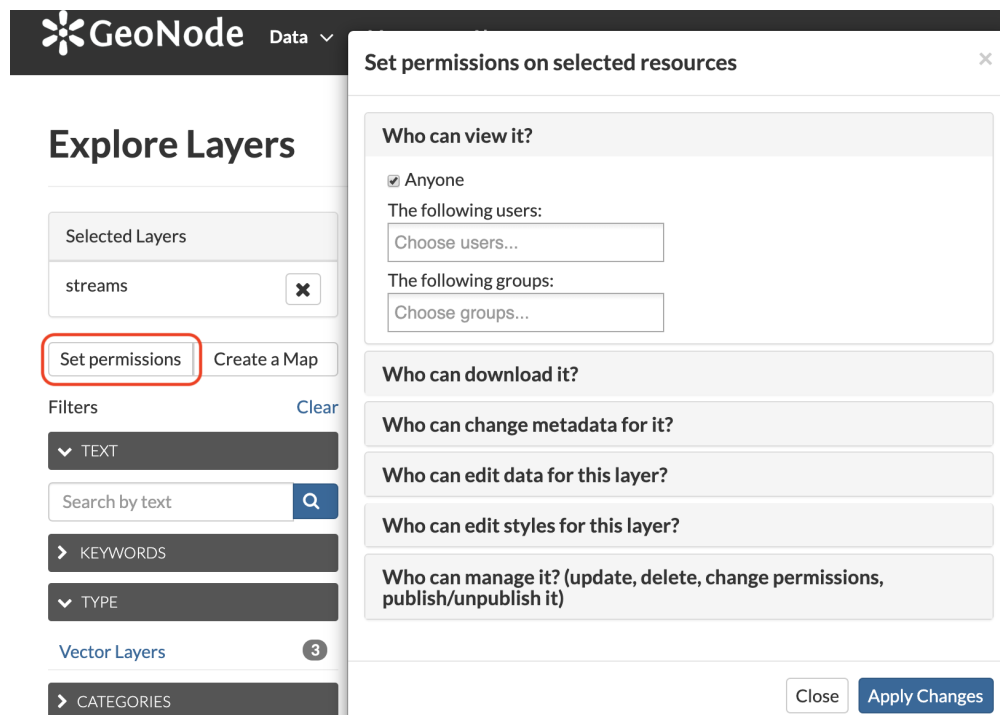
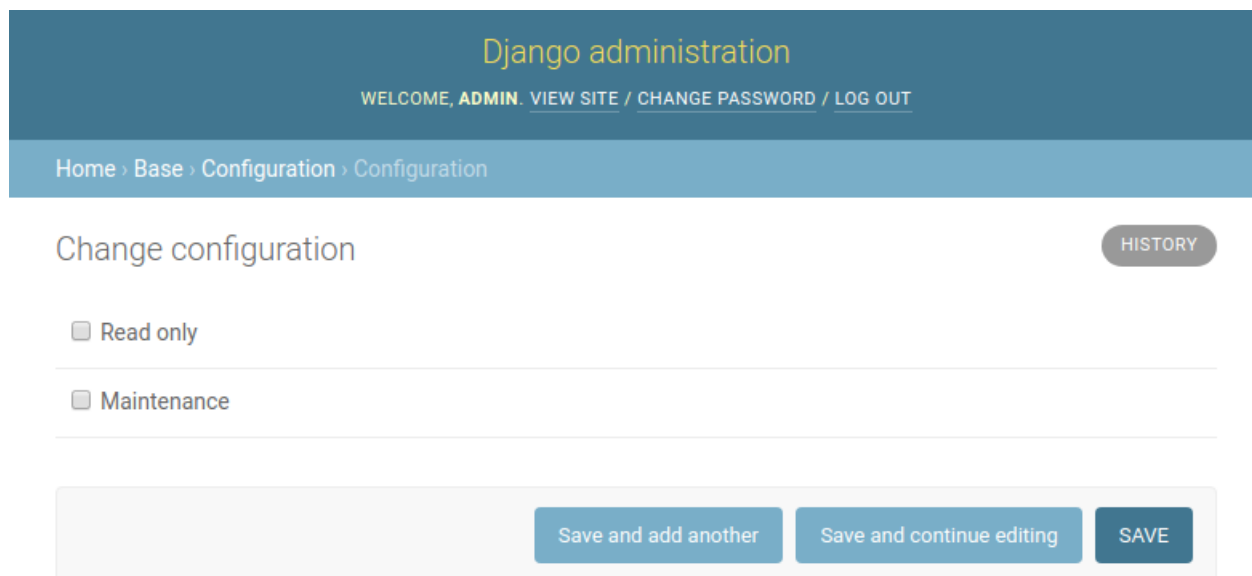
streams

No abstract provided.

simone 12 Jun 2019 0 0 0 Create a Map

+

Fig. 286: Add Resource To Shopping Cart

Fig. 287: *Sopping Cart Permissions*Fig. 288: *Configuration change in the admin panel*

Read-Only Mode

Activating the Read-Only Mode (by setting `Read only True` in the `Configuration`) activates a middleware rejecting all modifying requests (POST/PUT/DELETE), with an exception for:

- POST to login view
- POST to logout view
- POST to admin login view
- POST to admin logout view
- all requests to OWS endpoint
- all requests ordered by a super-user

Additionally, all UI elements allowing modifying GeoNode's content are hidden, so e.g. the button «Upload Layer» is not rendered in the templates.

In case a user tries to perform a forbidden request, they will be presented with a static HTML page informing them, the GeoNode is in the Read-Only mode and this action is currently forbidden.

Maintenance Mode

Activating the Maintenance Mode (by setting `Maintenance True` in the `Configuration`) activates the highest level middleware (the one executed as the first) rejecting all requests to the GeoNode instance, with an exception for:

- POST to admin login view
- POST to admin logout view
- all requests ordered by a super-user

In case a user tries to perform any request against the GeoNode (including GET requests), they will be presented with a static HTML page informing them, the maintenance actions are taken on the GeoNode instance, and asking them to try again soon.

The maintenance mode was implemented with a thought of the backup and restore procedures without a necessity to put down the instance, but at the same time with a restriction of any outer interference.

1.17 Monitoring

1.17.1 Monitoring

Internal Monitoring Application (`geonode.monitoring`)

Nota: This application requires MaxMind's GeoIP database file.

Base concepts and objects

GeoNode monitoring is a configurable monitoring application, that allows internal resources and hardware resources monitoring for GeoNode installations, including GeoServer deployments.

Monitoring application is configurable, so different deployment scenarios could be handled - from GeoNode and GeoServer running on single host, through distributed installations, where GeoServer is deployed to several hosts.

Monitoring application uses three base entity classes to describe elements of reality: `Host`, `Service Type` and `Service`.

- `Host` is an object describing physical (or virtual) instance of operating system on which GN or GS is running. This object exists only for grouping and is not used directly by monitoring.
- `Service Type` is a description of kind of `Service`. Depending on service type, different metrics are stored, and different data collection mechanisms are used. Additionally, for system monitoring, it's not conducted directly, but with GeoNode or GeoServer as monitoring agent. That means, no additional software installation is needed to monitor system, but also, hosts that don't have GeoNode or GeoServer installed, won't be monitored. There are four service types:
 - `hostgeonode`, `hostgeoserver` - those types describe system monitoring probes that are running with GeoNode or GeoServer respectively,
 - `geonode`, `geoserver` - application-level probes that monitor one specific GeoNode or GeoServer instance.
- `Service` describes one specific instance of probe, either host-level or application-level. Service references `Host` and `Service Type`. Each service must be named, and name should be system-wide unique.

As mentioned above, each `Service Type` keeps a set of `metrics`, specific for that type. A `metric` is a description of measured value, for example: number of requests, response size or time, cpu usage, free memory etc. Each `Service Type` has it's own metrics set. Metric value may be either value counter (like country of user), numeric counter (like number of requests) or rate (like bytes in/out on network interface).

Besides metric data, monitoring will also store exception information for exceptions that were captured during request handling.

Data are collected periodically (at most every 1 minute), aggregated and stored in aggregated form. User can see data from predefined relative periods (last minute, last 10 minutes, last hour, last day, last week).

User can enable and configure automated checks, which will be run after each collection/aggregation cycle, and will emit notifications if metric values in that run exceed configured thresholds.

Analytics

GeoNode monitoring application makes also available information about resources usage at user level.

Those information are collected whenever an event occurs about some resource. Events can be of different types (`EventType`) which refer to common user activities on resources (upload, view, download, etc.). Those data are stored using a dedicated `metric` and aggregated based on a configurable granularity, depending on the time interval considered and the wanted resolution.

So the analytics client, once defined a time interval and a time frame, can retrieve stats such as:

- total number of unique visitors;
- number of unique visitors who trigger a specific type of event;
- number of unique visitors who trigger events on some resource type;
- number of unique visitors in a given country;

- number of unique visitors who trigger events on some specific resource;
- number of unique visitors considering a combination of multiple conditions (for example an event type on some resource type).

Installation

Aviso: This plugin requires a Potgresql DB backend enabled

- ensure UTC Timezone to your DB

```
psql -c 'set timezone=UTC;'
```

- enable `MONITORING_ENABLED` flag and ensure that following code is in your settings:

```
# Settings for MONITORING plugin
CORS_ORIGIN_ALLOW_ALL = ast.literal_eval(os.environ.get('CORS_ORIGIN_ALLOW_ALL',
↪'False'))
GEOIP_PATH = os.getenv('GEOIP_PATH', os.path.join(PROJECT_ROOT, 'GeoIPCities.dat'))
MONITORING_ENABLED = ast.literal_eval(os.environ.get('MONITORING_ENABLED', 'True'))

MONITORING_CONFIG = os.getenv("MONITORING_CONFIG", None)
MONITORING_HOST_NAME = os.getenv("MONITORING_HOST_NAME", HOSTNAME)
MONITORING_SERVICE_NAME = os.getenv("MONITORING_SERVICE_NAME", 'local-geonode')

# how long monitoring data should be stored
MONITORING_DATA_TTL = timedelta(days=int(os.getenv("MONITORING_DATA_TTL", 7)))

# this will disable csrf check for notification config views,
# use with caution - for dev purpose only
MONITORING_DISABLE_CSRF = ast.literal_eval(os.environ.get('MONITORING_DISABLE_CSRF',
↪'False'))

if MONITORING_ENABLED:
    if 'geonode.monitoring' not in INSTALLED_APPS:
        INSTALLED_APPS += ('geonode.monitoring',)
    if 'geonode.monitoring.middleware.MonitoringMiddleware' not in MIDDLEWARE_CLASSES:
        MIDDLEWARE_CLASSES += \
            ('geonode.monitoring.middleware.MonitoringMiddleware',)

# skip certain paths to not to mud stats too much
MONITORING_SKIP_PATHS = ('/api/o/',
                          '/monitoring/',
                          '/admin',
                          '/jsi18n',
                          STATIC_URL,
                          MEDIA_URL,
                          re.compile('^[a-z]{2}/admin/'),
                          )

# configure aggregation of past data to control data resolution
# list of data age, aggregation, in reverse order
# for current data, 1 minute resolution
# for data older than 1 day, 1-hour resolution
# for data older than 2 weeks, 1 day resolution
```

(continues on next page)

(continuação da página anterior)

```

MONITORING_DATA_AGGREGATION = (
    (timedelta(seconds=0), timedelta(minutes=1)),
    (timedelta(days=1), timedelta(minutes=60)),
    (timedelta(days=14), timedelta(days=1)),
)

# privacy settings
USER_ANALYTICS_ENABLED = ast.literal_eval(os.getenv('USER_ANALYTICS_ENABLED', 'False
↳'))

```

- run

```

DJANGO_SETTINGS_MODULE=<project_name>.settings python manage.py migrate_
↳monitoring

```

to apply db schema changes and insert initial data

- run

```

DJANGO_SETTINGS_MODULE=<project_name>.settings python manage.py_
↳updategeoip

```

to fetch MaxMind's GeoIP database file. It will be written to path specified by *GEOIP_PATH* setting.

- run

```

DJANGO_SETTINGS_MODULE=<project_name>.settings python manage.py collect_
↳metrics -n -t xml -f --since='<yyyy-mm-dd HH:mm:ss>'

```

to create first metrics.

Aviso: Replace `<yyyy-mm-dd HH:mm:ss>` with a real date time to start with.

- update Sites from admin; make sure it contains a correct host name
- do not forget to enable notifications and configure them from user profile

Enable the collect_metrics cron

Aviso: Here below you will find instructions for a Ubuntu 16.04/18.04 based machine, but the procedure is similar for other OSs. The basic concept is that you must allow the system to run the command every minute (**without -f and since**):

```

DJANGO_SETTINGS_MODULE=<project_name>.settings python manage.py collect_metrics -n -
↳t xml

```

crontab job

```
sudo crontab -e
```

```
# Add the following line at the bottom; this will run the supervisor command every_
↪minute
* * * * * supervisorctl start geonode-monitoring
```

supervisor

```
sudo apt install supervisor
sudo service supervisor restart
sudo update-rc.d supervisor enable
```

```
sudo vim /etc/supervisor/conf.d/geonode-monitoring.conf
```

```
[program:geonode-monitoring]
command=<path_to_virtualenv>/geonode/bin/python -W ignore <path_to_your_project>/
↪geonode/manage.py collect_metrics -n -t xml
directory = <path_to_your_project>
environment=DJANGO_SETTINGS_MODULE="<your_project>.settings"
user=<your_user>
numproc=1
stdout_logfile=/var/log/geonode-celery.log
stderr_logfile=/var/log/geonode-celery.log
autostart = true
autorestart = true
startsecs = 10
stopwaitsecs = 600
priority = 998
```

```
sudo service supervisor restart
sudo supervisorctl start geonode-monitoring
sudo supervisorctl status geonode-monitoring
```

```
sudo vim /etc/hosts
```

```
127.0.0.1      localhost
<public_ip>   <your_host.your_domain> <your_host>

# The following lines are desirable for IPv6 capable hosts
```

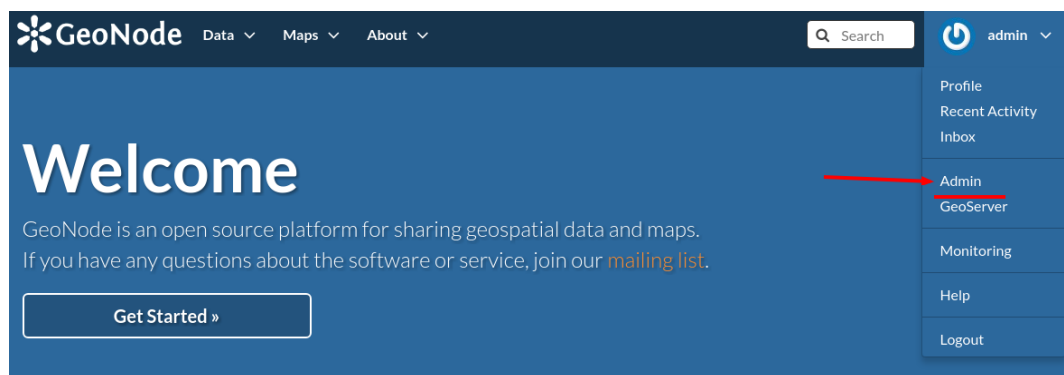
Configuration

In order to have working monitoring, at least `Service` should be configured. Let's assume following deployment scenario:

- there's one machine, `geo01`
- `geo01` hosts both GeoNode and GeoServer (including PostgreSQL).
- applications are served with `nginx+uwsgi`, on port 80, but they are reachable on `localhost` address.
- GeoServer is served from `/geoserver/` path
- GeoNode is served from `/` path

Here's step-by-step instruction how to create monitoring setup for deployment scenario:

1. Log in as admin, and go to admin section:



2. Go to **monitoring** section (or type `/admin/monitoring/` as a path in URL):

Layers	
Attributes	+ Add ✎ Change
Layers	+ Add ✎ Change
Styles	+ Add ✎ Change
Upload sessions	+ Add ✎ Change
Maps	
Map layers	+ Add ✎ Change
Map snapshots	+ Add ✎ Change
Maps	+ Add ✎ Change
Monitoring	
Exception events	+ Add ✎ Change
Hosts	+ Add ✎ Change
Metric labels	+ Add ✎ Change
Metric notification checks	+ Add ✎ Change
Metrics	+ Add ✎ Change
Monitored resources	+ Add ✎ Change

3. Go to **Hosts**:

Django administration Welcome, admin.

Home > Monitoring

Monitoring administration

Monitoring	
Exception events	+ Add Change
<u>Hosts</u>	+ Add Change
Metric labels	+ Add Change
Metric notification checks	+ Add Change
Metrics	+ Add Change
Monitored resources	+ Add Change
Notification checks	+ Add Change
Notification metric definitions	+ Add Change
Notification receivers	+ Add Change
Ows services	+ Add Change
Request events	+ Add Change
Service type metrics	+ Add Change
Service types	+ Add Change
<u>Services</u>	+ Add Change

4. Click on **Add host +**:

Django administration Welcome, admin. View site / Change password / Log out

Home > Monitoring > Hosts

Select host to change

[Add host +](#)

Action: 0 of 1 selected

5. Enter following information: * **host**: *localhost* * **ip**: *127.0.0.1* Note, that **host** value is arbitrary. You can enter other name if you like. Don't forget to save.

Django administration Welcome, admin. View site / Change password / Log out

Home > Monitoring > Hosts > Host: localhost (127.0.0.1)

Change host History

Name:

Ip:

☒ Active

[✖ Delete](#) [Save and add another](#) [Save and continue editing](#) [Save](#)

6. Go to **Services**:



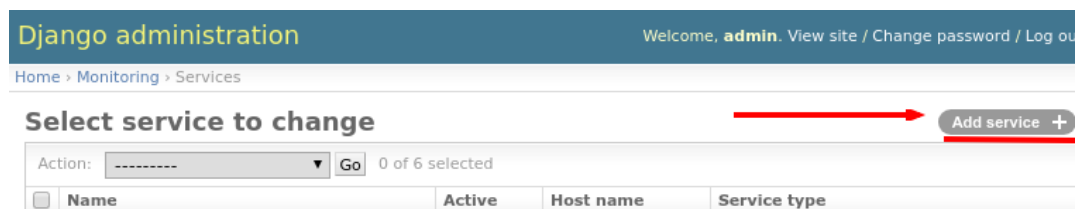
Django administration Welcome, **admin**.

[Home](#) > [Monitoring](#)

Monitoring administration

Monitoring	
Exception events	+ Add ✎ Change
Hosts	+ Add ✎ Change
Metric labels	+ Add ✎ Change
Metric notification checks	+ Add ✎ Change
Metrics	+ Add ✎ Change
Monitored resources	+ Add ✎ Change
Notification checks	+ Add ✎ Change
Notification metric definitions	+ Add ✎ Change
Notification receivers	+ Add ✎ Change
Ows services	+ Add ✎ Change
Request events	+ Add ✎ Change
Service type metrics	+ Add ✎ Change
Service types	+ Add ✎ Change
Services	+ Add ✎ Change

7. Click on **Add service +**:



Django administration Welcome, **admin**. [View site](#) / [Change password](#) / [Log out](#)

[Home](#) > [Monitoring](#) > [Services](#)

Select service to change

[Add service +](#)

Action: 0 of 6 selected

☐	Name	Active	Host name	Service type
--------------------------	------	--------	-----------	--------------

8. Enter following information:

- **name:** *local-geonode*
- **host:** *localhost*
- **service type:** *geonode*

Django administration Welcome, **admin**. [View site](#) / [Change password](#) / [Log o](#)

[Home](#) > [Monitoring](#) > [Services](#) > Service: local-geonode@localhost

Change service History

Name:

Host:  

Check interval:

Last check: **Date:** [Today](#) | 
Time: [Now](#) | 

Service type:  

☒ Active

Notes:

Url:

[✖ Delete](#) [Save and add another](#) [Save and continue editing](#) [Save](#)

9. Add another **Service** Enter following information:

- **name:** *local-system-geonode*
- **host:** *localhost*
- **service type:** *hostgeonode*
- **url:** *http://localhost/* (should point to GeoNode home page)

Django administration Welcome, **admin**. [View site](#) / [Change password](#)

[Home](#) > [Monitoring](#) > [Services](#) > Service: local-system@localhost

Change service

Name:

Host: Host: localhost (127.0.0.1) ▼  

Check interval:

Last check: **Date:** [Today](#) |  **Time:** [Now](#) | 

Service type: Service Type: hostgeonode ▼  

☒ Active

Notes:

Url: **Currently:** <http://localhost/>
Change:

[✖ Delete](#) [Save and add another](#) [Save and continue editing](#)

10. Add another **Service** and enter following information:

- **name:** *local-geoserver*
- **host:** *localhost*
- **service type:** *geoserver*
- **url:** *http://localhost/geoserver/* (should point to GeoServer home page)

Django administration Welcome, **admin**. [View site](#) / [Change password](#)

[Home](#) > [Monitoring](#) > [Services](#) > Service: local-geoserver@localhost

Change service

Name:

Host:  

Check interval:

Last check: **Date:** [Today](#) |  **Time:** [Now](#) | 

Service type:  

☒ Active

Notes:

Url: **Currently:** <http://localhost/geoserver/>
Change:

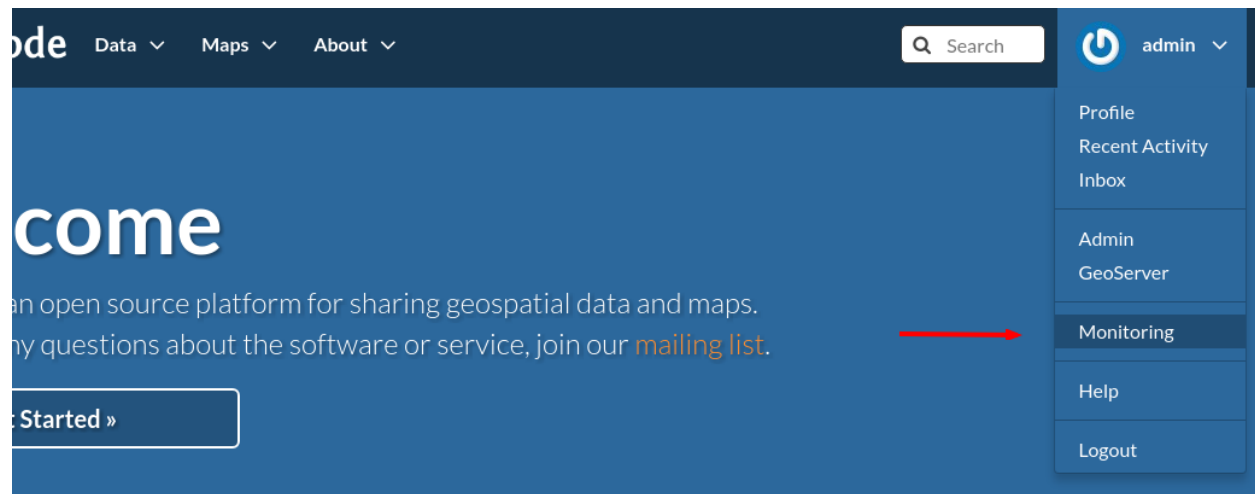
[✖ Delete](#) [Save and add another](#) [Save and continue editing](#)

To summarize, following entries should be created in admin/monitoring:

- Host: localhost, with ip: 127.0.0.1
- **Service: local-geonode:**
 - host localhost
 - type geonode
- **Service: local-geoserver:**
 - url <http://localhost/geoserver/>
 - host localhost
 - type geoserver
- **Service: local-system-geonode**
 - url <http://localhost/>
 - host localhost
 - type hostgeonode

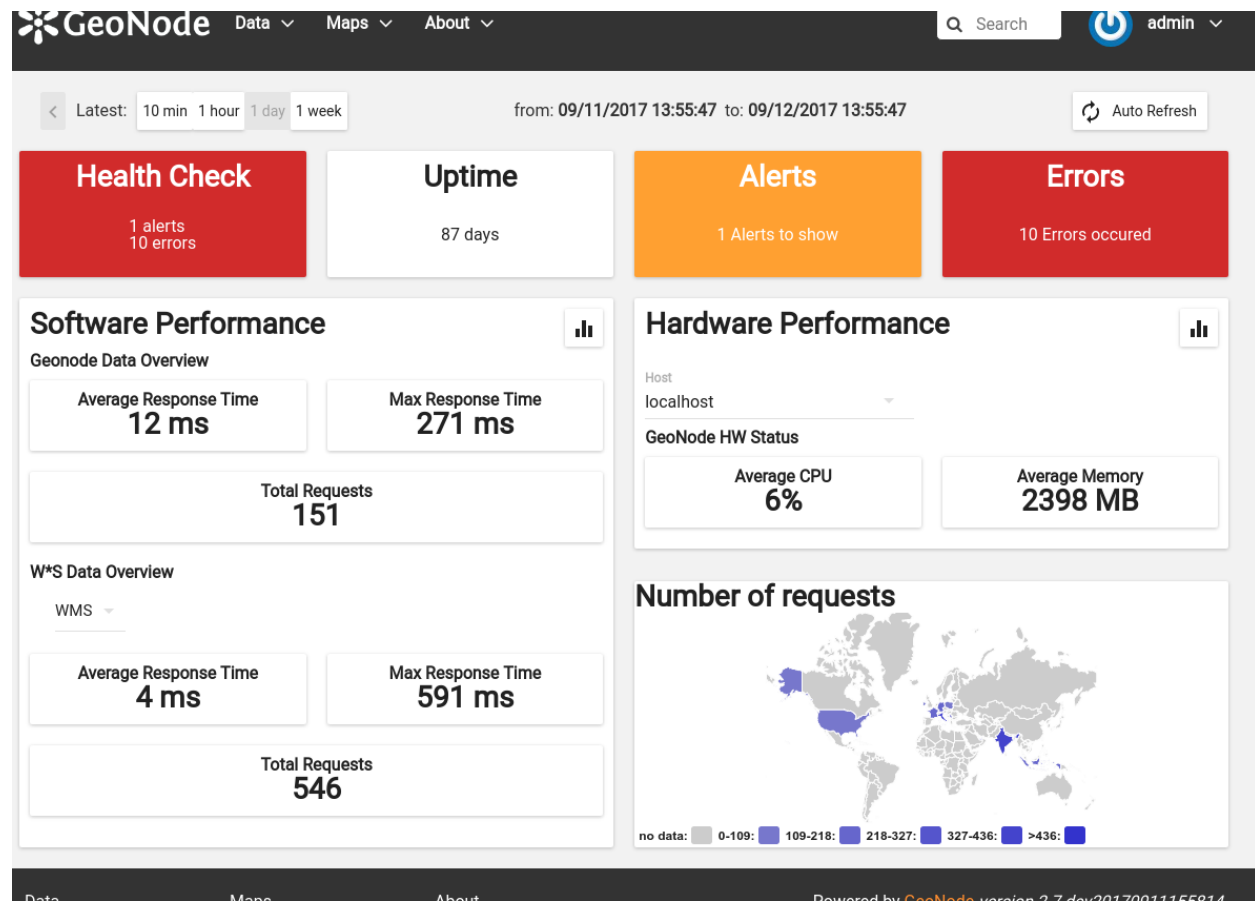
Usage

Monitoring interface is available for superusers only. It's available in profile menu:

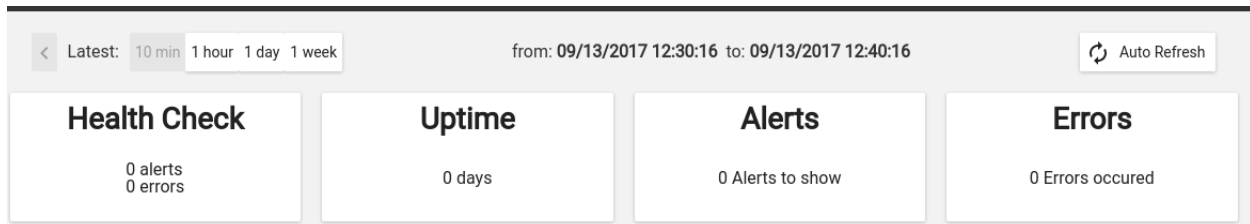


Dashboard

Main view offers overview of recent situation in GeoNode deployment.



Top bar and indicators

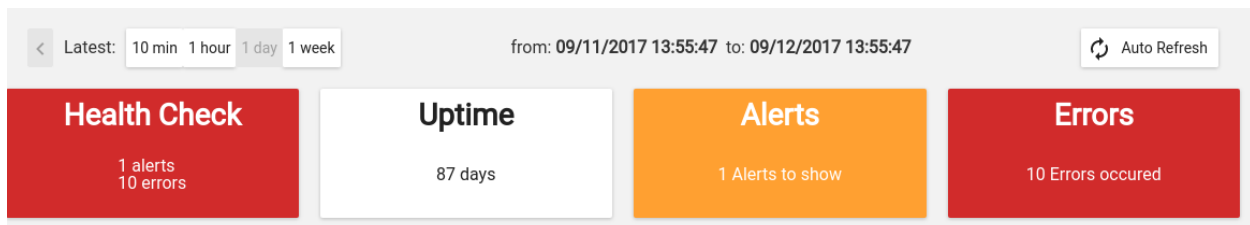


With top bar buttons User can:

- go back from nested interface elements (charts, alerts, errors)
- select time window from which data will be aggregated and shown (last 10 minutes, last 1 hour, last day or last week from now)
- see what's currently used time window
- enable/disable autorefresh

Below there are four main health indicators:

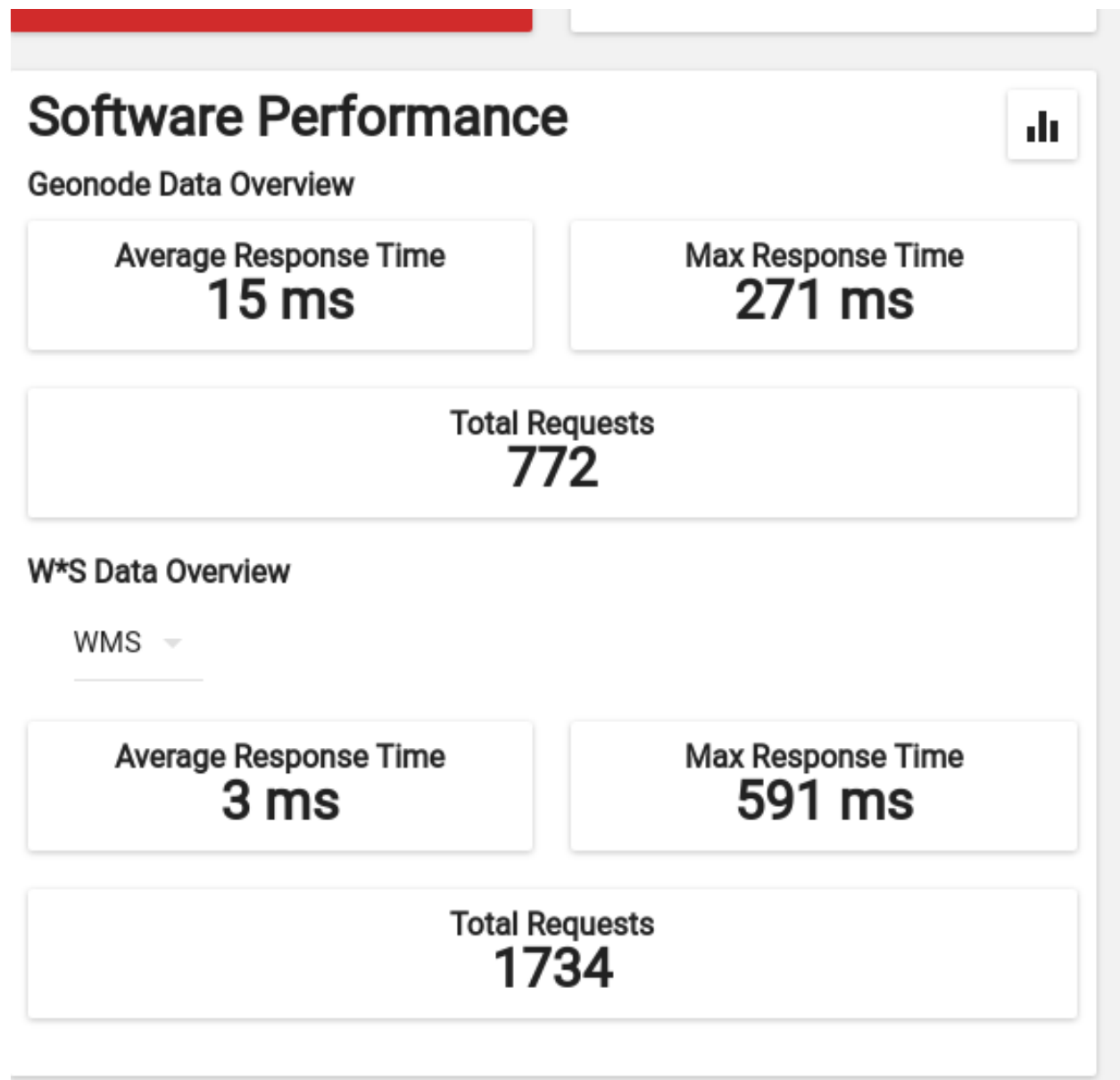
- **aggregated Health Check information.** This element will be:
 - *green* if there is no alerts nor errors
 - *yellow* if there are alerts
 - *red* if there are errors
- **Uptime** that shows GeoNode's system uptime.
- **Alerts** shows number of notifications from defined checks. When clicked, Alerts box will show detailed information . See Notifications description for details.
- **Errors** - shows how many errors were captured during request processing. When clicked, Errors box will show detailed list of captured errors. See Erros description for details.



Indicators in error state

Software Performance

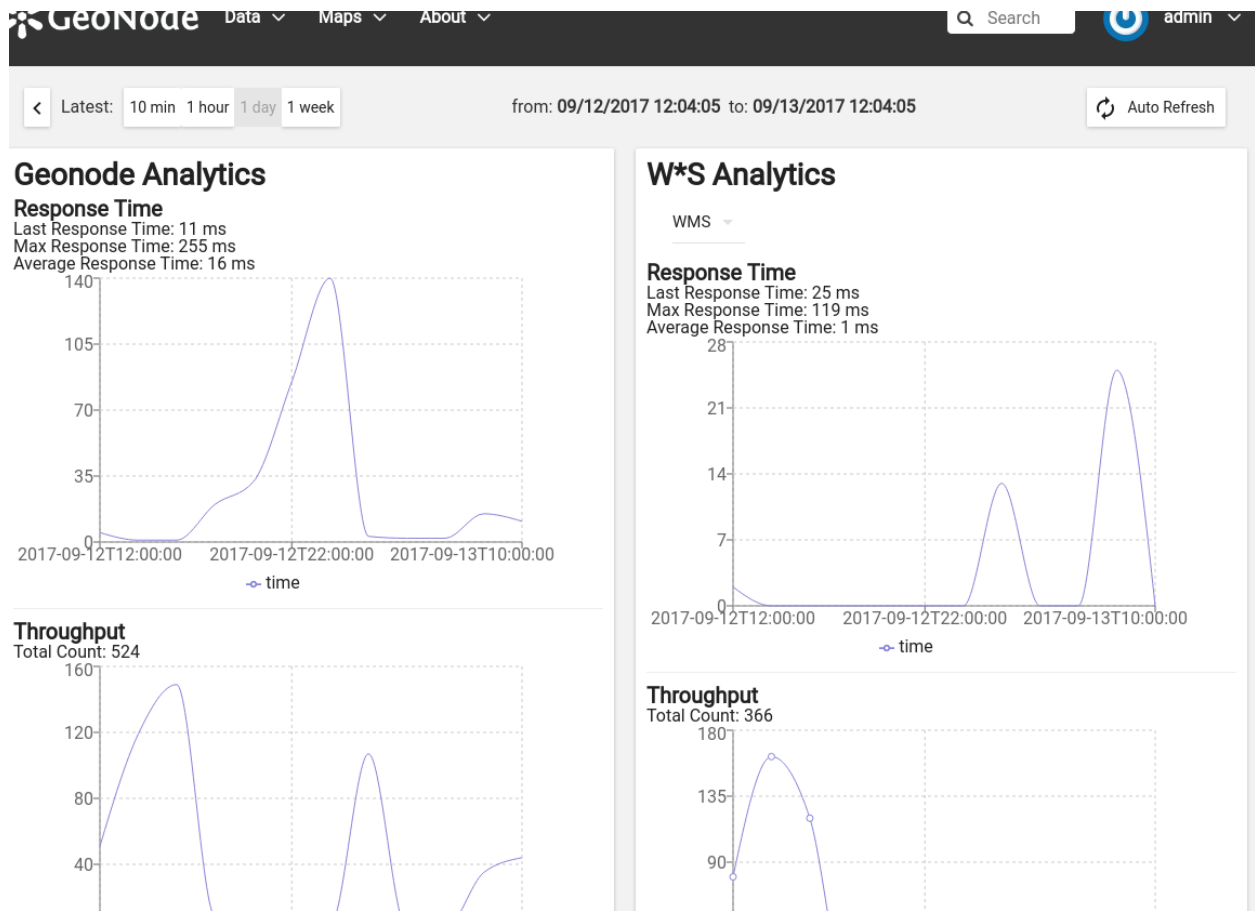
Software Performance view shows GeoServer web service statistics, for all requests monitored and detailed, OWS-specific, per service type (WMS, WFS, OCS etc).



Clicking on



will show charts with data history for overall performance and per-OWS performance:



Hardware Performance

Hardware performance box shows hardware usage statistics for selected host (monitored with any of hostgeonode or hostgeoserver type Service): % of CPU usage and average memory consumption. User can select from which host data will be presented.

017 12:30:16 to: 09/13/2017 12:40:16

 Auto Refresh

Alerts

0 Alerts to show

Errors

0 Errors occurred

Hardware Performance



Host

localhost

GeoNode HW Status

Average CPU
5%

Average Memory
1937 MB

Clicking on

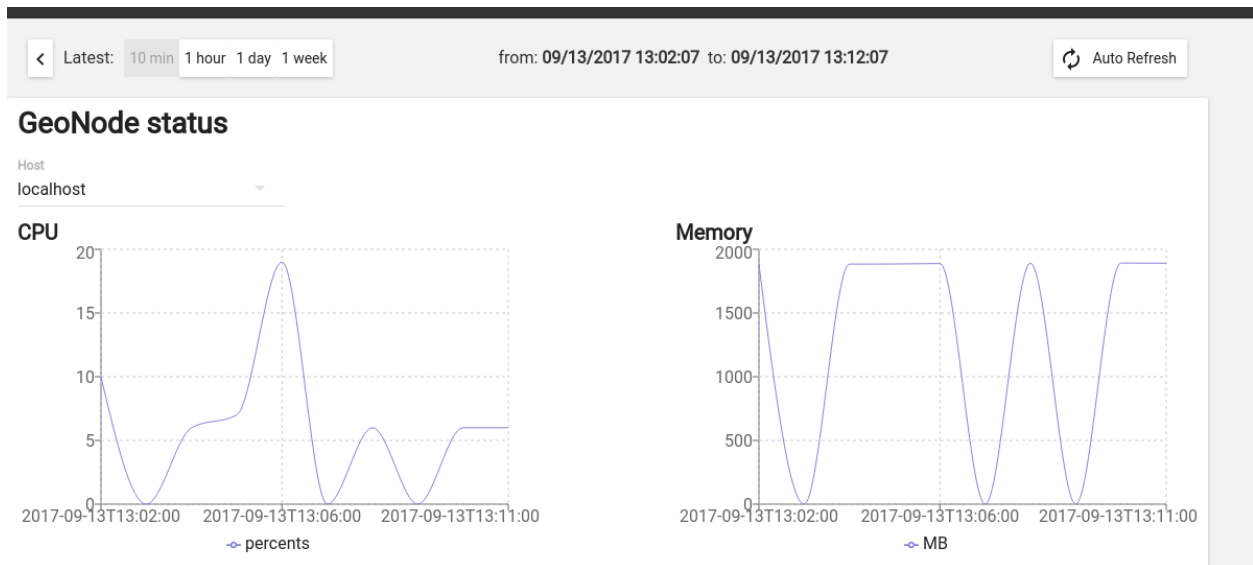
Hardware Performance



Host

localhost

will show charts with data history for selected host and time period



Errors

Errors view will show list of captured errors in GeoNode and GeoServer. List contents is displayed for selected time window.

GeoNode

Latest: 10 min 1 hour 1 day 1 week from: 09/05/2017 16:26:18 to: 09/12/2017 16:26:18 Auto Refresh

Errors

ID	Type	Service	Date
205	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-08T23:58:03.823
206	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-09T06:10:03.556
207	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-10T04:10:03.870
208	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-10T16:20:03.930
209	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-11T08:16:08.702
210	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-11T08:16:08.711
211	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-11T08:16:08.774
212	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-11T10:14:06.617
213	org.geoserver.platform.ServiceExcepti...	demo-geoserver	2017-09-11T10:14:06.729
214	django.template.base.TemplateSyntax...	local-geonode	2017-09-11T11:10:56.110
215	django.template.base.TemplateSyntax...	local-geonode	2017-09-11T11:11:06.116

For each error, details are available:

- error class, message and stack trace
- basic request context (IP, path, user agent)

<

Latest:

10 min

1 hour

1 day

1 week

from: 09/05/2017 16:26:50 to: 09/12/2017 16:26:50

↺

Auto Refresh

Error id: 206

METADATA	CLIENT
Date: 2017-09-09 06:10:03 Service: demo-geoserver Type: org.geoserver.platform.ServiceException Status code: 200 URL: /geosolutions/wms/reflect?format=application/atom+xml&layers=geosolutions:cleaned&featureid=cleaned.187	

Log

```
org.geoserver.wms.map.GetMapKvpRequestReader.parseLayers(GetMapKvpRequestReader.java:1357)
org.geoserver.wms.map.GetMapKvpRequestReader.read(GetMapKvpRequestReader.java:235)
org.geoserver.wms.map.GetMapKvpRequestReader.read(GetMapKvpRequestReader.java:85)
org.geoserver.ows.Dispatcher.parseRequestKVP(Dispatcher.java:1514)
org.geoserver.ows.Dispatcher.dispatch(Dispatcher.java:680)
org.geoserver.ows.Dispatcher.handleRequestInternal(Dispatcher.java:258)
org.springframework.web.servlet.mvc.AbstractController.handleRequest(AbstractController.java:147)
org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter.handle(SimpleControllerHandlerAdapter.java:50)
org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:959)
org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:893)
org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:968)
org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:859)
javax.servlet.http.HttpServlet.service(HttpServlet.java:621)
org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:844)
javax.servlet.http.HttpServlet.service(HttpServlet.java:722)
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:305)
org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:210)
org.geoserver.filters.ThreadLocalsCleanupFilter.doFilter(ThreadLocalsCleanupFilter.java:28)
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:243)
org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:210)
org.geoserver.filters.SpringDelegatingFilter$Chain.doFilter(SpringDelegatingFilter.java:75)
org.geoserver.wms.animate.AnimatorFilter.doFilter(AnimatorFilter.java:71)
```

Alerts

An alert is a descriptive information on situation when observed metric contains values outside allowed range (for example, response time is above 30 seconds, or no requests were served within last 30 minutes). Alerts are generated by notifications mechanism described below.

Alerts view will show list of alerts for current moment (alerts that were generated in past are not displayed here):

GeoNode

Data

Maps

About

Search

admin

<

Latest:

10 min

1 hour

1 day

1 week

from: 09/11/2017 13:56:39 to: 09/12/2017 13:56:39

↺

Auto Refresh

Alerts

Number of handled requests is lower than 800

←

Each alert contains more descriptive information what is wrong:

< Latest: 10 min 1 hour 1 day 1 week from: C

Alerts

Number of handled requests is lower than 800
 Number of requests should be at least 800, got 6 instead

Notifications

Notification mechanism (not to be confused with notifications application in GeoNode) is a way to inform selected users about situations, where collected metric data would indicate a problem with deployment. Notifications are accessible from Alerts view:

Alerts

Number of handled requests is lower than 100
 2017-09-13 15:31:41.142

There can be several notification configurations available.

Alerts Settings

GeoServer is not working
 detects when requests are not handled

Each notification configuration contains two main elements:

- list of email addressess which should be notified when alert is generated
- list of checks (at least one check must be in invalid state to generate alert)

GeoServer is not working

detects when requests are not handled

Who to alert:

When to alert

☒ Number of handled requests is lower than

100 ▾

☒ No response for at least

86400

☒ Response time is higher than

500

s

User can add arbitrary number of emails. Email address doesn't need to point to user registered in GeoNode instance. If email provided doesn't belong to any of users, alert will be send as a regular email. If email provided can be associated with specific user, notifications application (and thus, notification settings for that user) will be used to send alert.

Integration with GeoHealthCheck

GeoNode can also be easily monitored with external tools, like [GeoHealthCheck](#). See [Documentation on adding resources](#) for details.

1.17.2 Monitoring: API

Overview

Geonode monitoring is an optional infrastructure for monitoring resource usage in GeoNode, accompanying GeoServer(s) and hosts on which each service is running. This is not full-fledge monitoring, like zabbix or nagios, rather a moderate size tool to diagnose deployment health. It will be used by users that mostly are not full-time sysops, so usage is simplified.

API

Monitoring API exposes various data to monitoring client.

API root URL is `/monitoring/`, each path in this documentation is relative to that root.

Valid from/valid to

Monitoring collects data periodically, in fixed periods (usually 1 minute). Each metric data is a value (or values if they are split by additional indicators, like resource, label etc) accumulated within that period.

Host

Host is a physical or virtualized instance, on which specific service (GeoNode or GeoServer) is running. This entity is not monitored, but it's used to group services by their deployment location. Hosts list is available in API in `/api/hosts/` endpoint:

GET `/monitoring/api/hosts/`

```
{
  "hosts": [
    {
      "ip": "127.0.0.1",
      "name": "localhost"
    }
  ]
}
```

While host is not monitored directly, some service types (and services of those types) are responsible for monitoring underlying host, hardware resources are monitored indirectly (no dedicated system-level agent is needed).

Service

Service is a name of monitored service. Services are configurable from admin interface, and exposed in API in `/api/services/`:

GET `/monitoring/api/services/`

```
{
  "services": [
    {
      "name": "local-system",
      "last_check": "2017-08-03T13:33:26.674",
      "host": "localhost",
      "check_interval": 60,
      "type": "hostgeonode",
      "id": 3
    },
    {
      "name": "local-geoserver",
      "last_check": "2017-08-03T13:33:26.455",
      "host": "localhost",
      "check_interval": 60,
      "type": "geoserver",
      "id": 2
    },
    {
      "name": "local-geonode",
      "last_check": "2017-08-03T13:33:27.741",
      "host": "localhost",
      "check_interval": 60,
      "type": "geonode",
      "id": 1
    }
  ]
}
```

Each service is described by properties:

- *name* - unique name of service
- *type* - service type name
- *host* - host on which service is running
- *id* - object id
- *last_check* - timestamp with last check (data collection) on that service
- *check_interval* - interval in seconds, how often data should be collected from this service.

Service type

Service type describes kind of services to which it's assigned. There are several service types available:

- *geonode* - service is a GeoNode instance
- *geoserver* - service is a GeoServer instance
- *hostgeonode* - service is not an application, service is underlying host measured with GeoNode (see [Host](#))
- *hostserver* - service is not an application, service is underlying host measured with GeoServer (see [Host](#))

Resource

Resource is an object that can be served by GeoNode or GeoServer. There are several resource types monitored:

- layer
- document
- map
- url

Resource can be served from either GeoNode or GeoServer. We don't check if specific resource actually exists, just keep list of items used and recorded for monitoring. Also, it won't show renames/copies/moves of the same resource.

Resources list is available in `/api/resources/` endpoint:

GET `/monitoring/api/resources/`

```
{
  "resources": [
    {
      "type": "layer",
      "id": 13,
      "name": "unesco:Unesco_point"
    },
    {
      "type": "layer",
      "id": 7,
      "name": "geonode:test"
    },
    {
      "type": "layer",
      "id": 14,
      "name": "http://www.opengis.net/gml:GridCoverage"
    },
    {

```

(continues on next page)

(continuação da página anterior)

```

    "type": "map",
    "id": 17,
    "name": "some map"
  },
]
}

```

Resource is described with following attributes:

- *id* - numeric id of resource record in monitoring
- *type* - type of resource
- *name* - name of resource.

Resources list can be filtered with following query sting arguments:

- *metric_name* - name of metric for which resources should be returned
- *resource_type* - name of type of resource (*layer*, *map*, *document*, *style*, *url*)
- *valid_from* - list resources that are available since that timestamp
- *valid_to* - list resources that are available until that timestamp

Example:

GET /monitoring/api/resources/?resource_type=layer&metric_name=request.
count&valid_from=2017-08-01

```

{
  "resources": [
    {
      "type": "layer",
      "id": 24,
      "name": "atlantis:landmarks"
    },
    {
      "type": "layer",
      "id": 2,
      "name": "topp:states"
    },
    {
      "type": "layer",
      "id": 22,
      "name": "atlantis:island"
    },
    {
      "type": "layer",
      "id": 23,
      "name": "atlantis:poi"
    },
    {
      "type": "layer",
      "id": 16,
      "name": "dissolveroad2"
    },
    {
      "type": "layer",
      "id": 21,

```

(continues on next page)

(continuação da página anterior)

```
    "name": "atlantis:roads"
  }
]
}
```

Resource type

Resource Types describe which types of resource the GeoNode monitoring consider. To retrieve the full list of Resource Types the `/api/resource_types/` is available:

GET `/monitoring/api/resource_types/`

```
{
  "status": "ok",
  "data": {
    "key": "resource_types"
  },
  "errors": {},
  "resource_types": [
    {
      "type": "No resource",
      "name": ""
    },
    {
      "type": "Layer",
      "name": "layer"
    },
    {
      "type": "Map",
      "name": "map"
    },
    {
      "type": "Resource base",
      "name": "resource_base"
    },
    {
      "type": "Document",
      "name": "document"
    },
    {
      "type": "Style",
      "name": "style"
    },
    {
      "type": "Admin",
      "name": "admin"
    },
    {
      "type": "URL",
      "name": "url"
    },
    {
      "type": "Other",
      "name": "other"
    }
  ]
}
```

(continues on next page)

(continuação da página anterior)

```

},
"success": true
}

```

Event Types

Event Types describe the way resources were used in GeoNode. Resource can be accessed as a regular view (through GeoNode, like `/layers/X` url), or through OWS request. Full list of Event Types handled is available in `/api/event_types/` endpoint:

GET `/monitoring/api/event_types/`

```

{
  "status": "ok",
  "errors": { },
  "data": {
    "key": "event_types"
  },
  "event_types": [
    {
      "name": "all"
    },
    {
      "name": "other"
    },
    {
      "name": "download"
    },
    {
      "name": "view"
    },
    {
      "name": "OWS:TMS"
    },
    {
      "name": "OWS:WMS-C"
    },
    {
      "name": "OWS:WMTS"
    },
    {
      "name": "OWS:WCS"
    },
    {
      "name": "OWS:WFS"
    },
    {
      "name": "OWS:WMS"
    },
    {
      "name": "OWS:WPS"
    },
    {
      "name": "OWS:ALL"
    },
  ],
}

```

(continues on next page)

(continuação da página anterior)

```

    {
      "name": "create"
    },
    {
      "name": "upload"
    },
    {
      "name": "change"
    },
    {
      "name": "change_metadata"
    },
    {
      "name": "view_metadata"
    },
    {
      "name": "publish"
    },
    {
      "name": "remove"
    }
  ],
  "success": true
}

```

Event types starting with *OWS:* prefix mean they're related to OWS service. *OWS:ALL* is a cumulative event type, which keeps requests for any OWS.

Event type *other* means request not related to OWS. This is also cumulative event type, and should be used as a baseline of all non-ows requests.

In order to retrieve *OWS* only requests the *ows-service* flag (possible values are *True*, *true*, *False*, *false*, *0*, *1*) can be used:

- *OWS* event types

GET /monitoring/api/event_types/?ows_service=true

```

{
  "status": "ok",
  "errors": {},
  "data": {
    "key": "event_types"
  },
  "event_types": [
    {
      "name": "OWS:TMS",
      "type_label": "TMS"
    },
    {
      "name": "OWS:WMS-C",
      "type_label": "WMS-C"
    },
    {
      "name": "OWS:WMTS",
      "type_label": "WMTS"
    }
  ]
}

```

(continues on next page)

(continuação da página anterior)

```

    "name": "OWS:WCS",
    "type_label": "WCS"
  },
  {
    "name": "OWS:WFS",
    "type_label": "WFS"
  },
  {
    "name": "OWS:WMS",
    "type_label": "WMS"
  },
  {
    "name": "OWS:WPS",
    "type_label": "WPS"
  },
  {
    "name": "OWS:ALL",
    "type_label": "Any OWS"
  }
],
"success": true
}

```

- *non-OWS* event types

GET /monitoring/api/event_types/?ows_service=false

```

{
  "status": "ok",
  "errors": {},
  "data": {
    "key": "event_types"
  },
  "event_types": [
    {
      "name": "other",
      "type_label": "Not OWS"
    },
    {
      "name": "all",
      "type_label": "All"
    },
    {
      "name": "create",
      "type_label": "Create"
    },
    {
      "name": "upload",
      "type_label": "Upload"
    },
    {
      "name": "change",
      "type_label": "Change"
    },
    {
      "name": "change_metadata",
      "type_label": "Change Metadata"
    }
  ]
}

```

(continues on next page)

(continuação da página anterior)

```

    },
    {
      "name": "view_metadata",
      "type_label": "View Metadata"
    },
    {
      "name": "view",
      "type_label": "View"
    },
    {
      "name": "download",
      "type_label": "Download"
    },
    {
      "name": "publish",
      "type_label": "Publish"
    },
    {
      "name": "remove",
      "type_label": "Remove"
    },
    {
      "name": "geoserver",
      "type_label": "Geoserver event"
    }
  ],
  "success": true
}

```

Event type *all* means any request.

Label

Label is a description of subset of metric data that is not described by resources (it's not served as logical data set). Things that can be described with label:

- user tracking id
- volume mount point
- network interface name
- request path
- request method
- response status code
- etc ...

List of all labels recorded is available in `/api/labels/` endpoint:

GET `/monitoring/api/labels/`

```

{
  "labels": [
    {
      "id": 306,

```

(continues on next page)

(continuação da página anterior)

```

    "name": "Other / Other / Python Requests 2.13"
  },
  {
    "id": 315,
    "name": "Kent"
  },
  {
    "id": 298,
    "name": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_5) AppleWebKit/537.36_
↪(KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36"
  },
  {
    "id": 261,
    "name": "lo"
  },
  {
    "id": 331,
    "name": "PUT"
  },
  {
    "id": 334,
    "name": "Other / Other / Python Requests 2.18"
  }
]
}

```

Each metric data set will have at least one label attached. List of labels can be filtered with following query sting arguments:

- *metric_name* - name of metric for which labels should be returned
- *valid_from* - list labels that are available since that timestamp
- *valid_to* - list labels that are available until that timestamp

Example:

GET /monitoring/api/labels/?metric_name=request.ua&valid_from=2017-08-05

```

{
  "labels": [
    {
      "id": 298,
      "name": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_5) AppleWebKit/537.36_
↪(KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36"
    },
    {
      "id": 312,
      "name": "Java/1.8.0_131"
    },
    {
      "id": 293,
      "name": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)_
↪Chrome/36.0.1985.67 Safari/537.36"
    },
    {
      "id": 345,
      "name": "Mozilla/5.0 (Macintosh; Intel Mac OS X) AppleWebKit/538.1 (KHTML, like_
↪Gecko) PhantomJS/2.1.1 Safari/538.1"
    }
  ]
}

```

(continues on next page)

(continuação da página anterior)

```
    },  
    ...  
  ]  
}
```

Metric name

Metric name is a semi-namespace description of what kind of data metric stores. Typical metric names: - *request.count* - *request.ip* - *response.size* - *response.status*

Each service type has a set of metrics available. Application-level services will have different metric set than host-level services.

Full list of metrics is available in `/api/metrics/` endpoint. Returned list is not filterable. Sample response:

GET `/monitoring/api/metrics/`

```
{  
  "metrics": [  
    {  
      "metrics": [  
        {  
          "type": "count",  
          "name": "request.count",  
          "unit": "Count"  
        },  
        {  
          "type": "count",  
          "name": "request.ip",  
          "unit": "Count"  
        },  
        ...  
      ],  
      "service": "geonode"  
    },  
    { "service": "geoserver",  
      "metrics": [..]  
    }  
  ]  
}
```

Metrics are grouped by service. Each metric has following structure:

```
{  
  "type": "count",  
  "name": "request.ip",  
  "unit": "Count"  
}
```

where:

- *type* is a metric data type (it can be count, value or rate). This is internal description of how to deal with aggregation of data for metric.
- *name* name of metric
- *unit* suggested Y-axis label, describing data units

Metric Data

Core feature of monitoring API is ability to get data for given metric for specified period. Metric value is a data set for fixed period of time, from which data were collected and processed for one specific metric name. Additionally, each metric can have data calculated for specific services, resources, labels and event_types. Metric data API has several features:

- it can show metric data within specific time frame, down to 1 minute granularity (may be less if collection intervals are lower).
- it can show metric data aggregated with custom granularity (for example from last 48 hours with 15 minutes granularity).
- it can show metric data for whole monitored setup or for specific resource, label (like user agent type), monitored service (just for geonode or just for geoserver), Event type. Params can be joined in one query.

API endpoint is: `/api/metric_data/METRIC_NAME/`:

Sample request for *request.ua* metric in specific time window (between 10am and 2pm of 2017-08-03) and data granularity (1h)

```
GET /monitoring/api/metric_data/request.ua/?
valid_from=2017-08-03%2010:00:00&valid_to=2017-08-03%2014:00:00&interval=3600
```

```
{
  "data": {
    "input_valid_from": "2017-08-03T10:00:00",
    "input_valid_to": "2017-08-03T14:00:00",
    "data": [
      {
        "valid_from": "2017-08-03T10:00:00",
        "data": [],
        "valid_to": "2017-08-03T11:00:00"
      },
      {
        "valid_from": "2017-08-03T11:00:00",
        "data": [
          {
            "samples_count": 10,
            "val": "10.0000",
            "min": "1.0000",
            "max": "1.0000",
            "sum": "10.0000",
            "label": "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:53.0) Gecko/20100101_↵
↵Firefox/53.0",
            "metric_count": 10
          },
          {
            "samples_count": 790,
            "val": "790.0000",
            "min": "19.0000",
            "max": "79.0000",
            "sum": "790.0000",
            "label": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like_↵
↵Gecko) Chrome/60.0.3112.78 Safari/537.36",
```

(continues on next page)

(continuação da página anterior)

```

        "metric_count": 22
      },
      {
        "samples_count": 150,
        "val": "150.0000",
        "min": "15.0000",
        "max": "15.0000",
        "sum": "150.0000",
        "label": "Mozilla/5.0 (Macintosh; Intel Mac OS X) AppleWebKit/538.1
↪(KHTML, like Gecko) PhantomJS/2.1.1 Safari/538.1",
        "metric_count": 10
      }
    ],
    "valid_to": "2017-08-03T12:00:00"
  },
  {
    "valid_from": "2017-08-03T12:00:00",
    "data": [],
    "valid_to": "2017-08-03T13:00:00"
  },
  {
    "valid_from": "2017-08-03T13:00:00",
    "data": [
      {
        "samples_count": 37,
        "val": "37.0000",
        "min": "4.0000",
        "max": "12.0000",
        "sum": "37.0000",
        "label": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
↪Gecko) Chrome/60.0.3112.40 Safari/537.36",
        "metric_count": 4
      }
    ],
    "valid_to": "2017-08-03T14:00:00"
  }
],
"metric": "request.ua",
"interval": 3600,
"type": "count",
"axis_label": "Count",
"label": null
}
}

```

Metric data response is wrapped with following envelope:

```

"data": {
  "input_valid_from": "2017-08-03T10:00:00",
  "input_valid_to": "2017-08-03T14:00:00",
  "metric": "request.ua",
  "interval": 3600,
  "type": "count",
  "axis_label": "Count",
  "label": null,
  "data": [
    ... # actual data
  ]
}

```

(continues on next page)

(continuação da página anterior)

```

    ],
  }
}

```

where:

- *input_valid_from* and *input_valid_to* are parsed and aligned timestamps for which data are returned,
- *metric* is metric name for which response is returned,
- *interval* data aggregation interval used, in seconds (if none is provided, 60 seconds are used, unless time window is larger than 24 hours),
- *type* is metric data type, which describes internally how data are aggregated (sum, average or min/max function).
- *axis_label* is suggested value-axis label to be used in chart
- *label* is metric data label used (no label by default).

Metric data item is build as following structure:

```

{
  "valid_from": "2017-08-03T13:00:00",
  "valid_to": "2017-08-03T14:00:00",
  "data": [
    {
      "samples_count": 37,
      "val": "37.0000",
      "min": "4.0000",
      "max": "12.0000",
      "sum": "37.0000",
      "label": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like_
↪Gecko) Chrome/60.0.3112.40 Safari/537.36",
      "metric_count": 4
    },
    {
      "samples_count": 20,
      "val": "20.0000",
      "min": "4.0000",
      "max": "10.0000",
      "sum": "20.0000",
      "label": "Internet Explorer 4.0",
      "metric_count": 3
    },
  ],
}

```

where:

- *valid_from* and *valid_to* are timestamps of data aggregation period
- *data* is a list of value rows. When *data* is empty, that means no data were collected for input params.
- each *data* element contains:
 - *label* label value associated with metric data value. This can describe user-provided differentiation value (user agent string, request method etc), or, if such value is not in use, default, «count» or «value» label.
 - *val* is metric data aggregated value, which should be used by frontend application. For *request.ua* this means count of requests for given user agent string, for *response.time* that will return average response

time.

- *min*, *max*, *sum* are helper statistical values to give insight on data used,
- *samples_count* is a sum of all samples counts (actual requests) used for this calculation
- *metric_count* is a number of metric data used to calculate the value.
- *resource* (optional) key with resource structure (*id*, *name*, *type*). This element will be visible when grouping by resource is used.
- *event_type* (optional) key with name of event type related to rest of row. This element will be visible when grouping by event type is used

Metric data can be filtered with following params:

- *valid_from* timestamp (date or date + time) meaning that data should be newer than this timestamp
- *valid_to* timestamp (date or date + time) meaning that data should be older than this timestamp
- *interval* data aggregation interval, in seconds. See below notes about intervals and timestamps alignment
- *label* label value only for which data should be returned (see [Labels](#labels))
- *resource* id of resource (see [Resources](#resources)) for which data should be returned
- *service* name of service (see [Services](#services)) for which data should be returned
- *event_type* name of service (see [Event Types](#ows_service)) for which data should be returned
- *resource_type* name of resource type to filter by, for example *layer* to show only data for layer objects (exclude urls, documents, maps).

grouping metric data

Additionally, in some cases client application may want to receive list of data points in one period for several resources (typical usage scenario: list top-most requested layers). In such case, metric data should be queried also with following params:

- *group_by* - name of object which should be used for grouping. At the moment two grouping modes are available:
 - *resource* - group by resource affected. This will produce metrics for the same label but each resource affected will be listed separately. Returned metric data items will have additional *resource* key, which will hold dictionary with keys *name* and *type*. Sample response:

GET /monitoring/api/metric_data/request.count/?last=86400&interval=86400&group_by=resource

```
{
  "data": {
    "input_valid_from": "2017-09-01T00:00:00",
    "input_valid_to": "2017-09-08T13:50:34.024",
    "data": [
      ..
      {
        "valid_from": "2017-09-04T00:00:00",
        "data": [
          {
            "resource": {
              "type": "layer",
              "name": "nurc:Arc_Sample"
            },
            "samples_count": 300,

```

(continues on next page)

(continuação da página anterior)

```

        "val": "300.0000",
        "min": "100.0000",
        "max": "100.0000",
        "sum": "300.0000",
        "label": "count",
        "metric_count": 3,
        "id": 10
    },
    {
        "resource": {
            "type": "layer",
            "name": "sde:HYP_HR_SR_OB_DR"
        },
        "samples_count": 72,
        "val": "72.0000",
        "min": "24.0000",
        "max": "24.0000",
        "sum": "72.0000",
        "label": "count",
        "metric_count": 3,
        "id": 25
    }
],
"valid_to": "2017-09-05T00:00:00"
}
],
"valid_to": "2017-09-09T00:00:00"
}
],
"metric": "request.count",
"interval": 86400,
"type": "count",
"axis_label": "Count",
"label": null
}
}

```

- *resource_no_labels* - group by resource affected, but do not distinct by label. This will produce similar result as the other grouping, but it will not contain “label” key.

GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=

```

{
  "data": {
    "input_valid_from": "2018-07-10T15:13:50.784Z",
    "input_valid_to": "2018-07-11T15:13:50.784Z",
    "data": [
      {
        "val" id_from: "2018-07-10T15:13:50.784Z",
        "data": [
          {
            "resource": {
              "type": "url",
              "name": "/layers/",
              "id": 15
            },
            "metric_count": 4,

```

(continues on next page)

(continuação da página anterior)

```
        "val": 2,
        "min": "1.0000",
        "max": "1.0000",
        "sum": "4.0000",
        "samples_count": 4
    },
    {
        "resource": {
            "type": "url",
            "name": "/",
            "id": 16
        },
        "metric_count": 4,
        "val": 2,
        "min": "1.0000",
        "max": "4.0000",
        "sum": "7.0000",
        "samples_count": 7
    },
    {
        "resource": {
            "type": "url",
            "name": "/maps/",
            "id": 17
        },
        "metric_count": 4,
        "val": 2,
        "min": "1.0000",
        "max": "2.0000",
        "sum": "5.0000",
        "samples_count": 5
    },
    {
        "resource": {
            "type": "url",
            "name": "/maps/3",
            "id": 18
        },
        "metric_count": 1,
        "val": 1,
        "min": "1.0000",
        "max": "1.0000",
        "sum": "1.0000",
        "samples_count": 1
    },
    {
        "resource": {
            "type": "url",
            "name": "/maps/7",
            "id": 20
        },
        "metric_count": 1,
        "val": 1,
        "min": "1.0000",
        "max": "1.0000",
        "sum": "1.0000",
        "samples_count": 1
    }
```

(continues on next page)

(continuação da página anterior)

```

        }
      ],
      "valid_to": "2018-07-11T15:13:50.784Z"
    }
  ],
  "metric": "request.users",
  "interval": 86400,
  "type": "value",
  "axis_label": "Count",
  "label": null
}

```

- *label* - group by label. This will return number of unique label occurrences within selected period.

GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=label

```

{
  "data": {
    "input_valid_from": "2018-07-10T16:29:08.982Z",
    "input_valid_to": "2018-07-11T16:29:08.982Z",
    "data": [
      {
        "valid_from": "2018-07-10T16:29:08.982Z",
        "data": [
          {
            "samples_count": 243,
            "val": 13,
            "min": "0.0000",
            "max": "25.0000",
            "sum": "243.0000",
            "metric_count": 124
          }
        ],
        "valid_to": "2018-07-11T16:29:08.982Z"
      }
    ],
    "metric": "request.users",
    "interval": 86400.0,
    "type": "value",
    "axis_label": "Count",
    "label": null
  }
}

```

- *event_type* - group by event type. This will expose *event_type* field in data items. Grouping will return number of requests per each event type.
- *event_type_on_label* - group by event type but use label to do grouping (instead of metric data value). This will expose *event_type* field in data items. Grouping will return number of requests per label (especially for *request.users*, which uses label field as tracking id value, see [User Analytics](<https://github.com/geosolutions-it/geonode/wiki/Monitoring:-User-Analytics>)).

Timestamps alignment

Data collected by monitoring are aggregated into fixed period values. This have several consequences:

- you cannot query for time window smaller than aggregation period
- when querying for time window, input `valid_from` and `valid_to` will be aligned to possible actual `valid_from` and `valid_to` values. Alignment is calculated from 0:00h each day. For best results, you should use intervals that can be aligned without reminders.
- timestamps alignment may produce more rows than you expect in some cases. For example, let's say client application want to have data aggregated with 5 minutes interval. Search for data between 12:04 and 12:06, even if interval between those two (2 minutes) is smaller than data interval (5 minutes), this will be aligned to data intervals, which will be:
 - from 12:00 to 12:05
 - from 12:05 to 12:10

If data aggregation period ends in the future, there's good chance it will not contain any data.

Exceptions

Exceptions are served with separate API endpoints. Those endpoints will return:

- list of exceptions captured
- exception details

List of exceptions is available in `/api/exceptions/` endpoint:

GET `/monitoring/api/exceptions/`

```
{
  "exceptions": [
    {
      "url": "/monitoring/api/exceptions/8/",
      "error_type": "exceptions.ValueError",
      "id": 8,
      "service": {
        "type": "geonode",
        "name": "local-geonode"
      },
      "created": "2017-06-20T17:50:24.922"
    },
    {
      "url": "/monitoring/api/exceptions/9/",
      "error_type": "org.geoserver.platform.ServiceException",
      "id": 9,
      "service": {
        "type": "geoserver",
        "name": "local-geoserver"
      },
      "created": "2017-06-26T15:33:20.152"
    },
    {
      "url": "/monitoring/api/exceptions/10/",
      "error_type": "django.db.utils.ProgrammingError",
      "id": 10,
```

(continues on next page)

(continuação da página anterior)

```

    "service": {
        "type": "geonode",
        "name": "local-geonode"
    },
    "created": "2017-06-27T12:32:37.032"
},
]
}

```

Each exception in list contains:

- *error_type* which is a class of exception
- *id* object id for given exception recorded
- *service* service object, on which exception was recorded
- *created* exception recorded timestamp
- *url* url with exception details

Exception details:

GET /monitoring/api/exceptions/30/

```

{
  "error_data": "Traceback (most recent call last):\n File \"/home/cezio/.\n
  ↳virtualenvs/geonode/lib/python2.7/site-packages/django/core/handlers/base.py",\n
  ↳line 132, in get_response\n     response = wrapped_callback(request, *callback_args,\n
  ↳**callback_kwargs)\n File \"/home/cezio/.virtualenvs/geonode/lib/python2.7/site-\n
  ↳packages/django/views/generic/base.py", line 71, in view\n     return self.\n
  ↳dispatch(request, *args, **kwargs)\n File \"/home/cezio/.virtualenvs/geonode/lib/\n
  ↳python2.7/site-packages/django/views/generic/base.py", line 89, in dispatch\n
  ↳return handler(request, *args, **kwargs)\n File \"/mnt/work/cezio/geosolutions/\n
  ↳repos/geonode/geonode/contrib/monitoring/views.py", line 176, in get\n     return\n
  ↳json_response({self.output_name: out})\n File \"/mnt/work/cezio/geosolutions/repos/\n
  ↳geonode/geonode/utils.py", line 619, in json_response\n     body = json.dumps(body,\n
  ↳cls=DjangoJSONEncoder)\n File \"/usr/lib64/python2.7/json/__init__.py", line 251,\n
  ↳in dumps\n     sort_keys=sort_keys, **kw).encode(obj)\n File \"/usr/lib64/python2.7/\n
  ↳json/encoder.py", line 207, in encode\n     chunks = self.iterencode(o, _one_\n
  ↳shot=True)\n File \"/usr/lib64/python2.7/json/encoder.py", line 270, in\n
  ↳iterencode\n     return _iterencode(o, 0)\n File \"/home/cezio/.virtualenvs/geonode/\n
  ↳lib/python2.7/site-packages/django/core/serializers/json.py", line 115, in default\n
  ↳return super(DjangoJSONEncoder, self).default(o)\n File \"/usr/lib64/python2.\n
  ↳7/json/encoder.py", line 184, in default\n     raise TypeError(repr(o) + "\n
  ↳is not\n
  ↳JSON serializable")\nTypeError: <Service: Service: local-geoserver@localhost> is\n
  ↳not JSON serializable\n",
  "service": {
      "type": "geonode",
      "name": "local-geonode"
  },
  "created": "2017-07-24T13:29:28.321",
  "error_type": "exceptions.TypeError",
  "request": {
      "event_type": null,
      "client": {
          "ip": "127.0.0.1",
          "position": {
              "lat": null,

```

(continues on next page)

(continuação da página anterior)

```

    "country": null,
    "lon": null,
    "city": null
  },
  "user_agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like_
↪Gecko) Chrome/60.0.3112.40 Safari/537.36",
  "user_agent_family": "PC / Linux / Chrome 60.0.3112"
},
"request": {
  "path": "/monitoring/api/exceptions/",
  "host": "localhost:8000",
  "method": "GET",
  "created": "2017-07-24T13:29:28.280"
},
"response": {
  "status": 200,
  "time": 30,
  "type": "text/html; charset=utf-8",
  "size": 0
},
"resources": []
},
"error_message": "exceptions.TypeError"
}

```

Details contain:

- *error_type* which is a class of exception
- *error_message* message provided with error
- *error_data* is a plain text with stack trace
- *service* service object, on which exception was recorded
- *created* exception recorded timestamp
- *request* information on request associated with this error:
 - *event_type* name of Event Type associated with request
 - *client* requesting client information
 - *request* details on request received
 - *response* details on response send back
 - *resources* list of resources affected

Autoconfiguration

Autoconfiguration endpoint allows to perform monitoring configuration based on *settings* values. This API endpoint is available to superusers/staff only. Response is wrapped with standard envelope.

POST /monitoring/api/autoconfigure/

```

{
  "status": "ok",
  "success": true,

```

(continues on next page)

(continuação da página anterior)

```
"errors": {}
}
```

1.17.3 Monitoring: User Analytics

Purpose

UA should provide information about GeoNode resources usage at user level (not request level, like plain monitoring).

Requests

1. total number of unique sessions on GeoNode (excluding ows requests) per day. This gives a base view of the reach.

- requests from all sessions of all types, ows and non-ows

```
GET /monitoring/api/metric_data/request.users/?last=(x*86400)&interval=86400&group_
```

- non-ows related

```
GET /monitoring/api/metric_data/request.users/?last=(x*86400)&interval=86400&group_
```

- only ows related

```
GET /monitoring/api/metric_data/request.users/?last=(x*86400)&interval=86400&group_
```

```
{
  "data": {
    "input_valid_from": "2018-07-11T15:41:06.419Z",
    "input_valid_to": "2018-07-12T15:41:06.419Z",
    "data": [
      {
        "valid_from": "2018-07-11T15:41:06.419Z",
        "data": [
          {
            "samples_count": 82,
            "val": 9,
            "min": "0.0000",
            "max": "24.0000",
            "sum": "82.0000",
            "metric_count": 16
          }
        ],
        "valid_to": "2018-07-12T15:41:06.419Z"
      }
    ],
    "metric": "request.users",
    "interval": 86400.0,
    "type": "value",
    "axis_label": "Count",
    "label": null
  }
}
```

2. total number of unique sessions per URL (excluding ows requests). Let me see how many users visits the layers page or the maps page

- get number of unique tracking ids for urls

GET /monitoring/api/metric_data/request.users/?last=(x*86400)&interval=86400&group

```
{
  "data": {
    "input_valid_from": "2018-07-11T15:39:25.126Z",
    "input_valid_to": "2018-07-12T15:39:25.126Z",
    "data": [
      {
        "valid_from": "2018-07-11T15:39:25.126Z",
        "data": [
          {
            "resource": {
              "type": "url",
              "name": "/layers/",
              "id": 15
            },
            "metric_count": 2,
            "val": 2,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "2.0000",
            "samples_count": 2
          },
          {
            "resource": {
              "type": "url",
              "name": "/",
              "id": 16
            },
            "metric_count": 2,
            "val": 2,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "2.0000",
            "samples_count": 2
          },
          {
            "resource": {
              "type": "url",
              "name": "/documents/",
              "id": 21
            },
            "metric_count": 1,
            "val": 1,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "1.0000",
            "samples_count": 1
          }
        ],
        "valid_to": "2018-07-12T15:39:25.126Z"
      }
    ],
    "metric": "request.users",
    "interval": 86400.0,
    "type": "value",
```

(continues on next page)

(continuação da página anterior)

```

    "axis_label": "Count",
    "label": null
  }
}

```

3. total number of unique sessions per event_type: for example total number of unique visits of resource pages (independently by resource type and id)

- to get number of requests

```
GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=
```

- to get number of unique tracking ids

```
GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=
```

- to get number of unique tracking ids for each event_type on a given resource type

```
GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=
```

```

{
  "data": {
    "input_valid_from": "2018-07-11T17:54:41.467Z",
    "input_valid_to": "2018-07-12T17:54:41.467Z",
    "data": [
      {
        "valid_from": "2018-07-11T17:54:41.467Z",
        "data": [
          {
            "samples_count": 5,
            "event_type": "all",
            "val": 2,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "metric_count": 5
          },
          {
            "samples_count": 5,
            "event_type": "other",
            "val": 2,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "metric_count": 5
          },
          {
            "samples_count": 5,
            "event_type": "view",
            "val": 2,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "metric_count": 5
          }
        ],
        "valid_to": "2018-07-12T17:54:41.467Z"
      }
    ]
  }
}

```

(continues on next page)

(continuação da página anterior)

```

    "metric": "request.users",
    "interval": 86400.0,
    "type": "value",
    "axis_label": "Count",
    "label": null
  }
}

```

- to get number of unique users for each event type on specific resource type

GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=

```

{
  "data": {
    "input_valid_from": "2018-07-11T17:54:41.467Z",
    "input_valid_to": "2018-07-12T17:54:41.467Z",
    "data": [
      {
        "valid_from": "2018-07-11T17:54:41.467Z",
        "data": [
          {
            "samples_count": 5,
            "event_type": "all",
            "val": 2,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "metric_count": 5
          },
          {
            "samples_count": 5,
            "event_type": "other",
            "val": 1,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "metric_count": 5
          },
          {
            "samples_count": 5,
            "event_type": "view",
            "val": 1,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "metric_count": 5
          }
        ],
        "valid_to": "2018-07-12T17:54:41.467Z"
      }
    ],
    "metric": "request.users",
    "interval": 86400.0,
    "type": "value",
    "axis_label": "Count",
    "label": null
  }
}

```

(continues on next page)

(continuação da página anterior)

}

4. total number of unique sessions per event_type and single resource: let me see what was the most visited map page in this day, or what was the most downloaded document, what was the most requested ows layer, etc.

- list of most visited resources of *url* type

```
GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=
```

- list of unique tracking ids for each resource (can be narrowed down to specific resource type with *resource_type* values).

```
GET /monitoring/api/metric_data/request.users/?last=86400&interval=86400&group_by=
```

```
{
  "data": {
    "input_valid_from": "2018-07-11T17:56:49.381Z",
    "input_valid_to": "2018-07-12T17:56:49.381Z",
    "data": [
      {
        "valid_from": "2018-07-11T17:56:49.381Z",
        "data": [
          {
            "resource": {
              "type": "",
              "name": "",
              "id": 1
            },
            "metric_count": 16,
            "val": 9,
            "min": "0.0000",
            "max": "24.0000",
            "sum": "82.0000",
            "samples_count": 82
          },
          {
            "resource": {
              "type": "layer",
              "name": "geonode:ne_50m_admin_0_countries_lakes",
              "id": 2
            },
            "metric_count": 4,
            "val": 3,
            "min": "0.0000",
            "max": "2.0000",
            "sum": "3.0000",
            "samples_count": 3
          },
          {
            "resource": {
              "type": "layer",
              "name": "geonode:world_iso2",
              "id": 12
            },
            "metric_count": 4,
            "val": 2,
            "min": "0.0000",
            "max": "5.0000",
```

(continues on next page)

(continuação da página anterior)

```

        "sum": "8.0000",
        "samples_count": 8
    },
    {
        "resource": {
            "type": "url",
            "name": "/layers/",
            "id": 15
        },
        "metric_count": 2,
        "val": 2,
        "min": "1.0000",
        "max": "1.0000",
        "sum": "2.0000",
        "samples_count": 2
    },
    {
        "resource": {
            "type": "url",
            "name": "/",
            "id": 16
        },
        "metric_count": 2,
        "val": 2,
        "min": "1.0000",
        "max": "1.0000",
        "sum": "2.0000",
        "samples_count": 2
    },
    {
        "resource": {
            "type": "url",
            "name": "/documents/",
            "id": 21
        },
        "metric_count": 1,
        "val": 1,
        "min": "1.0000",
        "max": "1.0000",
        "sum": "1.0000",
        "samples_count": 1
    },
    {
        "resource": {
            "type": "document",
            "name": "GeoServer Configuration.pdf",
            "id": 22
        },
        "metric_count": 1,
        "val": 1,
        "min": "5.0000",
        "max": "5.0000",
        "sum": "5.0000",
        "samples_count": 5
    }
],
"valid_to": "2018-07-12T17:56:49.381Z"

```

(continues on next page)

(continuação da página anterior)

```

    }
  ],
  "metric": "request.users",
  "interval": 86400.0,
  "type": "value",
  "axis_label": "Count",
  "label": null
}
}

```

5. total number of unique visitor (user) per event_type and single resource: let me see how many users visited the map page in this day, or how many users download some resource, etc.

- number of unique visitors (users) in a year for a given event_type:

```
GET /monitoring/api/metric_data/request.users/ ?
valid_from=2019-01-01+00:00:00&valid_to=2019-12-31+23:59:59
&interval=31536000&event_type=upload&group_by=user
```

- number of unique visitors (users) in a given time interval and for a given resource_type.

```
GET /monitoring/api/metric_data/request.users/ ?
valid_from=2019-01-01+00:00:00&valid_to=2019-12-31+23:59:59
&interval=31536000&resource_type=layer&group_by=user
```

the responses should look like this:

```

{
  "data": {
    "input_valid_from": "2019-01-01T00:00:00Z",
    "input_valid_to": "2019-12-31T23:59:59Z",
    "data": [
      {
        "valid_from": "2019-01-01T00:00:00Z",
        "data": [
          {
            "samples_count": 3,
            "val": 2,
            "min": "1.0000",
            "max": "2.0000",
            "sum": "3.0000",
            "metric_count": 2
          }
        ],
        "valid_to": "2020-01-01T00:00:00Z"
      }
    ],
    "metric": "request.users",
    "interval": 31536000,
    "type": "value",
    "axis_label": "Count",
    "label": null
  }
}

```

6. total number of unique tracking ids/sessions for a given user.

- sessions count for anonymous users:

```
GET /monitoring/api/metric_data/request.users/?valid_from=2019-01-01+00:00:00&valid_to=2019-12-31T23:59:59Z&interval=31536000&group_by=label&user=AnonymousUser
```

```
{
  "data": {
    "input_valid_from": "2019-01-01T00:00:00Z",
    "input_valid_to": "2019-12-31T23:59:59Z",
    "data": [
      {
        "valid_from": "2019-01-01T00:00:00Z",
        "data": [
          {
            "samples_count": 5,
            "val": 5,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "metric_count": 5
          }
        ],
        "valid_to": "2020-01-01T00:00:00Z"
      }
    ],
    "metric": "request.users",
    "interval": 31536000,
    "type": "value",
    "axis_label": "Count",
    "label": null
  }
}
```

7. total number of unique tracking ids/sessions for each user.

- sessions count for each users:

```
GET /monitoring/api/metric_data/request.users/?valid_from=2019-01-01+00:00:00&valid_to=2019-12-31T23:59:59Z&interval=31536000&group_by=user_on_label
```

```
{
  "data": {
    "input_valid_from": "2019-01-01T00:00:00Z",
    "input_valid_to": "2019-12-31T23:59:59Z",
    "data": [
      {
        "valid_from": "2019-01-01T00:00:00Z",
        "data": [
          {
            "samples_count": 5,
            "val": 5,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "5.0000",
            "user": "AnonymousUser",
            "metric_count": 5
          }
        ],
        {
          "samples_count": 16,
          "val": 3,
          "min": "1.0000",

```

(continues on next page)

(continuação da página anterior)

```

        "max": "2.0000",
        "sum": "16.0000",
        "user": "admin",
        "metric_count": 14
    },
    {
        "samples_count": 4,
        "val": 1,
        "min": "1.0000",
        "max": "1.0000",
        "sum": "4.0000",
        "user": "user1_username",
        "metric_count": 4
    }
],
    "valid_to": "2020-01-01T00:00:00Z"
}
],
"metric": "request.users",
"interval": 31536000,
"type": "value",
"axis_label": "Count",
"label": null
}
}

```

- sessions count for each users which do something with a layer:

GET /monitoring/api/metric_data/request.users/?valid_from=2019-01-01+00:00:00&valid_to=2020-01-01+00:00:00&interval=31536000&resource_type=layer&group_by=user_on_label

```

{
  "data": {
    "input_valid_from": "2019-01-01T00:00:00Z",
    "input_valid_to": "2019-12-31T23:59:59Z",
    "data": [
      {
        "valid_from": "2019-01-01T00:00:00Z",
        "data": [
          {
            "samples_count": 2,
            "val": 1,
            "min": "2.0000",
            "max": "2.0000",
            "sum": "2.0000",
            "user": "admin",
            "metric_count": 1
          },
          {
            "samples_count": 1,
            "val": 1,
            "min": "1.0000",
            "max": "1.0000",
            "sum": "1.0000",
            "user": "user1_username",
            "metric_count": 1
          }
        ]
      }
    ]
  }
}

```

(continues on next page)

(continuação da página anterior)

```

    ],
    "valid_to": "2020-01-01T00:00:00Z"
  }
],
"metric": "request.users",
"interval": 31536000,
"type": "value",
"axis_label": "Count",
"label": null
}
}

```

8. total number of unique users for each monitored resource.

GET /monitoring/api/metric_data/request.users/ ?
 last=31536000&interval=31536000&group_by=resource_on_user

```

{
  "data": {
    "input_valid_from": "2018-09-10T14:15:39.454Z",
    "input_valid_to": "2019-09-10T14:15:39.454Z",
    "data": [
      {
        "valid_from": "2018-09-10T14:15:39.454Z",
        "data": [
          {
            "resource": {
              "href": "/",
              "type": "url",
              "name": "/",
              "id": 1
            },
            "metric_count": 36,
            "val": 4,
            "max": "2.0000",
            "sum": "35.0000",
            "min": "0.0000",
            "samples_count": 35
          },
          {
            "resource": {
              "href": "/maps/",
              "type": "url",
              "name": "/maps/",
              "id": 3
            },
            "metric_count": 3,
            "val": 2,
            "max": "1.0000",
            "sum": "3.0000",
            "min": "1.0000",
            "samples_count": 3
          },
          {
            "resource": {
              "href": "",
              "type": "layer",

```

(continues on next page)

(continuação da página anterior)

```

        "name": "geonode:railways",
        "id": 4
    },
    "metric_count": 5,
    "val": 2,
    "max": "2.0000",
    "sum": "3.0000",
    "min": "0.0000",
    "samples_count": 3
},
{
    "resource": {
        "href": "/layers/",
        "type": "url",
        "name": "/layers/",
        "id": 2
    },
    "metric_count": 4,
    "val": 1,
    "max": "1.0000",
    "sum": "4.0000",
    "min": "1.0000",
    "samples_count": 4
},
{
    "resource": {
        "href": "/documents/2",
        "type": "document",
        "name": "test_doc_1.txt",
        "id": 5
    },
    "metric_count": 2,
    "val": 1,
    "max": "2.0000",
    "sum": "4.0000",
    "min": "2.0000",
    "samples_count": 4
},
{
    "resource": {
        "href": "/maps/3",
        "type": "map",
        "name": "test_map",
        "id": 6
    },
    "metric_count": 1,
    "val": 1,
    "max": "1.0000",
    "sum": "1.0000",
    "min": "1.0000",
    "samples_count": 1
},
{
    "resource": {
        "href": "",
        "type": "layer",
        "name": "geonode:waterways",

```

(continues on next page)

(continuação da página anterior)

```

        "id": 7
      },
      "metric_count": 2,
      "val": 1,
      "max": "2.0000",
      "sum": "2.0000",
      "min": "0.0000",
      "samples_count": 2
    }
  ],
  "valid_to": "2019-09-10T14:15:39.454Z"
}
],
"metric": "request.users",
"interval": 31536000,
"type": "value",
"axis_label": "Count",
"label": null
}
}

```

9. total number of resource monitored in a given time range.

GET /monitoring/api/metric_data/request.users/
last=31536000&interval=31536000&group_by=count_on_resource

?

```

{
  "data": {
    "input_valid_from": "2018-09-10T14:20:27.335Z",
    "input_valid_to": "2019-09-10T14:20:27.335Z",
    "data": [
      {
        "valid_from": "2018-09-10T14:20:27.335Z",
        "data": [
          {
            "samples_count": 52,
            "val": 7,
            "min": "0.0000",
            "max": "2.0000",
            "sum": "52.0000",
            "metric_count": 53
          }
        ],
        "valid_to": "2019-09-10T14:20:27.335Z"
      }
    ],
    "metric": "request.users",
    "interval": 31536000,
    "type": "value",
    "axis_label": "Count",
    "label": null
  }
}

```

1.17.4 Monitoring: Notifications

Notifications are part of monitoring that is run after each data collection cycle. Its configurable mechanism to check if metrics values are within allowed value range, and if not, send notification to designated receivers (registered users or external emails).

Data model

Notification mechanism is composed of several classes, responsible for different aspects:

- High-level configuration: *NotificationCheck*:
 - keeps general description, list of metric check definition, send grace period configuration and last send marker, list of users to which notification should be delivered (in helper table, *NotificationReceiver* class).
- Per-metric definition: *MetricNotificationDefinition*:
 - keeps per-metric-per-check configuration: name of metric, min, max values allowed for user, check type (if value should be below or above given threshold, or should last read be not older than specific period from metric check), additional scope for check (resource, label, ows service - this part is partially implemented). Definition object is created from *NotificationCheck.user_thresholds* data, and is used to generate validation form. Note, that one *NotificationCheck* can have several definition items, for set of different metrics. Definition rows are created when *NotificationCheck* is created, or updated.
- Per-metric check configuration: *MetricNotificationCheck*
 - Keeps per-metric-per-check configuration: metric and threshold values. It is created after user submits configuration form for specific notification.

Workflow

Notifications are checked after each collection/processing period in collection script, by calling *CollectorAPI.emit_notifications(for_timestamp)*. This will do following:

- get all notifications,
- for each notification, will get all notification checks
- for each notification check, it will get metric valid for given timestamp and check if value matches given criteria
- each check can raise exception, which will be captured in caller, and for each notification, list of errors will be returned
- based on list of notifications and errors, alerts will be generated and send to users, unless last delivery was before grace period is finished.

Additionally, notifications expose `/monitoring/api/status/` *Status API*, which will show errors detected at the moment of request.

1.17.5 Web API

Status API

Status endpoint presents current state of error checking performed by notifications. Frontend can make requests periodically to this endpoint. There is no history view for status at the moment. Status response is wrapped with standard response envelope. Non-error response will have *status* key set to *ok* and *success* to *true*, otherwise *errors* will be not empty.

No errors response:

GET /monitoring/api/status/

```
{
  "status": "ok",
  "data": [],
  "success": true
}
```

Response with errors reported:

```
{
  "status": "ok",
  "data": [
    {
      "problems": [
        {
          "threshold_value": "2017-08-29T10:45:26.142",
          "message": "Value collected too far in the past",
          "name": "request.count",
          "severity": "warning",
          "offending_value": "2017-08-25T16:41:00"
        }
      ],
      "check": {
        "grace_period": {
          "seconds": 600,
          "class": "datetime.timedelta"
        },
        "last_send": null,
        "description": "detects when requests are not handled",
        "severity": "warning",
        "user_threshold": {
          "3": {
            "max": 10,
            "metric": "request.count",
            "steps": null,
            "description": "Number of handled requests is lower than",
            "min": 0
          },
          "4": {
            "max": null,
            "metric": "request.count",
            "steps": null,
            "description": "No response for at least",
            "min": 60
          },
          "5": {
```

(continues on next page)

(continuação da página anterior)

```

        "max": null,
        "metric": "response.time",
        "steps": null,
        "description": "Response time is higher than",
        "min": 500
    }
},
    "id": 2,
    "name": "geonode is not working"
}
],
"success": true
}

```

Response with reported errors contains list of check elements in *data* element. Each check element contains:

- *check* - serialized *NotificationCheck* object, which was used
- *problems* - list of metric checks that failed. Each element contains name of metric, severity, error message, measured and threshold value.

Severity

Severity is a textual description of potential impact of error. There are three values: *warning*, *error* and *fatal*.

Notification list

This call will return list of available notifications:

GET /monitoring/api/notifications/

```

{
  "status": "ok",
  "data": {
    "problems": [
      {
        "threshold_value": "10.0000",
        "check_url": "/monitoring/api/notifications/config/2/",
        "name": "request.count",
        "check_id": 2,
        "description": "Metric value for request.count should be at least 10, got 4_
↪instead",
        "offending_value": "4.0000",
        "message": "Number of handled requests is lower than 4",
        "severity": "error"
      }
    ],
    "health_level": "error"
  },
  "success": true
}

```

Response will contain list of notifications summary in *data* key. Each element will have:

- *name* of metric checked

- *message* is error message generated by notification. This describes what the problem is.
- *description* more detailed information what which check failed.
- *offending_value* and *threshold_value* are values that were compared (*offending_value* is actual value from metric data)
- *check_url* to notification details
- *severity* of error

Also, *data* will have highest *severity* value available in *health_level*.

Notification details

This will return details for notification, including form and list of allowed fields:

GET /monitoring/api/notifications/config/{{notification_id}}/

```
{
  "status": "ok",
  "errors": {},
  "data": {
    "fields": [
      {
        "is_enabled": true,
        "use_resource": false,
        "description": "Number of handled requests is lower than",
        "max_value": "10.0000",
        "metric": {
          "class": "geonode.contrib.monitoring.models.Metric",
          "name": "request.count",
          "id": 2
        },
        "min_value": "0.0000",
        "use_label": false,
        "use_ows_service": false,
        "field_option": "min_value",
        "use_service": false,
        "steps_calculated": [
          "0.0000",
          "3.33",
          "6.67",
          "10.0"
        ],
        "current_value": "30.0000",
        "steps": 3,
        "notification_check": {
          "class": "geonode.contrib.monitoring.models.NotificationCheck",
          "name": "geonode is not working",
          "id": 2
        },
        "field_name": "request.count.min_value",
        "id": 3,
        "unit": ""
      },
      {
        "is_enabled": true,
        "use_resource": false,
```

(continues on next page)

(continuação da página anterior)

```

    "description": "No response for at least",
    "max_value": null,
    "metric": {
      "class": "geonode.contrib.monitoring.models.Metric",
      "name": "request.count",
      "id": 2
    },
    "min_value": "60.0000",
    "use_label": false,
    "use_ows_service": false,
    "field_option": "max_timeout",
    "use_service": false,
    "steps_calculated": null,
    "current_value": {
      "seconds": 120,
      "class": "datetime.timedelta"
    },
    "steps": null,
    "notification_check": {
      "class": "geonode.contrib.monitoring.models.NotificationCheck",
      "name": "geonode is not working",
      "id": 2
    },
    "field_name": "request.count.max_timeout",
    "id": 4,
    "unit": ""
  },
  {
    "is_enabled": false,
    "use_resource": false,
    "description": "Response time is higher than",
    "max_value": null,
    "metric": {
      "class": "geonode.contrib.monitoring.models.Metric",
      "name": "response.time",
      "id": 11
    },
    "min_value": "500.0000",
    "use_label": false,
    "use_ows_service": false,
    "field_option": "max_value",
    "use_service": false,
    "steps_calculated": null,
    "current_value": null,
    "steps": null,
    "notification_check": {
      "class": "geonode.contrib.monitoring.models.NotificationCheck",
      "name": "geonode is not working",
      "id": 2
    },
    "field_name": "response.time.max_value",
    "id": 5,
    "unit": "s"
  },
  {
    "is_enabled": false,
    "use_resource": false,

```

(continues on next page)

(continuação da página anterior)

```

"notification": {
  "grace_period": {
    "seconds": 60,
    "class": "datetime.timedelta"
  },
  "last_send": "2017-09-04T13:13:15.203",
  "description": "detects when requests are not handled",
  "severity": "error",
  "user_threshold": {
    "request.count.max_timeout": {
      "max": null,
      "metric": "request.count",
      "steps": null,
      "description": "No response for at least",
      "min": 60
    },
    "response.time.max_value": {
      "max": null,
      "metric": "response.time",
      "steps": null,
      "description": "Response time is higher than",
      "min": 500
    },
    "request.count.min_value": {
      "max": 10,
      "metric": "request.count",
      "steps": 3,
      "description": "Number of handled requests is lower than",
      "min": 0
    }
  },
  "active": true,
  "id": 2,
  "name": "geonode is not working"
},
"success": true
}

```

Returned keys in *data* element:

- *fields* - list of form fields, including detailed per-resource configuration flags
- *form* - rendered user form, which can be displayed
- *notification* - serialized notification object with *user_thresholds* list (this is a base to create *fields* objects)

Frontend should use *fields* list to create whole form in client-side:

- field name is stored in *field_name*.
- field label can be constructed from *description*
- unit can be extracted from *unit* field
- if field definition provides list in *steps_calculated*, this should be used to construct selection/dropdown, otherwise text input should be displayed. If possible, validation should take into account *min_value* and *max_value*.
- currently set value is available in *current_value* field.

- each field has *is_enabled* property, which tells if field is enabled. Currently this value is calculated in following way: field is enabled if *current_value* is not *None*. This may change in the future.

Additionally, each notification configuration accepts list of emails in *emails* field. This field should be send as a list of emails joined with new line char (*n*).

Form should be submitted to the same url as configuration source (`/monitoring/api/notifications/config/{id}/`), see below.

Notification edition (by user)

Following API call allows user to configure notification by setting receivers and adjust threshold values for checks:

POST `/monitoring/api/notifications/config/{notification_check_id}/`

```
request.count.max_value=val
request.count.min_value=1
emails=list of emails
```

Response contains serialized *NotificationCheck* in *data* element, if no errors were captured during form processing:

```
{
  "status": "ok",
  "errors": {},
  "data": {
    "grace_period": {
      "seconds": 600,
      "class": "datetime.timedelta"
    },
    "last_send": null,
    "description": "more test",
    "severity": "error",
    "user_threshold": {
      "request.count.max_value": {
        "max": null,
        "metric": "request.count",
        "steps": null,
        "description": "Max number of request",
        "min": 1000
      },
      "request.count.min_value": {
        "max": 100,
        "metric": "request.count",
        "steps": null,
        "description": "Min number of request",
        "min": 0
      }
    },
    "id": 293,
    "name": "test"
  },
  "success": true
}
```

Error (non-200) response will have *errors* key populated:

```
{
  "status": "error",
```

(continues on next page)

(continuação da página anterior)

```

"errors": {
  "user_threshold": [
    "This field is required."
  ],
  "name": [
    "This field is required."
  ],
  "description": [
    "This field is required."
  ]
},
"data": [],
"success": false
}

```

Notification creation

This API call allows to create new notification, it's different in form layout from edition:

POST /monitoring/api/notifications/

```

name=Name of notification (geonode doesn't work)
description=This will check if geonode is serving any data
emails=
user_thresholds=
severity=

```

Payload elements:

- *name*, *description* are values visible for user
- *severity* severity value
- *emails* is a list of emails, however, it is encoded to a string, where each email is in new line:

```

email1@test.com
email2@test.com

```

- *user_thresholds* is a json encoded list of per-metric-per-check configurations. Each element of list should be a 10-element list, containing:
 - name of metric
 - field check option (one of three values: *min_value*, *max_value* or *max_timeout*)
 - flag, if metric check can use service
 - flag, if metric check can use resource
 - flag, if metric check can use label
 - flag, if metric check can use ows service
 - minimum value for user input (no minimum check if None)
 - maximum value for user input (no maximum check if None)
 - steps count is a number of steps to generate for user input, so user can select value from select list instead of typing. This will have effect only if both min and max values are also provided Sample payload for *user_thresholds*:

```
[
    ('request.count', 'min_value', False, False, False, False, 0, 100, None,
↪ "Min number of request"),
    ('request.count', 'max_value', False, False, False, False, 1000, None,
↪ None, "Max number of request"),
]
```

Response is a serialized *NotificationCheck* wrapped with standard response envelope (status, errors etc). Actual data is in *data* key. If processing failed, for example because of form validation errors, response will be non-200 OK, and *errors* key will be populated.

```
{
  "status": "ok",
  "errors": {},
  "data": {
    "grace_period": {
      "seconds": 600,
      "class": "datetime.timedelta"
    },
    "last_send": null,
    "description": "more test",
    "user_threshold": {
      "request.count.max_value": {
        "max": 100,
        "metric": "request.count",
        "steps": null,
        "description": "Min number of request",
        "min": 0
      },
      "request.count.min_value": {
        "max": null,
        "metric": "request.count",
        "steps": null,
        "description": "Max number of request",
        "min": 1000
      }
    },
    "id": 257,
    "name": "test"
  },
  "success": true
}
```

1.18 GeoNode Backup and Restore

1.18.1 Full GeoNode Backup & Restore

The admin command to backup and restore GeoNode, allows to extract consistently the GeoNode and GeoServer data models in a serializable meta-format which is being interpreted later by the restore procedure in order to exactly rebuild the whole structure.

In particular the tool helps developers and administrators to correctly extract and serialize the following resources:

- **GeoNode** (Resource Base Model):
 1. Layers (both raster and vectors)

2. Maps
 3. Documents
 4. People with Credentials
 5. Permissions
 6. Associated Styles
 7. Static data and templates
- **GeoServer** (Catalog):
 1. OWS Services configuration and limits
 2. Security model along with auth filters configuration, users and credentials
 3. Workspaces
 4. Stores (both DataStores and CoverageStores)
 5. Layers
 6. Styles

The tool exposes two GeoNode Management Commands, “backup” and “restore”.

The commands allow to:

1. Fully backup GeoNode data and fixtures on a zip archive
2. Fully backup GeoServer configuration (physical datasets - tables, shapefiles, geotiffs)
3. Fully restore GeoNode and GeoServer fixtures and catalog from the zip archive

The usage of those commands is quite easy and straightforward.

The first step is to ensure that everything is correctly configured and the requisites respected in order to successfully perform a backup and restore of GeoNode.

Aviso: It is worth to notice that this functionality requires the latest [GeoServer Extension](#) (2.9.x or greater) for GeoNode in order to correctly work.

Nota: GeoServer full documentation is also available here [GeoServer Docs](#)

Requisites and Setup

Before running a GeoNode backup / restore, it is necessary to ensure everything is correctly configured and setup.

Settings

Accordingly to the admin needs, the file `settings.ini` must be created before running a backup or restore.

The default files can be found at `geonode/br/management/commands/settings_sample.ini` and `geonode/br/management/commands/settings_docker_sample.ini` for the classic and Docker environments accordingly. The content is similar in both of them (an example from `settings_sample.ini`):

```
[database]
pgdump = pg_dump
pgrestore = pg_restore

[geoserver]
datadir = geoserver/data
dumpvectordata = yes
dumprasterdata = yes

[fixtures]
# NOTE: Order is important
apps = contenttypes,auth,people,groups,account,guardian,admin,actstream,
↪ announcements,avatar,base,dialogs,documents,geoserver,invitations,pinax_
↪ notifications,layers,maps,oauth2_provider,services,sites,socialaccount,taggit,
↪ tastypie,upload,user_messages
dumps = contenttypes,auth,people,groups,account,guardian,admin,actstream,
↪ announcements,avatar,base,dialogs,documents,geoserver,invitations,pinax_
↪ notifications,layers,maps,oauth2_provider,services,sites,socialaccount,taggit,
↪ tastypie,upload,user_messages
```

The `settings.ini` file can be created in any directory accessible by GeoNode, and its path can be passed to the backup / restore procedures using `-c` (`-config`) argument.

There are few different sections of the configuration file, that must be carefully checked before running a backup / restore command.

Settings: [database] Section

```
[database]
pgdump = pg_dump
pgrestore = pg_restore
```

This section is quite simple. It contains only two properties:

- `pgdump`; the path of the `pg_dump` local command.
- `pgrestore`; the path of the `pg_restore` local command.

Aviso: Those properties are ignored in case GeoNode is not configured to use a DataBase as backend (see `settings.py` and `local_settings.py` sections)

Nota: Database connection settings (both for GeoNode and GeoServer) will be taken from `settings.py` and `local_settings.py` configuration files. Make sure they are correctly configured (on the target GeoNode instance, too) and the DataBase server is accessible while executing a backup / restore command.

Settings: [geoserver] Section

```
[geoserver]
datadir = /opt/gs_data_dir
datadir_exclude_file_path =
dumpvectordata = yes
dumprasterdata = yes
data_dt_filter =
data_layername_filter =
data_layername_exclude_filter =
```

This section allows to enable / disable a full data backup / restore of GeoServer.

- *datadir*: the full path of GeoServer Data Dir, by default `/opt/gs_data_dir`. The path **must** be accessible and **fully writable** by the geonode and / or httpd server users when executing a backup / restore command.
- *datadir_exclude_file_path*: comma separated list of paths to exclude from `geoserver_catalog.zip`; This list will be sent and managed directly by the GeoServer Backup REST API.
- *dumpvectordata*: a boolean flag enabling or disabling creation of a vector data dump from GeoServer (shapefiles or DB tables). If `false` (or `no`) vector data won't be stored / re-stored.
- *dumprasterdata*: a boolean flag enabling or disabling creation of a raster data dump from GeoServer (geotiffs). If `false` (or `no`) raster data won't be stored / re-stored.
- *data_dt_filter*: {cmp_operator} {ISO8601} e.g. `> 2019-04-05T24:00` which means «include on backup archive only the files that have been modified later than 2019-04-05T24:00»
- *data_layername_filter*: comma separated list of layer names, optionally with glob syntax e.g.: `tuscany_*,italy`; Only RASTER original data and VECTORIAL table dumps matching those filters will be **included** into the backup ZIP archive
- *data_layername_exclude_filter*: comma separated list of layer names, optionally with glob syntax e.g.: `tuscany_*,italy`; The RASTER original data and VECTORIAL table dumps matching those filters will be **excluded** from the backup ZIP archive

Aviso: Enabling these options **requires** the GeoServer Data Dir to be accessible and **fully writable** for the geonode and / or httpd server users when executing a backup / restore command.

Settings: [fixtures] Section

```
[fixtures]
#NOTE: Order is important
apps = people,account,avatar.avatar,base.backup,base.license,base.topiccategory,
↳base.region,base.resourcebase,base.contactrole,base.link,base.restrictioncodetype,
↳base.spatialrepresentationtype,guardian.userobjectpermission,guardian.
↳groupobjectpermission,layers.uploadsession,layers.style,layers.layer,layers.
↳attribute,layers.layerfile,maps.map,maps.maplayer,maps.mapsnapshot,documents.
↳document,taggit

dumps = people,accounts,avatars,backups,licenses,topiccategories,regions,
↳resourcebases,contactroles,links,restrictioncodetypes,spatialrepresentationtypes,
↳userpermissions,grouppermissions,uploadsessions,styles,layers,attributes,
↳layerfiles,maps,maplayers,mapsnapshots,documents,tags
```

This section is the most complex one. Usually you don't need to modify it. Only an expert user who knows Python and GeoNode model structure should modify this section.

What its properties mean:

- *apps*; an ordered list of GeoNode Django applications. The backup / restore procedure will dump / restore the fixtures in a portable format.
- *dumps*; this is the list of `files` associated to the Django applications. The order **must** be the same as in the *apps* property above. Each name represents the `file name` where to dump to / read from the single app's fixtures.

Executing from the CLI

The following sections shows instructions on how to perform backup / restore from the command line by using the Django Admin Management Commands.

In order to obtain a basic user guide for the management command from the command line, just run

```
python manage.py backup --help  
python manage.py restore --help
```

`--help` will provide the list of available command line options with a brief description.

By default both procedures activate *Read Only* mode, disabling any content modifying requests, which is reverted to the previous state (from before the execution) after finish, regardless of the command's result (success or failure). To disable activation of this mode, `--skip-read-only` argument can be passed to the command.

It is worth notice that both commands allows the following option

```
python manage.py backup --force / -f  
python manage.py restore --force / -f
```

Which enables a non-interactive mode, meaning the user will not be asked for an explicit confirmation.

Backup

In order to perform a backup just run the command:

```
python manage.py backup --backup-dir=<target_bk_folder_path> --config=</path/  
↪to/settings.ini>
```

The management command will automatically generate a `.zip` archive file on the target folder in case of success. In the target directory `.md5` file with the same name as backup will be created. It contains the MD5 hash of the backup file, which can be used to check archive's integrity before restoration.

It is worth to mention that `br` (Backup & Restore GeoNode application) will not be dumped, even if specified in the `settings.ini` as its content is strictly related to the certain GeoNode instance.

Currently, GeoNode does not support any automatic extraction of the backup file. It should be manually transferred, if needed to the target instance environment.

Restore

The `restore` command has a number of arguments, modifying its execution:

`-c / --config`: path to the `settings.ini` configuration file. If the Backup archive is provided with his settings, the latter will be used by the restore command and this option won't be mandatory anymore

1. `--skip-geoserver`: the GeoServer backup restoration won't be performed
2. `--skip-geoserver-info`: {Default: True} Skips GeoServer Global Infos, like the proxy base url and other global GeoServer metadata info
3. `--skip-geoserver-security`: {Default: True} Skips GeoServer all the Security Settings
4. `--backup-file`: (exclusive together with `--backup-files-dir`) path to the backup `.zip` archive
5. `--backup-files-dir`: (exclusive together with `--backup-file`) directory containing backup archives. The directory may contain a number of files, but **only** backup archives are allowed with a `.zip` extension. In case multiple archives are present in the directory, the newest one, created after the last already restored backup creation time, will be restored. This option was implemented with a thought of automated restores.
6. `--recovery-file`: Backup archive containing GeoNode data to restore in case of failure.
7. `-l / --with-logs`: the backup file will be checked against the restoration logs (history). In case this backup has already been restored (MD5 based comparison), `RuntimeError` is raised, preventing restore execution.
8. `-n / --notify`: the restore procedure outcome will be send by an e-mail notification to the superusers of the instance (note: notification will be sent to the superusers of the instance before restoration).
9. `--skip-read-only`: the restore procedure will be conducted without setting *Read Only* mode during execution.
10. `--soft-reset`: the restore procedure will preserve geoserver table / resources during the restore. By default the procedure will drop tables and resources

In order to perform a default backup restoration just run the command:

```
python manage.py restore --backup-file=<target_restore_file_path> --config=</
↳path/to/settings.ini>
```

For restore to run it requires either `--backup-file` or `--backup-files-dir` argument defined.

Aviso: The Restore will **overwrite** the whole target instances of GeoNode (and by default GeoServer) including users, catalog and database, so be very careful.

GeoNode Admin GUI Inspection

The history of restored backups can be verified in the admin panel.

Login to the admin panel and select `Restored backups` table from `BACKUP/RESTORE` application.

BACKUP/RESTORE

Restored backups

 View

A list will be displayed with a history of all restored backups. You can select a certain backup to view it's data.

Select restored backup to view

NAME	RESTORATION DATE	ARCHIVE MD5	CREATION DATE
2020-03-19_151016.zip	March 24, 2020, 1:33 p.m.	3dfdec2ac8df0ce78e2ef046bed2ef81	March 19, 2020, 3:11 p.m.
2020-03-19_151016.zip	March 24, 2020, 12:11 p.m.	3dfdec2ac8df0ce78e2ef046bed2ef81	March 19, 2020, 3:11 p.m.

2 restored backups

The detailed view of the restored backup shows backup archive's name, it's MD5 hash, it's creation/modification date (in the target folder), and the date of the restoration. Please note Restored Backup history cannot be modified.

View restored backup

HISTORY

Name: 2020-03-19_151016.zip

Restoration date: March 24, 2020, 1:33 p.m.

Archive md5: 3dfdec2ac8df0ce78e2ef046bed2ef81

Creation date: March 19, 2020, 3:11 p.m.

Close

B/R in Docker environment

When executing B/R in the Docker environment, creation backup to / restoration from should be executed in / backup_restore directory. It is a shared volume between Geoserver and Geonode images, created for this purpose only. Pointing at another location will fail, as one of the images won't have an access to the files.

Aviso: When executing B/R in Docker environment **remember** to create `settings.ini` file basing on `settings_docker_sample.ini` to point at a proper Geoserver data directory! In other case configuration mismatch may cause unexpected errors.

Aviso: The only other volume shared between images is `/geoserver_data/data`, but backup creation **should not** be performed there, as the recursive Geoserver backups may be created in such case.

B/R Jenkins Job in Docker environment

When installing GeoNode through the *geonode-project* Docker (see *GeoNode Basic Installation*), an instance of Jenkins CI/CD is also automatically deployed and available through `http://<geonode_host>/jenkins`.

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
/var/jenkins_home/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password



Continue

Configure Jenkins at first startup

The very first time you try to access Jenkins, you will need to unlock it and generate a new administrator username and password.

In order to do that, you need to print the contents of the auto-generated file `/var/jenkins_home/secrets/initialAdminPassword`

1. First of all search for the Jenkins container ID, usually `jenkins4{{project_name}}` where `{{project_name}}` is the name of your *geonode-project* instance (e.g. `my_geonode`)

```
$> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
e9fc97a75d1a	geonode/nginx:geoserver	"/docker-entrypoint...."	2 hours
ago	Up 2 hours	0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp	
c5496400b1b9	my_geonode_django	"/bin/sh -c 'service...'"	2 hours
ago	Up 2 hours		
bc899f81fa28	my_geonode_celery	"/bin/sh -c 'service...'"	2 hours
ago	Up 2 hours		
3b213400d630	geonode/geoserver:2.17.1	"/usr/local/tomcat/t..."	2 hours
ago	Up 2 hours	8080/tcp	
d2f59d70a0d3	geonode/postgis:11	"docker-entrypoint.s..."	2 hours
ago	Up 2 hours	5432/tcp	
3f9ce0be7f88	rabbitmq	"docker-entrypoint.s..."	2 hours
ago	Up 2 hours	4369/tcp, 5671-5672/tcp, 25672/tcp	
02fdbce9ae73	geonode/letsencrypt:latest	". /docker-entrypoint..."	2 hours
ago	Up 14 seconds		
c745520fd551	jenkins/jenkins:lts	"/sbin/tini -- /usr/..."	2 hours
ago	Up 2 hours	0.0.0.0:9080->9080/tcp, 8080/tcp, 0.0.0.0:50000->50000/tcp, 0.0.0.0:9443->8443/tcp	

1. Now just cat the file above inside the Jenkins container

```
$> docker container exec -u 0 -it jenkins4my_geonode sh -c 'cat /var/jenkins_home/
secrets/initialAdminPassword'
```

b91e9d*****373834

1. Copy the hash code you just got from the print above, and copy-and-paste to the browser window

In the next step just install the *Default Plugins*. You can install more of them later on from the management page.

Wait until Jenkins has finished configuring the plugins

Provide the administrator credentials as requested

Confirm the Jenkins instance URL, this can be changed from the configuration later in case you will need to update the server address

Well done, Jenkins is ready now

The next step is to configure a Jenkins Job able to interact with the Django Docker container and run a full backup

Getting Started

Unlock Jenkins

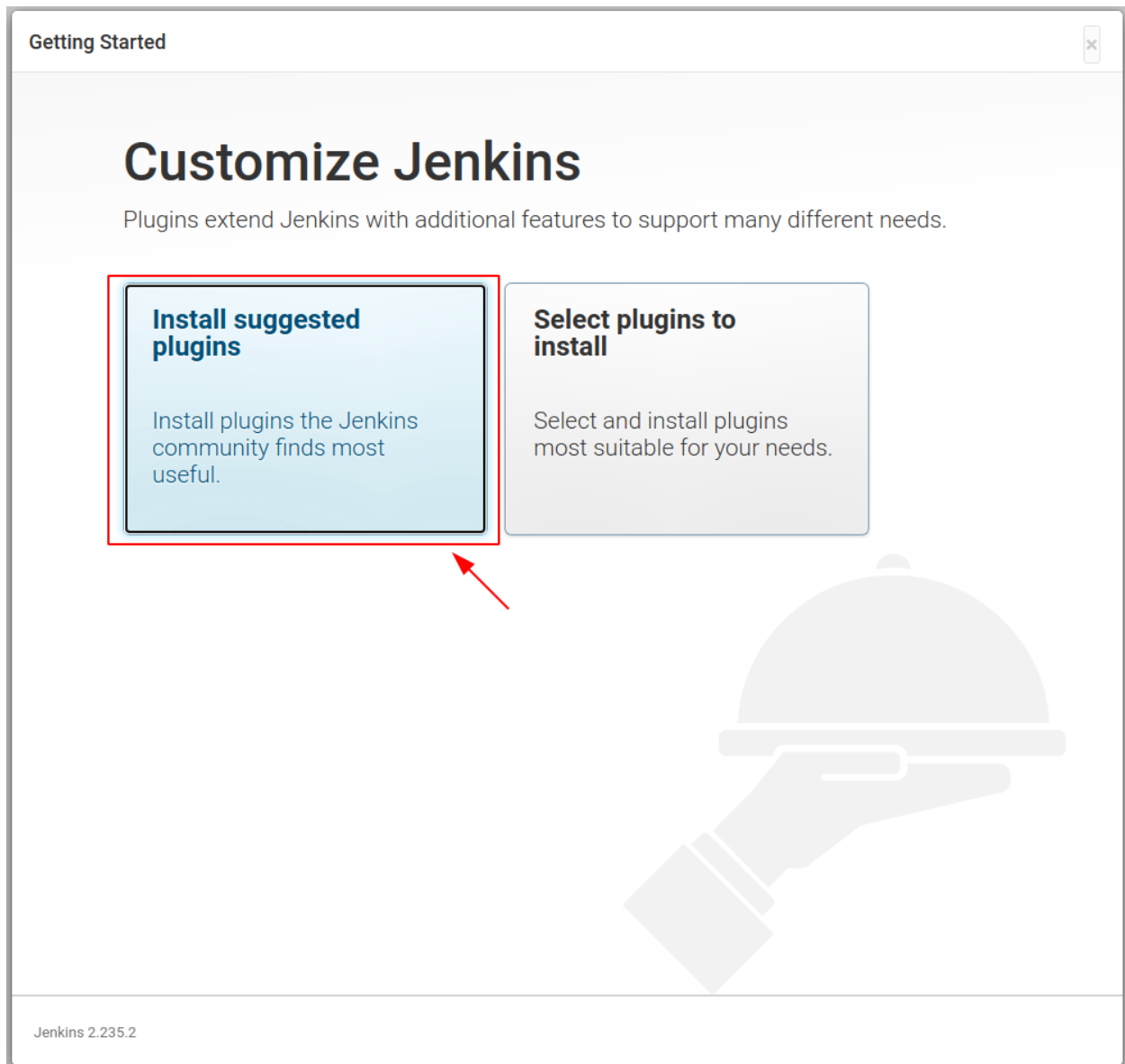
To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/jenkins_home/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password





Getting Started

Getting Started



✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	<pre> ** bouncycastle API ** JavaScript GUI Lib: ACE Editor bundle ** JavaScript GUI Lib: jQuery bundles (jQuery and jQuery UI) ** Pipeline: SCM Step ** Pipeline: Groovy ** Pipeline: Job ** Apache HttpComponents Client 4.x API ** Display URL API Mailer ** Pipeline: Basic Steps Gradle ** Pipeline: Milestone Step ** Snakeyaml API ** Jackson 2 API ** Pipeline: Input Step ** Pipeline: Stage Step ** Pipeline Graph Analysis ** Pipeline: REST API ** JavaScript GUI Lib: Handlebars bundle ** JavaScript GUI Lib: Moment.js bundle Pipeline: Stage View ** Pipeline: Build Step ** Pipeline: Model API ** Pipeline: Declarative Extension Points API ** JSch dependency ** Git client ** GIT server ** Pipeline: Shared Groovy Libraries ** Branch API ** - required dependency </pre>
✓ Timestamper	✓ Workspace Cleanup	✓ Ant	✓ Gradle	
🔄 Pipeline	🔄 GitHub Branch Source	🔄 Pipeline: GitHub Groovy Libraries	✓ Pipeline: Stage View	
🔄 Git	🔄 Subversion	🔄 SSH Build Agents	☐ Matrix Authorization Strategy	
🔄 PAM Authentication	🔄 LDAP	🔄 Email Extension	✓ Mailer	

Jenkins 2.235.2

Getting Started

Create First Admin User

Username:

admin

Password:

.....

Confirm password:

.....

Full name:

Jenkins Admin

E-mail address:

alessio.fabiani@geo-solutions.it

Jenkins 2.235.2

Skip and continue as admin

Save and Continue

Getting Started

Instance Configuration

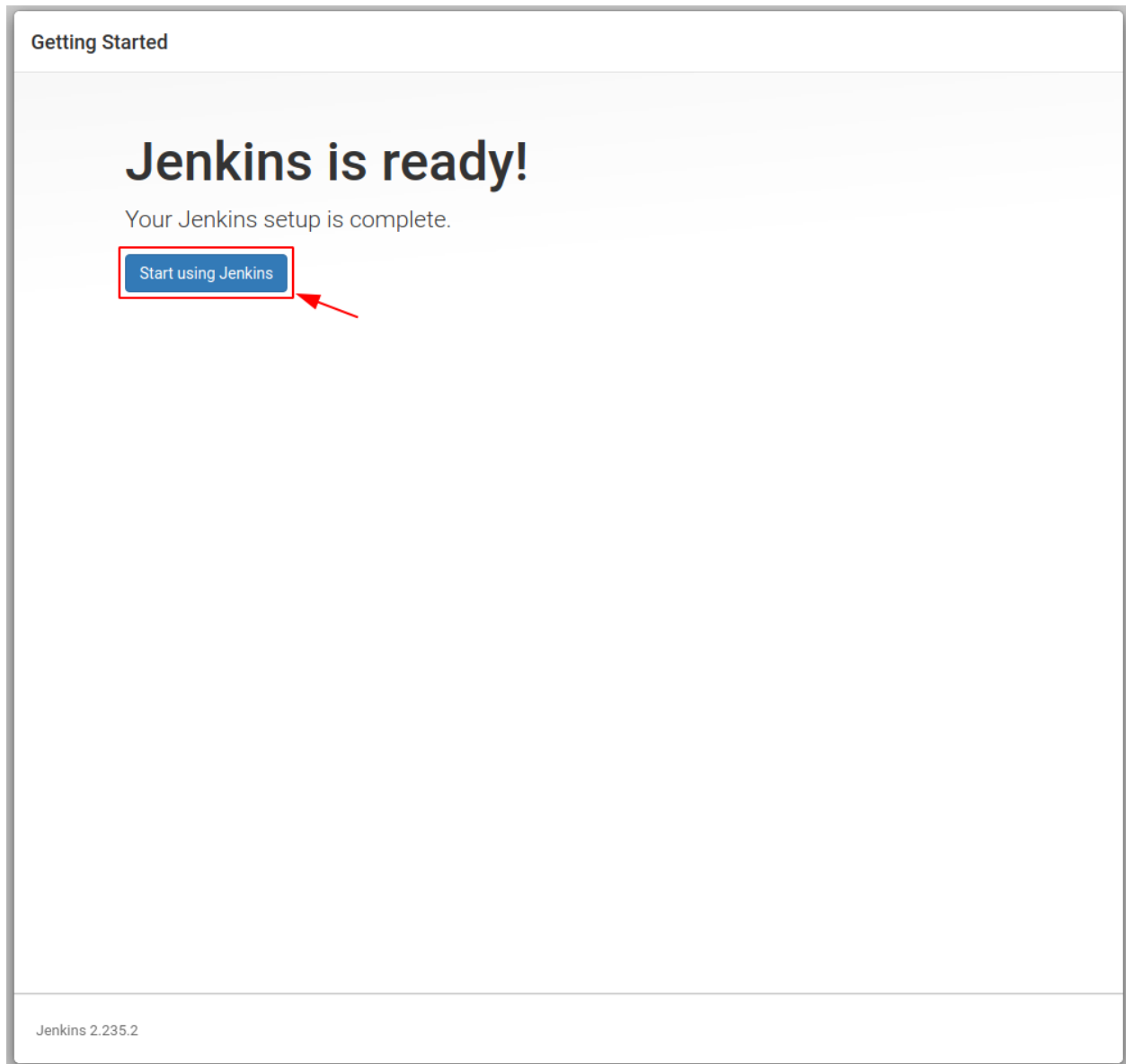
Jenkins URL:

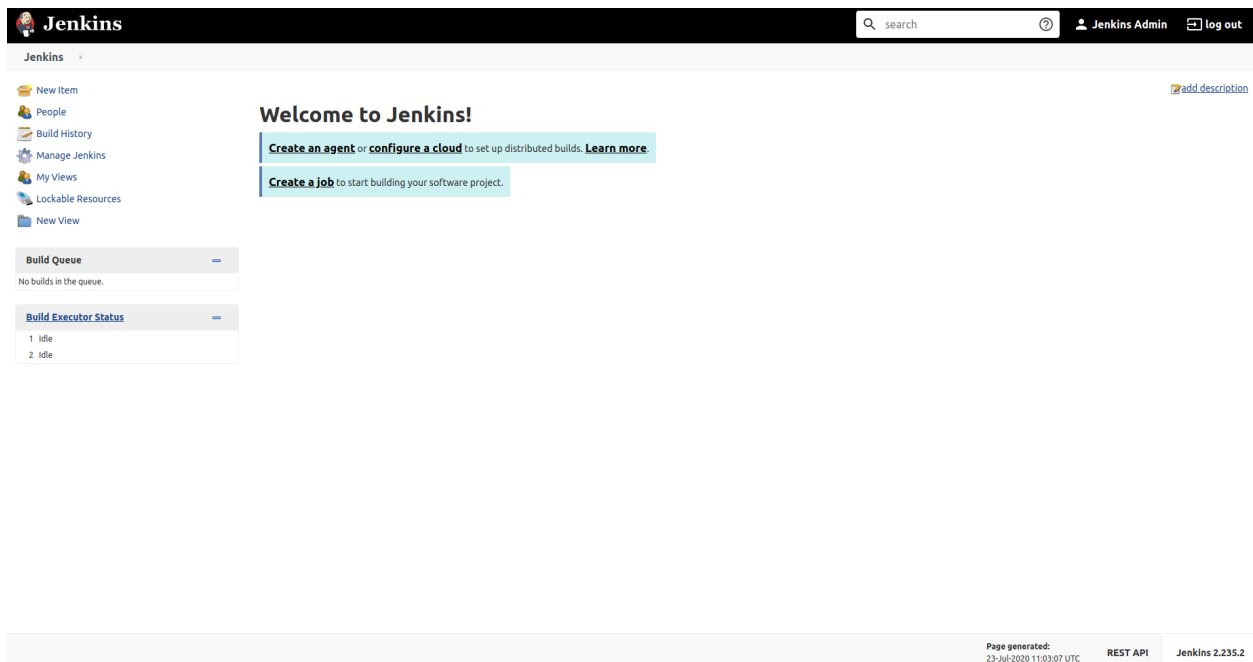
The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.235.2

[Not now](#)





Configure a Jenkins Job to run a full backup on the Django Container

Before creating the new Jenkins job, we need to install and configure a new plugin, [Publish over SSH](#)

In order to do that, once logged in as `admin`, go to the Jenkins *Management Page* > *Manage Plugins* tab

Click on *Available* tab and search for SSH available plugins

Select and check the `Publish over SSH` one

Install the plugins and restart Jenkins

The next step is to configure the SSH `Server Connection` for the *Publish over SSH* plugin.

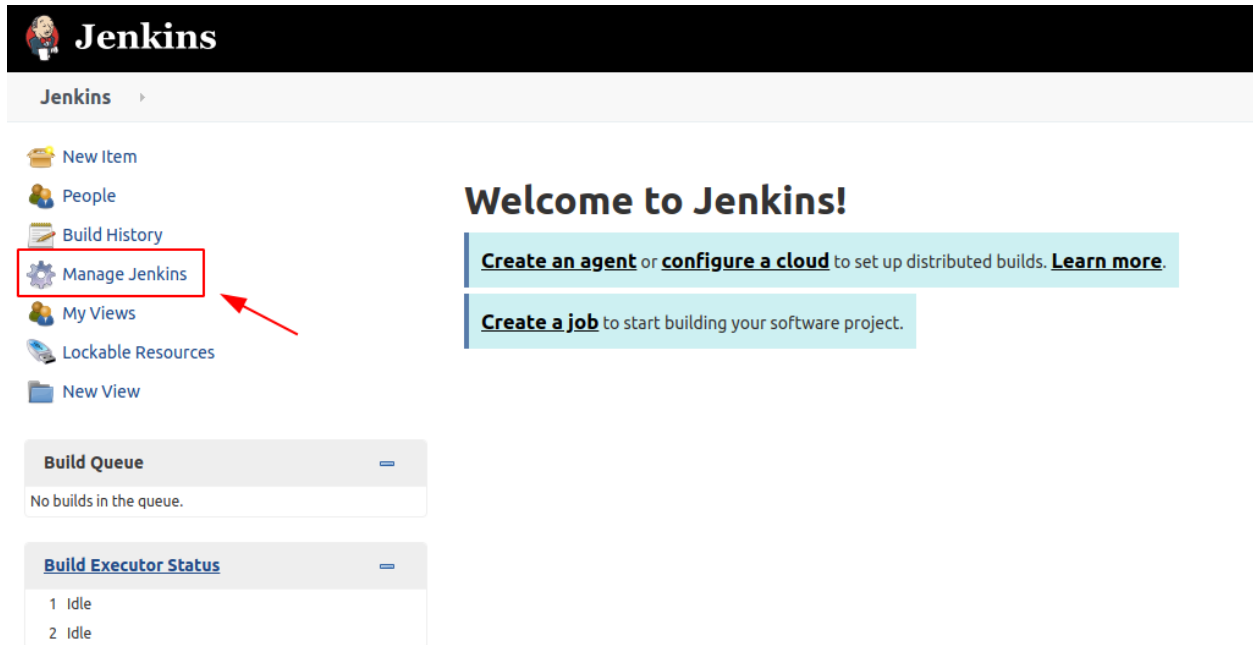
Move to *Jenkins Configuration*

Scroll down until you find the *Publish over SSH* plugin section

Depending on how your `HOST SSH service` has been configured, you might need several information in order to setup the connection.

Here below an example using a global host (`master.demo.geonode.org`) accepting SSH connections via RSA keys

Nota: Before saving the configuration always ensure the connection is ok by using the *Test Configuration* button



The Jenkins Welcome Screen features a dark header with the Jenkins logo and name. Below the header is a navigation sidebar on the left with links: New Item, People, Build History, Manage Jenkins (highlighted with a red box and a red arrow), My Views, Lockable Resources, and New View. The main content area has a large 'Welcome to Jenkins!' heading. Below this are two light blue boxes: the first contains the text 'Create an agent or configure a cloud to set up distributed builds. Learn more.' and the second contains 'Create a job to start building your software project.' At the bottom left, there are two sections: 'Build Queue' showing 'No builds in the queue.' and 'Build Executor Status' showing a list of 1 Idle and 2 Idle executors.

Jenkins

Jenkins >

- New Item
- People
- Build History
- Manage Jenkins**
- My Views
- Lockable Resources
- New View

Welcome to Jenkins!

Create an agent or configure a cloud to set up distributed builds. [Learn more.](#)

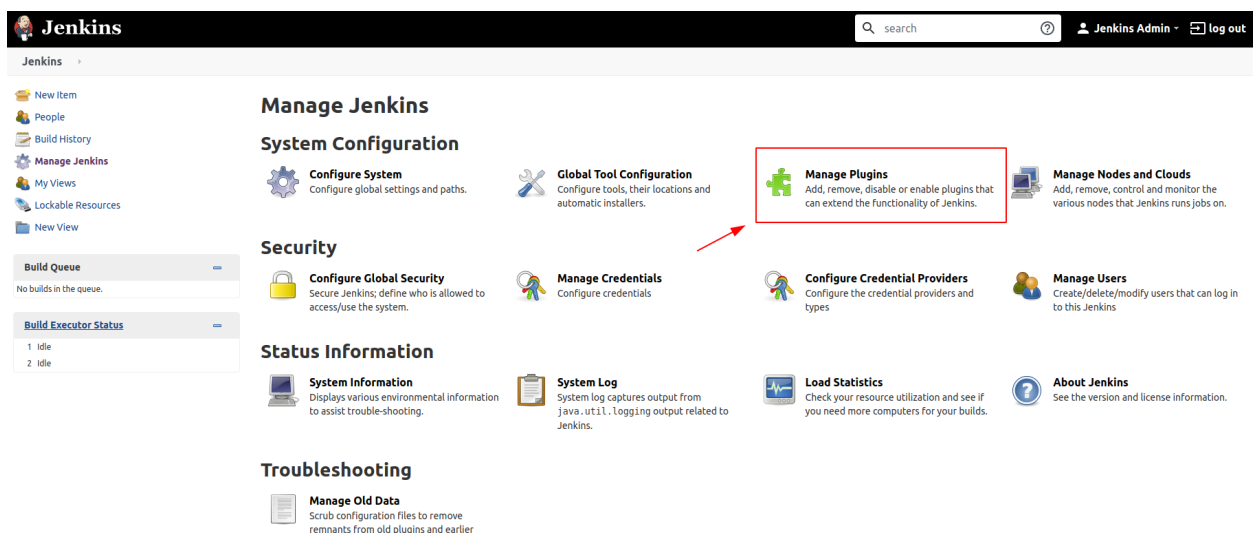
Create a job to start building your software project.

Build Queue

No builds in the queue.

Build Executor Status

- 1 Idle
- 2 Idle



The Jenkins Manage Jenkins page has a dark header with the Jenkins logo, name, a search bar, and user information (Jenkins Admin, log out). The left sidebar is identical to the welcome screen. The main content area is titled 'Manage Jenkins' and is organized into several sections: 'System Configuration' (Configure System, Global Tool Configuration, Manage Plugins - highlighted with a red box and a red arrow, Manage Nodes and Clouds), 'Security' (Configure Global Security, Manage Credentials, Configure Credential Providers, Manage Users), 'Status Information' (System Information, System Log, Load Statistics, About Jenkins), and 'Troubleshooting' (Manage Old Data). Each section contains a brief description of its sub-items.

Jenkins

Jenkins >

- New Item
- People
- Build History
- Manage Jenkins**
- My Views
- Lockable Resources
- New View

Manage Jenkins

System Configuration

- Configure System**
Configure global settings and paths.
- Global Tool Configuration**
Configure tools, their locations and automatic installers.
- Manage Plugins**
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Manage Nodes and Clouds**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security

- Configure Global Security**
Secure Jenkins; define who is allowed to access/use the system.
- Manage Credentials**
Configure credentials
- Configure Credential Providers**
Configure the credential providers and types
- Manage Users**
Create/delete/modify users that can log in to this Jenkins

Status Information

- System Information**
Displays various environmental information to assist trouble-shooting.
- System Log**
System log captures output from java.util.logging output related to Jenkins.
- Load Statistics**
Check your resource utilization and see if you need more computers for your builds.
- About Jenkins**
See the version and license information.

Troubleshooting

- Manage Old Data**
Scrub configuration files to remove remnants from old plugins and earlier

Jenkins > **Plugin Manager**

Back to Dashboard
Manage Jenkins
Update Center

filter

Updates Available Installed Advanced

Install	Name	Version	Released
			No updates available

Update information obtained: 4 hr 53 min ago [Check now](#)

Select: [All](#), [Compatible](#), [None](#)

This page lists updates to the plugins you currently use.

Disabled rows are already upgraded, awaiting restart. Shaded but selectable rows are [in progress or failed](#).

Jenkins > **Plugin Manager**

Back to Dashboard
Manage Jenkins
Update Center

SSH

Updates Available Installed Advanced

Install	Name	Version	Released
☒	Publish Over SSH Build Tools Artifact Uploaders Send build artifacts over SSH This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	1.20.1	1 yr 10 mo ago
☐	SSH Agent This plugin allows you to provide SSH credentials to builds via a ssh-agent in Jenkins	1.20	24 days ago
☐	SSH Build Wrappers This plugin executes shell commands remotely using SSH protocol.	2.6.1	2 yr 3 mo ago
☐	SSH Pipeline Steps pipeline SSH Pipeline Steps	2.0.0	1 yr 0 mo ago
☐	SSH2 Easy This plugin allows you to ssh2 remote server to execute linux commands, shell, sftp upload, download etc	1.4	4 yr 1 mo ago
☐	SCP publisher Artifact Uploaders This plugin uploads build artifacts to repository sites using SCP (SSH) protocol. Warning: This plugin version may not be safe to use. Please review the following security notices: Insecure credential storage and transmission	1.8	9 yr 6 mo ago
☐	Terminate ssh processes	1.0	8 yr 1 mo ago

[Install without restart](#)
[Download now and install after restart](#)
 Update information obtained: 4 hr 54 min ago [Check now](#)

Mailer	 Success
Loading plugin extensions	 Success
Infrastructure plugin for Publish Over X	 Pending
Publish Over SSH	 Pending
Restarting Jenkins	 Pending

[Go back to the top page](#)
(you can start using the installed plugins right away)

 ☒ Restart Jenkins when installation is complete and no jobs are running


Jenkins

Jenkins >

New Item

People

Build History

Manage Jenkins

My Views

Lockable Resources

New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle

Manage Jenkins

System Configuration


Configure System
Configure global settings and paths.


Global Tool Configuration
Configure tools, their locations and automatic installers.


Manage Jenkins
Add, remove and extend plugins.

Security


Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.


Manage Credentials
Configure credentials


Configure System
Configure system types

Status Information


System Information
Displays various environmental information to assist trouble-shooting.


System Log
System log captures output from java.util.logging output related to Jenkins


Load Status
Check the load status of your nodes

Jenkins » configuration

☐ Enable Debug Mode
☐ Require Administrator for Template Testing
☐ Enable watching for jobs
☐ Allow sending to unregistered users

Content Token Reference [Default Triggers...](#)

E-mail Notification
SMTP server
Default user e-mail suffix

☐ Test configuration by sending test e-mail

Publish over SSH
Jenkins SSH Key
Passphrase
Path to key
Key
Disable exec
SSH Servers [Add](#)

[Save](#) [Apply](#)

Page generated: 23-Jul-2020 16:23:26 UTC REST API Jenkins 2.235.2

Publish over SSH

Jenkins SSH Key

Passphrase [Change Password](#)

Path to key `master_priv.key`

Key

Disable exec ☐

SSH Servers

SSH Server

Name `master.demo.geonode.org`

Hostname `master.demo.geonode.org`

Username `geosolutions`

Remote Directory

☐ Use password authentication, or use a different key

Jump host

Port `22`

Timeout (ms) `300000`

Disable exec ☐

Proxy type

Proxy host

[Save](#) [Apply](#)

Proxy host

Proxy port

Proxy user

Proxy password [Change Password](#)

Success

[Test Configuration](#)

[Delete](#)

[Add](#)

It is possible also to run and configure *Jenkins* to run locally, as an instance on *localhost*. In that case you will need to change few things in order to allow *Jenkins* to access your local network.

1. First of all, be sure *OpenSSH Server* is correctly installed and running on your PC. Eventually check any firewall rules.

```
$> sudo apt install openssh-server

# Test your connection locally
$> ssh -p 22 user@localhost
user@localhost's password:
```

2. You will need to do some changes to your `docker-compose.yml` file in order to enable the *host network* configuration.

Nota: Enable `network_mode: "host"` on Jenkins container

```
$> vim docker-compose.yml

...
jenkins:
  image: jenkins/jenkins:lts
  # image: istresearch/jenkins:latest
  container_name: jenkins4${COMPOSE_PROJECT_NAME}
  user: jenkins
  ports:
    - '${JENKINS_HTTP_PORT}:${JENKINS_HTTP_PORT}'
    - '${JENKINS_HTTPS_PORT}:${JENKINS_HTTPS_PORT}'
    - '50000:50000'
  network_mode: "host"
  volumes:
    - jenkins_data:/var/jenkins_home
    - backup-restore:/backup_restore
    # - /var/run/docker.sock:/var/run/docker.sock
  environment:
    - 'JENKINS_OPTS=--httpPort=${JENKINS_HTTP_PORT} --httpsPort=${JENKINS_HTTPS_
    ↪PORT} --prefix=/jenkins'
  ...

# Recreate the Jenkins container
$> docker-compose stop jenkins
$> docker-compose rm jenkins
$> docker-compose up -d jenkins
```

Aviso: From now on, your local Jenkins instance will be accessible from <http://localhost:9080/jenkins>

3. Add localhost Server to the *Publish over SSH* plugin configuration

Mode to <http://localhost:9080/jenkins/configure> and fill the required information

Publish over SSH

Jenkins SSH Key

Passphrase

Path to key

Key

Disable exec ☐

SSH Servers

SSH Server

Name localhost

Hostname localhost

Username afabiani

Remote Directory

☒ Use password authentication, or use a different key

Passphrase / Password

Path to key

Key

Jump host

Port 22

Timeout (ms) 300000

Save Apply

Nota: Before saving the configuration always ensure the connection is ok by using the *Test Configuration* button

Proxy host

Proxy port 0

Proxy user

Proxy password Concealed Change Password

Success

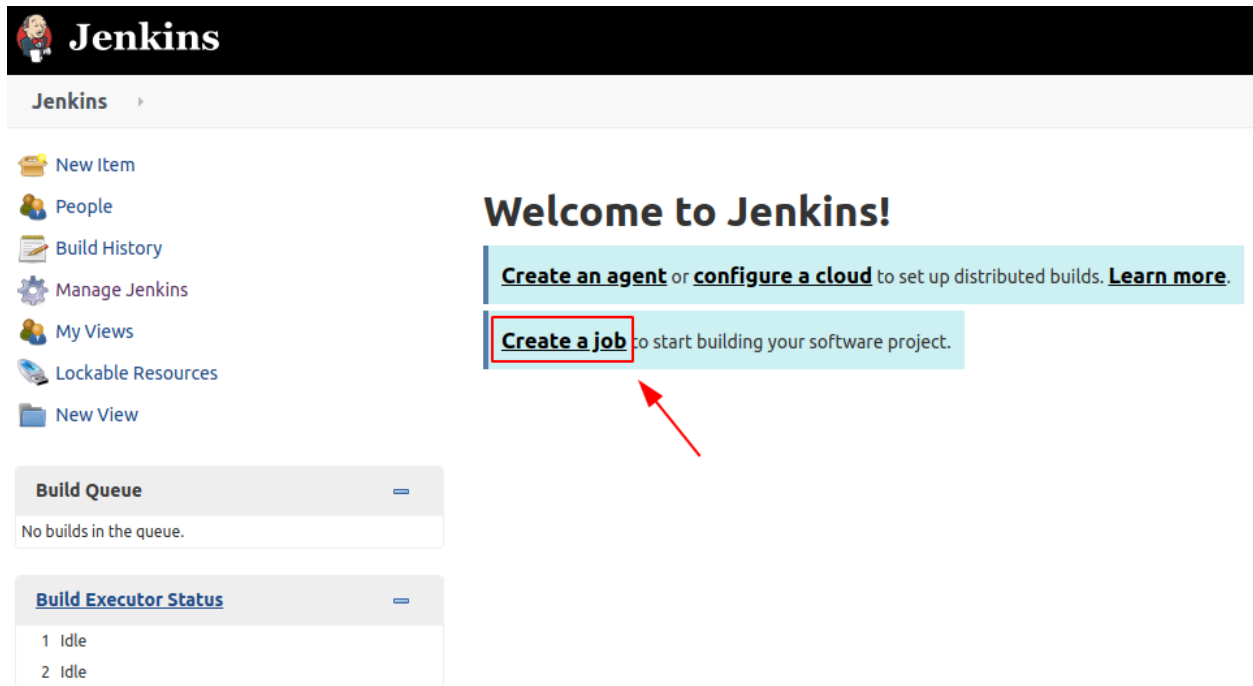
Test Configuration

Delete

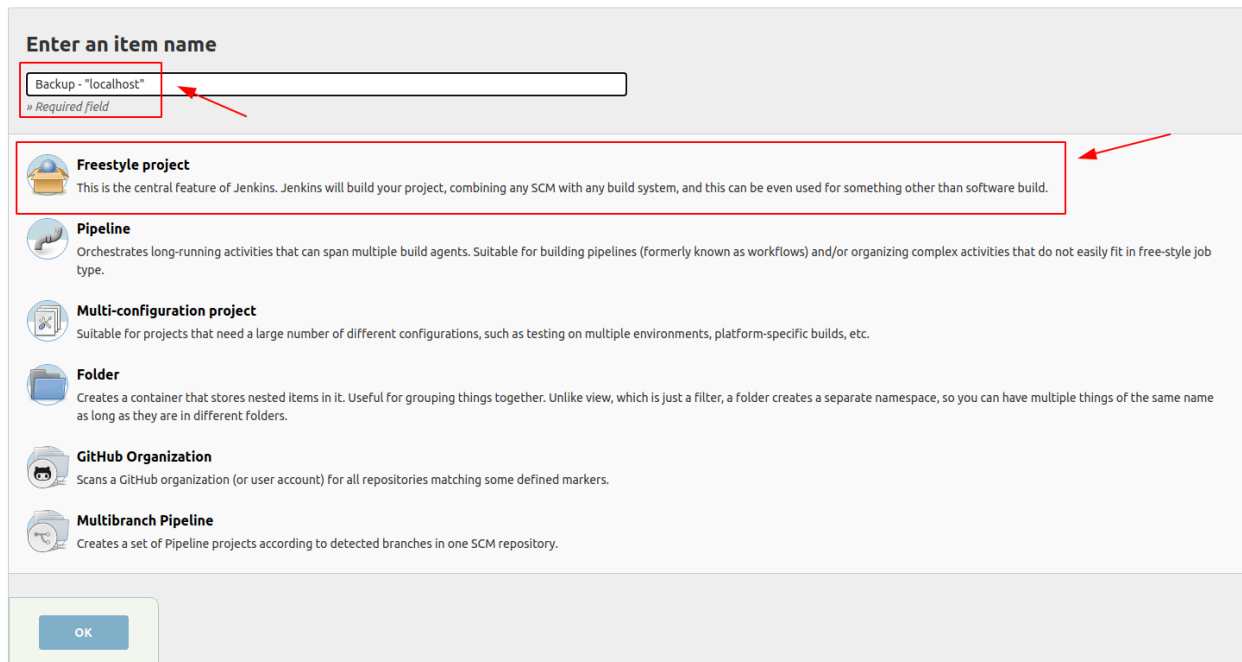
Add

We are now ready to create the Jenkins Job which will run a full backup & restore of our GeoNode dockerized instance.

1. Move to the Jenkins Home and click on *Create a Job* button
2. Provide a name to the Job and select *Freestyle project*



The image shows the Jenkins 'Welcome to Jenkins!' screen. On the left is a sidebar with navigation links: 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Lockable Resources', and 'New View'. The main area has a large heading 'Welcome to Jenkins!' followed by two light blue boxes. The first box contains the text 'Create an agent or configure a cloud to set up distributed builds. [Learn more.](#)'. The second box contains 'Create a job' (highlighted with a red rectangle) followed by 'to start building your software project.' (indicated by a red arrow). Below these are two sections: 'Build Queue' showing 'No builds in the queue.' and 'Build Executor Status' showing a list of 2 'Idle' executors.



The image shows the 'Enter an item name' dialog box in Jenkins. At the top is a text input field containing 'Backup - "localhost"' (highlighted with a red rectangle and a red arrow pointing to it). Below the field is a red asterisk and the text 'Required field'. Below the input field is a list of project types, each with an icon and a description. The first item, 'Freestyle project', is highlighted with a red rectangle and a red arrow. The other items are 'Pipeline', 'Multi-configuration project', 'Folder', 'GitHub Organization', and 'Multibranch Pipeline'. At the bottom left is an 'OK' button.

Enter an item name

Backup - "localhost" * Required field

- Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- GitHub Organization**
Scans a GitHub organization (or user account) for all repositories matching some defined markers.
- Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

3. Enable the *Log rotation* strategy if needed

4. Configure the *Job Parameters* which will be used by the script later on.

Add three *String Parameters*

as shown below

1. *BKP_FOLDER_NAME*
2. *SOURCE_URL*

Aviso: Provide the correct URL of your GeoNode instance

3. *TARGET_URL*

Aviso: Provide the correct URL of your GeoNode instance

String Parameter

Name

BKP_FOLDER_NAME

Default Value

backup_restore

Description

Shared Backup Folder name.
The scripts assume it is located on "root" e.g.: /\$BKP_FOLDER_NAME/

[Plain text] [Preview](#)

☒ Trim the string

String Parameter

Name

SOURCE_URL

Default Value

http://localhost

Description

Source Server URL, the one generating the "backup" file.

[Plain text] [Preview](#)

☒ Trim the string

String Parameter

Name

TARGET_URL

Default Value

http://localhost

Description

Target Server URL, the one which must be synched.

[Plain text] [Preview](#)

☒ Trim the string

5. Enable the *Delete workspace before build starts* and *Add timestamps to the Console Output Build Environment* options

Build Environment

☒ Delete workspace before build starts

Patterns for files to be deleted

Apply pattern also on directories ☐

Check parameter

External Deletion Command

Disable deferred wipeout ☐

☐ Use secret text(s) or file(s)

☐ Send files or execute commands over SSH before the build starts

☐ Send files or execute commands over SSH after the build runs

☐ Abort the build if it's stuck

☒ Add timestamps to the Console Output

☐ Inspect build log for published Gradle build scans

☐ With Ant

6. Finally let's create the *SSH Build Step*

Build

- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- ☒ Send files or execute commands over SSH
- Set build status to "pending" on GitHub commit

Select the correct *SSH Server* and provide the *Exec Command* below

Aviso: Replace `{{project_name}}` with your *geonode-project* instance name (e.g. *my_geonode*)

```
# Replace {{project_name}} with your geonode-project instance name (e.g.
↪ my_geonode)
# docker exec -u 0 -it django4{{project_name}} sh -c 'SOURCE_URL=$SOURCE_
↪ URL TARGET_URL=$TARGET_URL ./{{project_name}}/br/backup.sh $BKP_FOLDER_
↪ NAME'
# e.g.:
docker exec -u 0 -it django4my_geonode sh -c 'SOURCE_URL=$SOURCE_URL_
↪ TARGET_URL=$TARGET_URL ./my_geonode/br/backup.sh $BKP_FOLDER_NAME'
```

Click on *Advanced* and change the parameters as shown below

Send files or execute commands over SSH

SSH Publishers

SSH Server

Name: localhost

Advanced...

Transfers

Transfer Set

Source files: ❌ Either Source files, Exec command or both must be supplied

Remove prefix:

Remote directory:

Exec command: `docker exec -u 0 -it django4{{project_name}} sh -c 'SOURCE_URL=$SOURCE_URL TARGET_URL=$TARGET_URL ./{{project_name}}/br/backup.sh $BKP_FOLDER_NAME'`

All of the transfer fields (except for Exec timeout) support substitution of [Jenkins environment variables](#)

is a date format

Exec timeout (ms): 0

Exec in pty: ☒

Exec using Agent Forwarding: ☐

Add Transfer Set

Save! You are ready to run the Job...

Link the *backup_restore* folder to a local folder on the *HOST*

In the case you need to save the backup archives outside the docker container, there's the possibility to directly link the *backup_restore* folder to a local folder on the *HOST*.

In that case you won't need to *docker cp* the files everytime from the containers, they will be directly available on the host filesystem.

Aviso: Always keep an eye to the disk space. Backups archives may be huge.

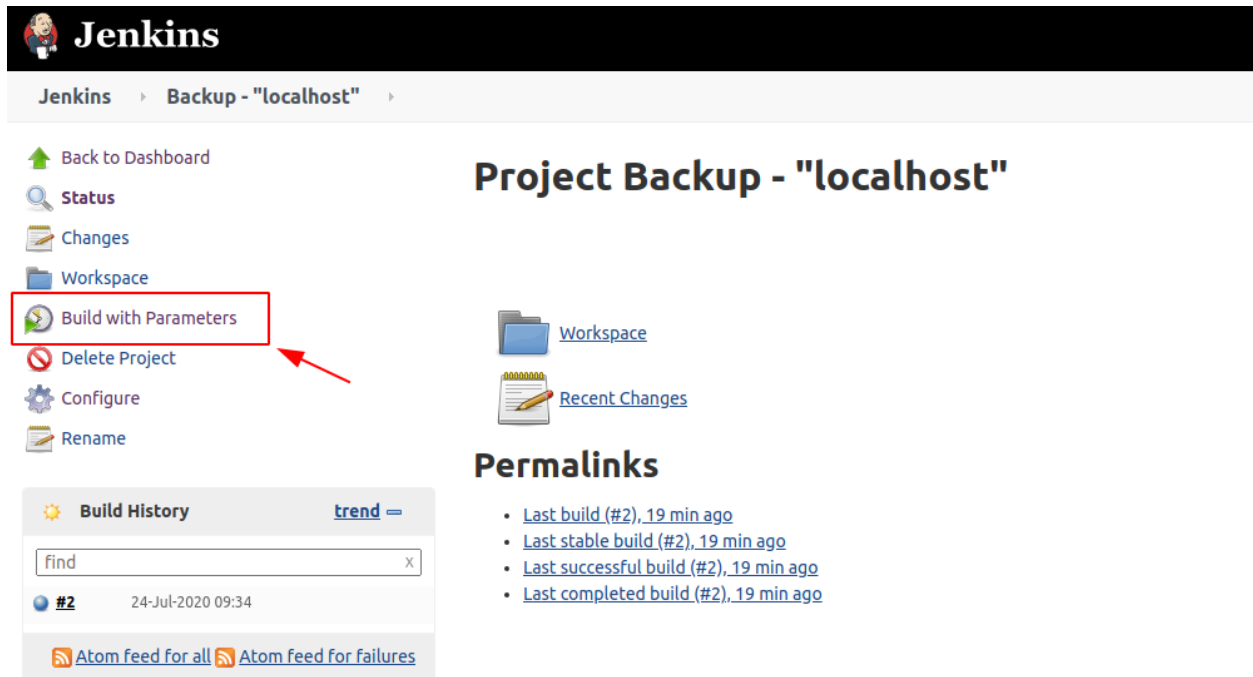
Nota: You might want also to consider filtering the files through the backup dt filters on the *settings.ini* in order to reduce the size of the archive files, including only the new ones.

Modify the `docker-compose.override.yml` as follows in order to link the backup folders outside.

Nota: `/data/backup_restore` is a folder physically located into the host filesystem.

```
$> vim docker-compose.override.yml
```

(continues on next page)



Jenkins Backup - "localhost"

Back to Dashboard
Status
Changes
Workspace
Build with Parameters
Delete Project
Configure
Rename

Project Backup - "localhost"

Workspace
Recent Changes

Permalinks

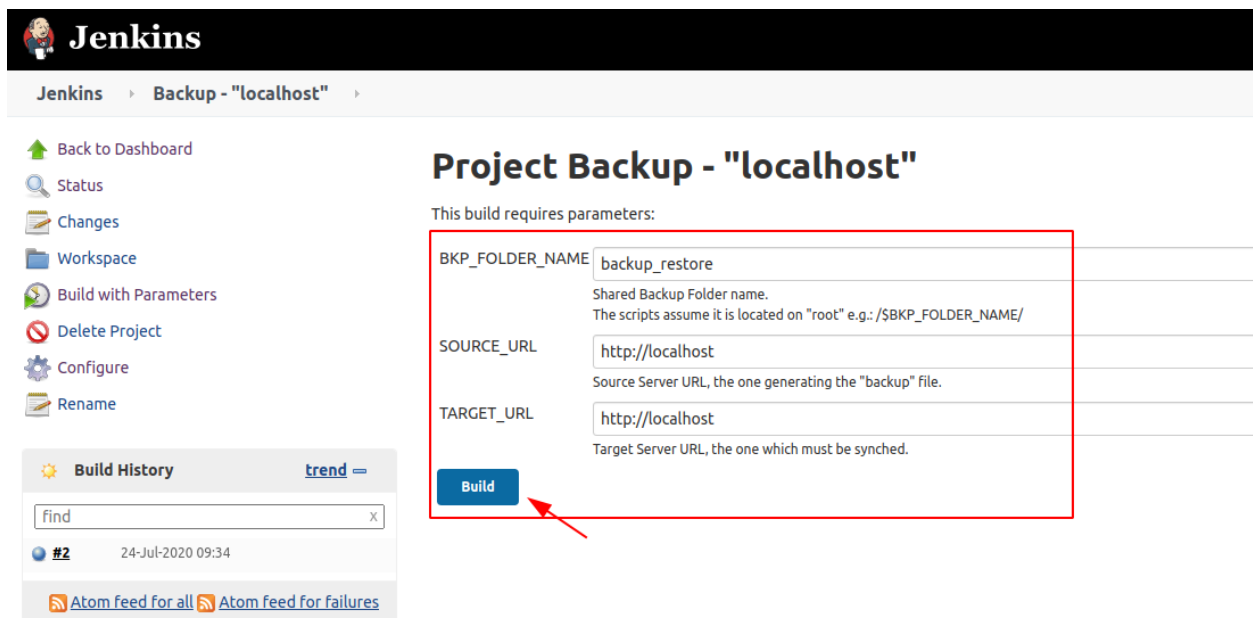
- [Last build \(#2\), 19 min ago](#)
- [Last stable build \(#2\), 19 min ago](#)
- [Last successful build \(#2\), 19 min ago](#)
- [Last completed build \(#2\), 19 min ago](#)

Build History trend

find X

#2 24-Jul-2020 09:34

Atom feed for all Atom feed for failures



Jenkins Backup - "localhost"

Back to Dashboard
Status
Changes
Workspace
Build with Parameters
Delete Project
Configure
Rename

Project Backup - "localhost"

This build requires parameters:

BKP_FOLDER_NAME	backup_restore	Shared Backup Folder name. The scripts assume it is located on "root" e.g.: /\$BKP_FOLDER_NAME/
SOURCE_URL	http://localhost	Source Server URL, the one generating the "backup" file.
TARGET_URL	http://localhost	Target Server URL, the one which must be synched.

Build

Build History trend

find X

#2 24-Jul-2020 09:34

Atom feed for all Atom feed for failures

Jenkins

Jenkins > Backup - "localhost" > #3

[Back to Project](#)
[Status](#)
[Changes](#)

Console Output

[Edit Build Information](#)
[Parameters](#)
[Previous Build](#)

Build #3 (24-Jul-2020 09:54:54)

No changes.
 Started by user [Jenkins Admin](#)

Jenkins

Jenkins > Backup - "localhost" > #3

[Back to Project](#)
[Status](#)
[Changes](#)

Console Output

[View as plain text](#)
[Edit Build Information](#)
[Delete build '#3'](#)
[Parameters](#)
[Previous Build](#)

Time stamps
[View as plain text](#)

☒ System clock time
☒ Use browser timezone
☐ Elapsed time
☐ None

Console Output

```

11:54:54 Started by user Jenkins Admin
11:54:54 Running as SYSTEM
11:54:54 Building in workspace /var/jenkins_home/workspace/Backup - "localhost"
11:54:54 [WS-CLEANUP] Deleting project workspace...
11:54:54 [WS-CLEANUP] Deferred wipeout is used...
11:54:54 [WS-CLEANUP] Done
11:54:54 SSH: Connecting from host [ideapad-afabiani]
11:54:54 SSH: Connecting with configuration [localhost] ...
11:54:55 SSH: EXEC: STDOUT/STDERR from command [docker exec -u 0 -it django4my_geonode sh -c 'SOURCE_URL=http://localhost TARGET_URL=http://localhost ./my_geonode/br/backup.sh backup_restore'] ...
11:54:55 .....
11:54:55 STARTING my_geonode BACKUP Fri Jul 24 09:54:55 UTC 2020
11:54:55 .....
11:54:59 Before proceeding with the Backup, please ensure that:
11:54:59 1. The GeoServer (DB or whatever) is accessible and you have rights
11:54:59 2. The GeoServer is up and running and reachable from this machine
11:54:59 Dumping 'GeoServer Catalog [http://geoserver:8080/geoserver/]' into '/backup_restore/2020-07-24_095459/geoserver_catalog.zip'.
11:54:59 STARTED - 1/11
11:55:02 STARTED - 11/11
11:55:05 COMPLETED - 11/11
11:55:08 Dumping GeoServer Uploaded Data from '/geoserver_data/data/geonode'.
11:55:08 Dumped GeoServer Uploaded Data from '/geoserver_data/data/geonode'.
11:55:08 Dumping GeoServer Uploaded Data from '/geoserver_data/data/data/geonode'.
11:55:08 Skipped GeoServer Uploaded Data '/geoserver_data/data/data/geonode'.
11:55:08 Dumping GeoServer Vectorial Data : my_geonode_data:gf_guser
11:55:08 Dumping GeoServer Vectorial Data : my_geonode_data:gf_gfuser
11:55:09 Dumping GeoServer Vectorial Data : my_geonode_data:gf_gsinstance
11:55:09 Dumping GeoServer Vectorial Data : my_geonode_data:gf_layer_details
11:55:09 Dumping GeoServer Vectorial Data : my_geonode_data:gf_adminrule
11:55:09 Dumping GeoServer Vectorial Data : my_geonode_data:gf_layer_attributes
11:55:09 Dumping GeoServer Vectorial Data : my_geonode_data:gf_rule
11:55:10 Dumping GeoServer Vectorial Data : my_geonode_data:gf_layer_styles
11:55:10 Dumping GeoServer Vectorial Data : my_geonode_data:gf_usergroup
11:55:10 Dumping GeoServer Vectorial Data : my_geonode_data:gf_rule_limits
11:55:10 Dumping GeoServer Vectorial Data : my_geonode_data:gf_user_groups
11:55:10 Dumping geoserver external resources
11:55:10 Dumping 'contenttypes' into 'contenttypes.json'.
11:55:10 Dumping 'auth' into 'auth.json'.
11:55:11 Dumping 'people' into 'people.json'.
11:55:11 Dumping 'groups' into 'groups.json'.
11:55:11 Dumping 'account' into 'account.json'.

```

(continuação da página anterior)

```
version: '2.2'
services:

django:
  build: .
  # Loading the app is defined here to allow for
  # autoreload on changes it is mounted on top of the
  # old copy that docker added when creating the image
  volumes:
    - './usr/src/my_geonode'
    - '/data/backup_restore:/backup_restore' # Link to local volume in the HOST

celery:
  volumes:
    - '/data/backup_restore:/backup_restore' # Link to local volume in the HOST

geoserver:
  volumes:
    - '/data/backup_restore:/backup_restore' # Link to local volume in the HOST

jenkins:
  volumes:
    - '/data/backup_restore:/backup_restore' # Link to local volume in the HOST

# Restart the containers
$> docker-compose up -d
```

1.19 Viewer and Hooksets

1.19.1 GXP

1.19.2 MapStore 2

1.19.3 Leaflet

1.20 GeoNode Components and Architecture

1.20.1 Overview

TODO*

1.20.2 Django

TODO

1.20.3 WebServers

TODO

Apache

TODO

NGINX

TODO

1.20.4 GeoServer

TODO

1.20.5 Databases

TODO

1.20.6 OAuth2 Security: Authentication and Authorization

GeoNode interacts with GeoServer through an advanced security mechanism based on OAuth2 Protocol and GeoFence. This section is a walk through of the configuration and setup of GeoNode and GeoServer Advanced Security.

What we will see in this section is:

- **Introduction**
- **GeoNode (Security Backend):**
 1. Django Authentication
 2. Django OAuth Toolkit Setup and Configuration
 3. Details on `settings.py` Security Settings
- **GeoServer (Security Backend):**
 1. GeoServer Security Subsystem
 2. Introduction to the GeoServer OAuth2 Security Plugin
 3. Configuration of the GeoNode REST Role Service
 4. Configuration of the GeoNode OAuth2 Authentication Filter
 5. The GeoServer Authentication Filter Chains
 6. Introduction to GeoFence Plugin, the Advanced Security Framework for GeoServer
- **Troubleshooting and Advanced Features:**

1. Common Issues and Fixes
2. How to setup HTTPS secured endpoints
3. GeoFence Advanced Features

Introduction

GeoServer, i.e. the geospatial backend server of GeoNode, is a spatial server which needs authenticated users in order to access protected resources or administration functions.

GeoServer supports several kind of Authentication and Authorization mechanisms. Those systems are pluggable and GeoServer can use them at the same time by the use of a `Filter Chain`. Briefly this mechanism allows GeoServer to check for different A&A protocols one by one. The first one matching is used by GeoServer to authorize the users.

GeoNode Authentication is based by default on Django Security Subsystem. Django authentication allows GeoNode to manage its internal users, groups, roles and sessions.

GeoNode has some external components, like GeoServer or QGIS Server, which are pluggable and stand-alone services, devoted to the management of geospatial data. Those external services have their own authentication and authorization mechanisms which must be synchronized somehow with the GeoNode one. Also, those external services maintain, in most of the cases and unless specific configuration does not disable this, alternative security access which for instance allow GeoNode to modify the geospatial catalog under the hood, or a system administrator to have independent and privileged access to the servers.

Before going deeply on how GeoServer/GeoNode A&A works and how it can be configured in order to work correctly with GeoNode, let's quickly clarify the difference between the `Authentication` and `Authorization` concepts.

Authentication

Authentication is the process of verifying the identity of someone through the use of some sort of credentials and a handshake protocol. If the credentials are valid, the authorization process starts. Authentication process always proceeds to Authorization process (although they may often seem to be combined). The two terms are often used synonymously but they are two different processes.

For more details and explanation about the authentication concepts, take a look [here](#).

Authorization

Authorization is the process of allowing authenticated users to access protected resources by checking its roles and rights against some sort of security rules mechanism or protocol. In other words it allows to control access rights by granting or denying specific permissions to specific authorized users.

GeoNode Security Backend

Django Authentication

The Django authentication system handles both authentication and authorization.

The auth system consists of:

1. Users
2. Permissions: Binary (yes/no) flags designating whether a user may perform a certain task.
3. Groups: A generic way of applying labels and permissions to more than one user.

4. A configurable password hashing system
5. Forms and view tools for logging in users, or restricting content
6. A pluggable backend system

The authentication system in Django aims to be very generic and doesn't provide some features commonly found in web authentication systems. Solutions for some of these common problems have been implemented in third-party packages:

1. Password strength checking
2. Throttling of login attempts
3. Authentication against third-parties (OAuth, for example)

Nota: For more details on installation and configuration of Django authentication system, please refer to the official guide <https://docs.djangoproject.com/en/3.2/topics/auth/>.

GeoNode communicates with GeoServer through Basic Authentication under the hood, in order to configure the data and the GeoServer catalog.

In order to do this, you must be sure that GeoNode knows the **internal** admin user and password of GeoServer.

Aviso: This must be an internal GeoServer user with admin rights, not a GeoNode one.

Make sure the credentials are correctly configured into the file `settings.py`

OGC_SERVER

Ensure that the OGC_SERVER settings are correctly configured.

Notice that the two properties `LOGIN_ENDPOINT` and `LOGOUT_ENDPOINT` must specify the GeoServer OAuth2 Endpoints (see details below). The default values `'j_spring_oauth2_geonode_login'` and `'j_spring_oauth2_geonode_logout'` work in most of the cases, unless you need some specific endpoints different from the later. In any case those values **must** be coherent with the GeoServer OAuth2 Plugin configuration.

If in doubt, please use the default values here below.

Default values are:

```
...
# OGC (WMS/WFS/WCS) Server Settings
# OGC (WMS/WFS/WCS) Server Settings
OGC_SERVER = {
    'default': {
        'BACKEND': 'geonode.geoserver',
        'LOCATION': GEOSERVER_LOCATION,
        'LOGIN_ENDPOINT': 'j_spring_oauth2_geonode_login',
        'LOGOUT_ENDPOINT': 'j_spring_oauth2_geonode_logout',
        # PUBLIC_LOCATION needs to be kept like this because in dev mode
        # the proxy won't work and the integration tests will fail
        # the entire block has to be overridden in the local_settings
        'PUBLIC_LOCATION': GEOSERVER_PUBLIC_LOCATION,
        'USER': 'admin',
        'PASSWORD': 'geoserver',
```

(continues on next page)

(continuação da página anterior)

```

    'MAPFISH_PRINT_ENABLED': True,
    'PRINT_NG_ENABLED': True,
    'GEONODE_SECURITY_ENABLED': True,
    'WMST_ENABLED': False,
    'BACKEND_WRITE_ENABLED': True,
    'WPS_ENABLED': False,
    'LOG_FILE': '%s/geoserver/data/logs/geoserver.log' % os.path.abspath(os.path.
→join(PROJECT_ROOT, os.pardir)),
    # Set to name of database in DATABASES dictionary to enable
    'DATASTORE': '', # 'datastore',
    'TIMEOUT': 10 # number of seconds to allow for HTTP requests
}
}
...

```

GeoNode and GeoServer A&A Interaction

The GeoServer instance used by GeoNode, has a particular setup that allows the two frameworks to correctly interact and exchange informations on users credentials and permissions.

In particular GeoServer is configured with a `Filter Chain` for Authorization that makes use of the two following protocols:

1. **Basic Authentication; this is the default GeoServer Authentication mechanism. This makes use of [rfc2617 - Basic and Digest](#)**

In other words, GeoServer takes a `username` and a `password` encoded [Base64](#) on the HTTP Request Headers and compare them against its internal database (which by default is an encrypted XML file on the GeoServer Data Dir). If the user's credentials match, then GeoServer checks for Authorization through its `Role Services` (we will see those services in details on the *GeoServer (Security Backend)* section below).

Nota: GeoServer ships by default with `admin` and `geoserver` as the default administrator user name and password. Before putting the GeoServer on-line it is imperative to change at least the administrator password.

2. **OAuth2 Authentication;** this module allows GeoServer to authenticate against the [OAuth2 Protocol](#). If the Basic Authentication fails, GeoServer falls back to this by using GeoNode as OAuth2 Provider by default.

Nota: Further details can be found directly on the official GeoServer documentation at section «[Authentication Chain](#)»

From the **GeoNode backend (server) side**, the server will make use of **Basic Authentication** with administrator credentials to configure the GeoServer catalog. GeoServer must be reachable by GeoNode of course, and GeoNode must know the internal GeoServer admin credentials.

From the **GeoNode frontend (browser and GUI) side**, the *Authentication* goal is to allow GeoServer to recognize as valid a user which has been already logged into GeoNode, providing kind of an [SSO](#) mechanism between the two applications.

GeoServer must know and must be able to access GeoNode via HTTP/HTTPS. In other words, an external user connected to GeoNode must be authenticated to GeoServer with same permissions. This is possible through the **OAuth2 Authentication** Protocol.

GeoNode / GeoServer Authentication Mechanism

GeoNode as OAuth2 Provider (OP)

OpenID Connect is an identity framework built on OAuth 2.0 protocol which extends the authorization of OAuth 2.0 processes to implement its authentication mechanism. OpenID Connect adds a discovery mechanism allowing users to use an external trusted authority as an identity provider. From another point of view, this can be seen as a single sign on (SSO) system.

OAuth 2.0 is an authorization framework which is capable of providing a way for clients to access a resource with restricted access on behalf of the resource owner. OpenID Connect allows clients to verify the users with an authorization server based authentication.

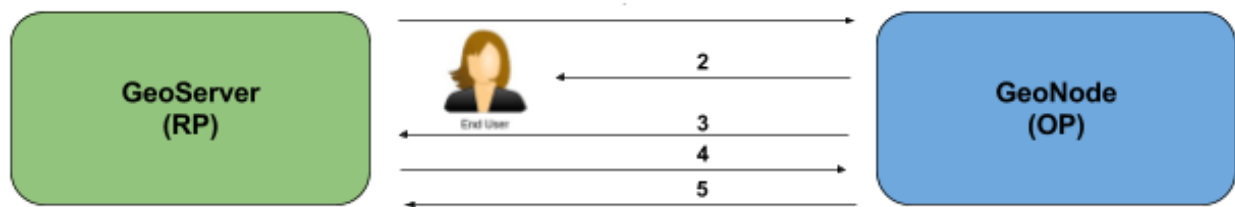
As an OP, GeoNode will be able to act as trusted identity provider, thus allowing the system working on an isolated environment and/or allow GeoNode to authenticate private users managed by the local Django auth subsystem.

GeoServer as OAuth2 Relying Party (RP)

Thanks to the **OAuth2 Authentication** GeoServer is able to retrieve an end user's identity directly from the OAuth2 Provider (OP).

With GeoNode acting as an OP, the mechanism will avoid the use of cookies relying, instead, on the OAuth2 secure protocol.

How the OAuth2 Protocol works:



1. The relying party sends the request to the OAuth2 provider to authenticate the end user
2. The OAuth2 provider authenticates the user
3. The OAuth2 provider sends the ID token and access token to the relying party
4. The relying party sends a request to the user info endpoint with the access token received from OAuth2 provider
5. The user info endpoint returns the claims.

GeoNode / GeoServer Authorization Mechanism

Allowing GeoServer to make use of a OAuth2 in order to act as an OAuth2 RP, is not sufficient to map a user identity to its roles though.

On GeoServer side we will still need to a `RoleService` which would be able to talk to GeoNode and transform the tokens into a User Principal to be used within the GeoServer Security subsystem itself.

In other words after a successful Authentication, GeoServer needs to Authorize the user in order to understand which resources he is enable to access or not. A `REST` based `RoleService` on GeoNode side, allows GeoServer to talk to GeoNode via `REST` to get the current user along with the list of its Roles.

Nevertheless knowing the Roles associated to a user is not sufficient. The complete GeoServer Authorization needs to catch a set of `Access Rules`, associated to the Roles, in order to establish which resources and data are accessible by the user.

The GeoServer Authorization is based on Roles only, therefore for each authenticated user we need also to know:

1. The Roles associated to a valid user session
2. The access permissions associated to a GeoServer Resource

The Authentication mechanism above allows GeoServer to get information about the user and his Roles, which addresses point 1.

About point 2, GeoServer makes use of the [GeoFence Embedded Server](#) plugin. GeoFence is a java web application that provides an advanced authentication / authorization engine for GeoServer using the interface described in [here](#). GeoFence has its own rules database for the management of Authorization rules, and overrides the standard GeoServer security management system by implementing a sophisticated Resource Access Manager. Least but not last, GeoFence implements and exposes a [REST API](#) allowing remote authorized clients to read / write / modify security rules.

The advantages using such plugin are multiple:

1. The Authorizations rules have a fine granularity. The security rules are handled by GeoFence in a way similar to the iptables ones, and allow to define security constraints even on sub-regions and attributes of layers.
2. GeoFence exposes a REST interface to its internal rule database, allowing external managers to update the security constraints programmatically
3. GeoFence implements an internal caching mechanism which improves considerably the performances under load.

GeoNode interaction with GeoFence

GeoNode itself is able to push/manage Authorization rules to GeoServer through the GeoFence [REST API](#), acting as an administrator for GeoServer. GeoNode properly configures the GeoFence rules anytime it is needed, i.e. the permissions of a Resource / Layer are updated.

GeoServer must know and must be able to access GeoNode via HTTP/HTTPS. In other words, an external user connected to GeoNode must be authenticated to GeoServer with same permissions. This is possible through the **GeoNodeCookieProcessingFilter**.

Summarizing we will have different ways to access GeoNode Layers:

1. Through GeoNode via Django Authentication and **GeoNodeCookieProcessingFilter**; basically the users available in GeoNode are also valid for GeoServer or any other backend.

Aviso: If a GeoNode user has «administrator» rights, he will be able to administer GeoServer too.

2. Through GeoServer Security Subsystem; it will be always possible to access to GeoServer using its internal security system and users, unless explicitly disabled (**warning** this is dangerous, you must know what you are doing).

Let's now see in details how the single pieces are configured and how they can be configured.

Django OAuth Toolkit Setup and Configuration

As stated above, GeoNode makes use of the OAuth2 protocol for all the frontend interactions with GeoServer. GeoNode must be configured as an OAuth2 Provider and provide a `Client ID` and a `Client Secret` key to GeoServer. This is possible by enabling and configuring the [Django OAuth Toolkit Plugin](#).

Aviso: GeoNode and GeoServer won't work at all if the following steps are not executed at the first installation.

Default `settings.py` Security Settings for OAuth2

Double check that the OAuth2 Provider and Security Plugin is enabled and that the settings below are correctly configured.

`AUTH_IP_WHITELIST`

`AUTH_IP_WHITELIST` property limits access to users/groups REST Role Service endpoints to the only whitelisted IP addresses. Empty list means “allow all”. If you need to limit “api” REST calls to only some specific IPs fill the list like this: `AUTH_IP_WHITELIST = ['192.168.1.158', '192.168.1.159']`

Default values are:

```
...
AUTH_IP_WHITELIST = []
...
```

`INSTALLED_APPS`

In order to allow GeoNode to act as an OAuth2 Provider, we need to enable the `oauth2_provider` Django application provided by the «Django OAuth Toolkit».

Default values are:

```
...
INSTALLED_APPS = (

    'modeltranslation',

    ...
    'guardian',
    'oauth2_provider',
    ...

) + GEONODE_APPS
...
```

MIDDLEWARE_CLASSES

Installing the `oauth2_provider` Django application is not sufficient to enable the full functionality. We need also GeoNode to include additional entities to its internal model.

Default values are:

```
...
MIDDLEWARE_CLASSES = (
    'django.middleware.common.CommonMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',

    # The setting below makes it possible to serve different languages per
    # user depending on things like headers in HTTP requests.
    'django.middleware.locale.LocaleMiddleware',
    'pagination.middleware.PaginationMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',

    # If you use SessionAuthenticationMiddleware, be sure it appears before
    ↪ OAuth2TokenMiddleware.
    # SessionAuthenticationMiddleware is NOT required for using django-oauth-toolkit.
    'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
    'oauth2_provider.middleware.OAuth2TokenMiddleware',
)
...
```

AUTHENTICATION_BACKENDS

In order to allow GeoNode to act as an OAuth2 Provider, we need to enable the `oauth2_provider.backends.OAuth2Backend` Django backend provided by the «Django OAuth Toolkit». Also notice that we need to specify the OAuth2 Provider scopes and declare which generator to use in order to create OAuth2 Client IDs.

Default values are:

```
...
# Replacement of default authentication backend in order to support
# permissions per object.
AUTHENTICATION_BACKENDS = (
    'oauth2_provider.backends.OAuth2Backend',
    'django.contrib.auth.backends.ModelBackend',
    'guardian.backends.ObjectPermissionBackend',
)

OAUTH2_PROVIDER = {
    'SCOPES': {
        'read': 'Read scope',
        'write': 'Write scope',
        'groups': 'Access to your groups'
    },

    'CLIENT_ID_GENERATOR_CLASS': 'oauth2_provider.generators.ClientIdGenerator',
}
...
```

Django OAuth Toolkit Admin Setup

Once the `settings.py` and `local_settings.py` have been correctly configured for your system:

1. Complete the GeoNode setup steps

- Prepare the model

```
python manage.py makemigrations
python manage.py migrate
python manage.py syncdb
```

- Prepare the static data

```
python manage.py collectstatic
```

- Make sure the database has been populated with initial default data

Aviso: *Deprecated* this command will be replaced by migrations in the future, so be careful.

```
python manage.py loaddata initial_data.json
```

- Make sure there exists a superuser for your environment

Aviso: *Deprecated* this command will be replaced by migrations in the future, so be careful.

```
python manage.py createsuperuser
```

Nota: Read the base tutorials on GeoNode Developer documentation for details on the specific commands and how to use them.

2. Start the application

Start GeoNode accordingly on how the setup has been done; run debug mode through `paver`, or proxied by an HTTP Server like Apache2 HTTPD, Nginx or others.

3. Finalize the setup of the OAuth2 Provider

First of all you need to configure and create a new OAuth2 Application called `GeoServer` through the GeoNode Admin Dashboard

- Access the GeoNode Admin Dashboard
- Go to Django OAuth Toolkit > Applications
- Update or create the Application named `GeoServer`

Menu



Upload Layers



Profile



Recent Activity



Inbox



Announcements



Remote Services



Invite User



GeoServer



Admin



Help

Log out

Django OAuth Toolkit	
Access tokens	+ Add ✎ Change
Applications	+ Add ✎ Change
Grants	+ Add ✎ Change
Refresh tokens	+ Add ✎ Change

Aviso: The Application name **must** be GeoServer

Change application

Client id:	Jrchz2oPY3akmzndmgUTYrs9gcZlgoV2i
User:	2  admin
Redirect uris:	http://localhost:8080/geoserver http://localhost:8080/geoserver/ http://<host_name_or_ip>/geoserver http://<host_name_or_ip>/geoserver/
	Allowed URIs list, space separated
Client type:	Confidential
Authorization grant type:	Authorization code
Client secret:	rCnp5txobUo83EpQEblM8fVj3QT5zb5qI
Name:	GeoServer
☐ Skip authorization	

- **Client id**; An alphanumeric code representing the OAuth2 Client Id. GeoServer OAuth2 Plugin **will** use **this** value.

Aviso: In a production environment it is **highly** recommended to modify the default value provided with GeoNode installation.

- **User**; Search for the admin user. Its ID will be automatically updated into the form.
- **Redirect uris**; It is possible to specify many URIs here. Those must coincide with the GeoServer instances URIs.

- Client type; Choose Confidential
- Authorization grant type; Choose Authorization code
- Client secret; An alphanumeric code representing the OAuth2 Client Secret. GeoServer OAuth2 Plugin **will** use **this** value.

Aviso: In a production environment it is **highly** recommended to modify the default value provided with GeoNode installation.

- Name; **Must** be GeoServer

GeoServer Security Backend

GeoServer Security Subsystem

GeoServer has a robust security subsystem, modeled on Spring Security. Most of the security features are available through the Web administration interface.

For more details on how this works and how to configure and modify it, please refer to the official GeoServer guide <http://docs.geoserver.org/stable/en/user/security/webadmin/index.html>

By using the `GeoServer Data Dir` provided with GeoNode build, the following configuration are already available. You will need just to update them accordingly to your environment (like IP addresses and Host names, OAuth2 Keys, and similar things). However it is recommended to read carefully all the following passages in order to understand exactly how the different component are configured and easily identify any possible issue during the deployment.

The main topics of this section are:

1. Connection to the GeoNode REST Role Service
2. Setup of the GeoServer OAuth2 Authentication Filter
3. Configuration of the GeoServer Filter Chains
4. Setup and test of the GeoFence Server and Default Rules

Connection to the GeoNode REST Role Service

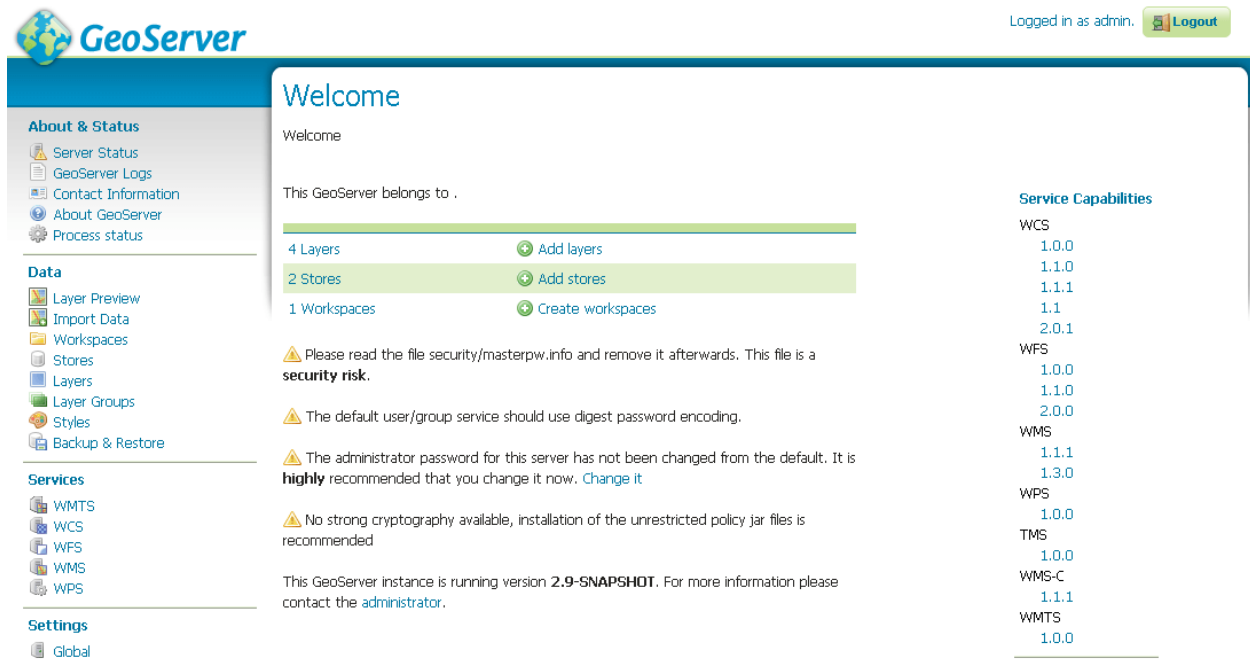
Preliminary checks

- GeoServer is up and running and you have admin rights
- GeoServer must reach the GeoNode instance via HTTP
- The GeoServer Host IP Address must be allowed to access the GeoNode Role Service APIs (see the section `AUTH_IP_WHITELIST` above)

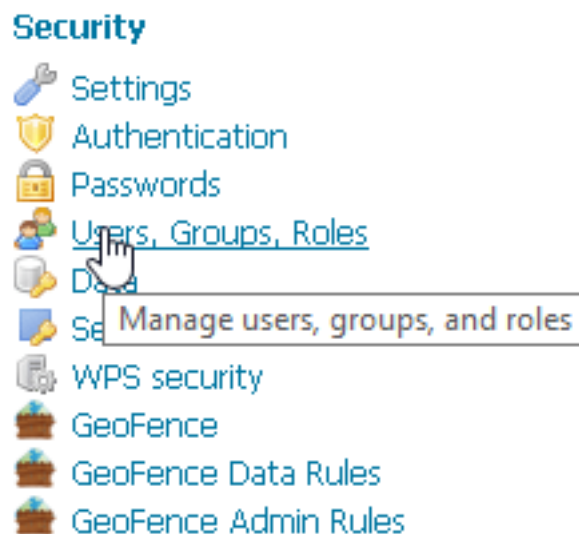
Setup of the GeoNode REST Role Service

1. Login as admin to the GeoServer GUI

Aviso: In a production system remember to change the default admin credentials `admin` `geoserver`



2. Access the Security > Users, Groups, Roles section



3. If not yet configured the service geonode REST role service, click on Role Services > Add

new

Nota: This passage is **not** needed if the `geonode REST role service` has been already created. If so it will be displayed among the Role Services list

Role Services

 Add new

 Remove selected

☐	Name	Type	Administrator Role
☐	default	Default XML role service	ADMIN
☐	geonode REST role service	AuthKEY REST Role Service	ROLE_ADMIN

Results 1 to 2 (out of 2 items)

Users, Groups, and Roles

Manage user group and role services

Services

Users/Groups

Roles

User Group Services

 Add new

 Remove selected

☐	Name	Type
☐	default	Default XML user/group service

Results 1 to 1 (out of 1 ite

Role Services

 Add new

 Remove selected

- If not yet configured the service `geonode REST role service`, choose `AuthKEY REST - Role`

service from REST endpoint

New Role Service

Create and configure a new Role Service

XML - Default role service stored as XML

J2EE - Role service extracting roles from web.xml

AuthKEY REST - Role service from REST endpoint

JDBC - Role service stored in database

LDAP - Role service stored in LDAP repository

5. Create / update the geonode REST role service accordingly

- Name; **Must be** geonode REST role service
- Base Server URL; **Must point to the GeoNode instance base URL** (e.g. `http://<geonode_host_url>`)
- Roles REST Endpoint; **Enter** `/api/roles`
- Admin Role REST Endpoint; **Enter** `/api/adminRole`
- Users REST Endpoint; **Enter** `/api/users`
- Roles JSON Path; **Enter** `$.groups`
- Admin Role JSON Path; **Enter** `$.adminRole`
- Users JSON Path; **Enter** `$.users[0].groups`

Once everything has been setup and it is working, choose the Administrator role and Group administrator role as `ROLE_ADMIN`

Allow GeoFence to validate rules with ROLES

Aviso: The following instruction are different accordingly to the GeoServer version you are currently using.

GeoServer 2.9.x and 2.10.x

1. Access the Security > Settings section
2. Choose the geonode REST role service as Active role service

AuthKEY REST Role Service

Role service from REST endpoint

Settings

Roles

Name

geonode REST role service

Administrator role

ROLE_ADMIN ▼

Group administrator role

ROLE_ADMIN ▼

REST Role Service Settings

Base Server URL

http://<geonode_host_url>

Roles REST Endpoint

/api/roles

Admin Role REST Endpoint

/api/adminRole

Users REST Endpoint

/api/users

Roles JSON Path

\$.groups

Admin Role JSON Path

\$.adminRole

Users JSON Path

\$.users

Security

-  [Settings](#)
-  [Authentication](#)
-  [Passwords](#) Configure global security settings
-  [Users, Groups, Roles](#)
-  [Data](#)

Security Settings

Configure security settings

Active role service

geonode REST role service ▼

default

geonode REST role service

☐ Encrypt web admin URL parameters

Password encryption

Weak PBE ▼

 No strong cryptography available

Save

Cancel

GeoServer 2.12.x and above

With the latest updates to GeoFence Plugin, the latter no more recognizes the Role Service from the default settings but from the `geofence-server.properties` file.

That said, it is important that the Security > Settings role service will be set to **default**, in order to allow GeoServer following the standard authorization chain.

On the other side, you will need to be sure that the `geofence-server.properties` file under the `$GEOSERVER_DATA_DIR/geofence` folder, contains the two following additional properties:

```
gwc.context.suffix=gwc
org.geoserver.rest.DefaultUserGroupName=geonode REST role service
```

Setup of the GeoServer OAuth2 Authentication Filter

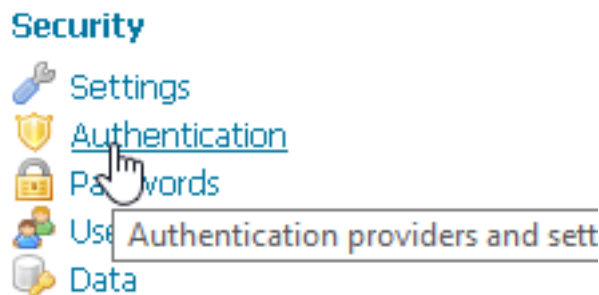
It is necessary now check that GeoServer can connect to OAuth2 Providers (specifically to GeoNode OP), and being able to Authenticate users through it.

Preliminary checks

- GeoServer is up and running and you have admin rights
- GeoServer must reach the GeoNode instance via HTTP
- OAuth2 Client ID and Client Secret have been generated on GeoNode and known

Setup of the GeoNode OAuth2 Security Filter

1. Access the Security > Authentication section



2. If **not yet configured** the Authentication Filter `geonode-oauth2 - Authentication` using a GeoNode OAuth2, click on Authentication Filters > Add new

Nota: This passage is **not** needed if the `geonode-oauth2 - Authentication` using a GeoNode OAuth2 has been already created. If so it will be displayed among the Authentication Filters list

Authentication Filters

[+ Add new](#)[- Remove selected](#)

Search	
☐ Name	Type
☐ anonymous	Anonymous authentication
☐ basic	Basic HTTP authentication
☐ form	Form authentication
☐ geonode-oauth2	Authentication using a GeoNode OAuth2
☐ geonodeAnonymousFilter	org.geonode.security.GeoNodeAnonymousProcessingFilter
☐ geonodeCookieFilter	org.geonode.security.GeoNodeCookieProcessingFilter
☐ rememberme	Remember me authentication
<< < 1 > >> Results 1 to 7 (out of 7 items)	

Authentication Filters



3. If **not yet configured** the Authentication Filter `geonode-oauth2` - Authentication using a GeoNode OAuth2, choose `GeoNode OAuth2` - Authenticates by looking up for a valid GeoNode OAuth2 `access_token` key sent as URL parameter

New Authentication Filter

Create and configure a new Authentication Filter

`J2EE` - Delegates to servlet container for authentication`GeoNode OAuth2` - Authenticates by looking up for a valid GeoNode OAuth2 `access_token` key sent as URL parameter

4. Create / update the `geonode-oauth2` - Authentication using a GeoNode OAuth2 accordingly

- Name; **Must be** `geonode-oauth2`
- Enable Redirect Authentication EntryPoint; It is recommended to put this to `False`, otherwise GeoServer won't allow you to connect to its Admin GUI through the `Form` but only through GeoNode
- Login Authentication EndPoint; Unless you have specific needs, keep the default value `/j_spring_oauth2_geonode_login`

Authentication using a GeoNode OAuth2 geonode-oauth2

Authenticates by looking up for a valid GeoNode OAuth2 access_token key sent as URL parameter

Name	geonode-oauth2	
OAuth2 provider connection		
Enable Redirect Authentication EntryPoint	☐	
Login Authentication EndPoint	/j_spring_oauth2_geonode_login	
Logout Authentication EndPoint	/j_spring_oauth2_geonode_logout	
Force Access Token URI HTTPS Secured Protocol	☐	
Access Token URI		

- Logout Authentication EndPoint; Unless you have specific needs, keep the default value `/j_spring_oauth2_geonode_logout`
- Force Access Token URI HTTPS Secured Protocol; This must be False unless you enabled a Secured Connection on GeoNode. In that case you will need to trust the GeoNode Certificate on the GeoServer JVM Keystore. Please see details below
- Access Token URI; Set this to `http://<geonode_host_base_url>/o/token/`
- Force User Authorization URI HTTPS Secured Protocol; This must be False unless you enabled a Secured Connection on GeoNode. In that case you will need to trust the GeoNode Certificate on the GeoServer JVM Keystore. Please see details below
- User Authorization URI; Set this to `http://<geonode_host_base_url>/o/authorize/`
- Redirect URI; Set this to `http://<geoserver_host>/geoserver`. This address **must** be present on the Redirect uris of GeoNode OAuth2 > Applications > GeoServer (see above)
- Check Token Endpoint URL; Set this to `http://<geonode_host_base_url>/api/o/v4/tokeninfo/`
- Logout URI; Set this to `http://<geonode_host_base_url>/account/logout/`
- Scopes; Unless you have specific needs, keep the default value `read,write,groups`
- Client ID; The Client id alphanumeric key generated by the GeoNode OAuth2 > Applications > GeoServer (see above)
- Client Secret; The Client secret alphanumeric key generated by the GeoNode OAuth2 > Applications > GeoServer (see above)
- Role source; In order to authorize the user against GeoNode, choose Role service > geonode REST role service

Configuration of the GeoServer Filter Chains

The following steps ensure GeoServer can adopt more Authentication methods. As stated above, it is possible to Authenticate to GeoServer using different protocols.

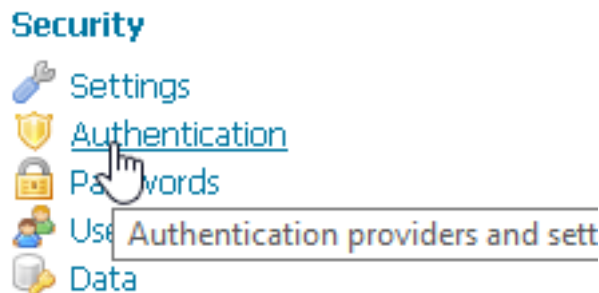
GeoServer scans the authentication filters chain associated to the specified path and tries them one by one sequentially. The first one matching the protocol and able to grant access to the user, breaks the cycle by creating a `User Principal` and injecting it into the `GeoServer SecurityContext`. The Authentication process, then, ends here and the control goes to the Authorization one, which will try to retrieve the authenticated user's Roles through the available GeoServer Role Services associated to the Authentication Filter that granted the access.

Preliminary checks

- GeoServer is up and running and you have admin rights
- GeoServer must reach the GeoNode instance via HTTP
- The `geonode-oauth2` - Authentication using a GeoNode OAuth2 Authentication Filter and the `geonode REST role service` have been correctly configured

Setup of the GeoServer Filter Chains

1. Access the `Security > Authentication` section



2. Identify the section `Filter Chains`
3. Make sure the web `Filter Chain` is configured as shown below

Aviso: Every time you modify a Filter Chain, **don't forget to save** the Authentication settings. This **must** be done for **each** change.

Filter Chains

 [Add service chain](#)

 [Add HTML chain](#)

Position	Name	Patterns
	web	/web/**,/gwc/rest/web/**,/,
 	webLogin	/j_spring_security_check,/j_spring_security_check/,/,
 	webLogout	/j_spring_security_logout,/j_spring_security_logout/,/,
 	rest	/rest/**
 	gwc	/gwc/rest/**
	default	/**



1


 Results 1 to 6 (out of 6 items)

Available		Selected
basic geonodeAnonymousFilter geonodeCookieFilter	   	geonode-oauth2 rememberme form anonymous

☐ georence org.geoserver.geoserver.autr
☒ geonodeAuthProvider org.geonode.security.GeoNode

<< < 1 > >> Results 1 to 3 (out of 3 items)

Provider Chain

Available	
geonodeAuthProvider	⇒ ⇐ ⇕ ⇓

4. Make sure the rest Filter Chain is configured as shown below

Available		Selected
geonodeAnonymousFilter geonodeCookieFilter	⇒ ⇐ ⇕ ⇓	basic geonode-oauth2 anonymous

Aviso: Every time you modify a Filter Chain, **don't forget to save** the Authentication settings. This **must** be done for **each** change.

☐ georence org.geoserver.geoserver.auth
☒ geonodeAuthProvider org.geonode.security.GeoNodeAuthProvider

<< < 1 > >> Results 1 to 3 (out of 3 items)

Provider Chain

Available	Selected
geonodeAuthProvider	

⇒
 ⇐
 ⇑
 ⇓

Save Cancel

5. Make sure the gwc Filter Chain is configured as shown below

Available		Selected
anonymous	⇒	basic
geonodeAnonymousFilter	⇐	geonode-oauth2
geonodeCookieFilter	⇑	
	⇓	

Aviso: Every time you modify a Filter Chain, **don't forget to save** the Authentication settings. This **must** be done for **each** change.

☐ georence org.geoserver.geoserver.autr
☒ geonodeAuthProvider org.geonode.security.GeoNo

<< < 1 > >> Results 1 to 3 (out of 3 items)

Provider Chain

Available	
geonodeAuthProvider	⇒ ⇐ ⬆ ⬇

6. Make sure the default Filter Chain is configured as shown below

Available		Selected
geonodeAnonymousFilter geonodeCookieFilter	⇒ ⇐ ⬆ ⬇	basic geonode-oauth2 anonymous

Aviso: Every time you modify a Filter Chain, **don't forget to save** the Authentication settings. This **must** be done for **each** change.

The screenshot displays a web interface for configuring the security provider chain. At the top, there is a list of available providers with checkboxes. The first provider is 'georence' with the class 'org.geoserver.geoserver.auth'. The second provider, 'geonodeAuthProvider', with the class 'org.geonode.security.GeoNodeAuthProvider', is highlighted in green. Below this list are navigation buttons: '<<', '<', '1', '>', and '>>', followed by the text 'Results 1 to 3 (out of 3 items)'. The main section is titled 'Provider Chain' and contains a table with a header 'Available'. Below the header, the provider 'geonodeAuthProvider' is listed in a scrollable box. To the right of this box are four buttons for moving providers: '⇒', '⇐', '⇕', and '⇓'. At the bottom of the interface are two buttons: 'Save' and 'Cancel'. A mouse cursor is pointing at the 'Save' button.

Available
geonodeAuthProvider

Save **Cancel**

7. Add the GeoNode Login Endpoints to the comma-delimited list of the webLogin Filter Chain

Aviso: Every time you modify a Filter Chain, **don't forget to save** the Authentication settings. This **must** be done for **each** change.

Filter chain

Configure an individual filter chain

Chain settings

Name

Comma delimited list of ANT patterns (with optional query string)

- ☐ Disable security for this chain
- ☐ Allow creation of an HTTP session for storing the authentication token
- ☐ Accept only SSL requests

Role filter

☐ georence org.geoserver.geoserver.auth
☒ geonodeAuthProvider org.geonode.security.GeoNodeAuthProvider

<< < 1 > >> Results 1 to 3 (out of 3 items)

Provider Chain

Available
geonodeAuthProvider

Save Cancel

8. Add the GeoNode Logout Endpoints to the comma-delimited list of the webLogout Filter Chain

Aviso: Every time you modify a Filter Chain, **don't forget to save** the Authentication settings. This **must** be done for **each** change.

Filter chain

Configure an individual filter chain

Chain settings

Name

Comma delimited list of ANT patterns (with optional query string)

- ☐ Disable security for this chain
- ☐ Allow creation of an HTTP session for storing the authentication
- ☐ Accept only SSL requests

Role filter

9. Add the GeoNode Logout Endpoints to the comma-delimited list of the `formLogoutChain` XML node in `<GEOSERVER_DATA_DIR>/security/filter/formLogout/config.xml`

You will need a text editor to modify the file.

Nota: If the `<formLogoutChain>` XML node does not exist at all, create a **new one** as specified below

```
<logoutFilter>
...
<redirectURL>/web/</redirectURL>
<formLogoutChain>/j_spring_security_logout,/j_spring_security_logout/,/
↪j_spring_oauth2_geonode_logout,/j_spring_oauth2_geonode_logout/</
↪formLogoutChain>
</logoutFilter>
```

Aviso: The value `j_spring_oauth2_geonode_logout` **must** be the same specified as Logout Authentication EndPoint in the `geonode-oauth2 - Authentication` using a GeoNode OAuth2 above.

Setup and test of the GeoFence Server and Default Rules

In order to work correctly, GeoServer needs the [GeoFence Embedded Server](#) plugin to be installed and configured on the system.

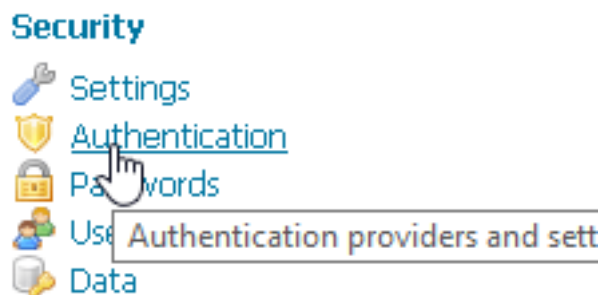
The GeoServer configuration provided for GeoNode, has the plugin already installed with a default configuration. In that case double check that the plugin works correctly and the default rules have been setup by following the next steps.

Preliminary checks

- GeoServer is up and running and you have admin rights
- The [GeoFence Embedded Server](#) plugin has been installed on GeoServer

Setup of the GeoServer Filter Chains

1. Access the `Security > Authentication` section



2. Identify the section `Authentication Providers` and make sure the `geofence Authentication Provider` is present

Authentication Providers

Add new

Remove selected

Search

☐	Name	Type
☐	default	Basic username/password authentication
☐	geofence	org.geoserver.geoserver.authentication.auth.GeoFenceAuthenticationProvider
☐	geonodeAuthProvider	org.geonode.security.GeoNodeAuthenticationProvider

<<
<
1
>
>>
Results 1 to 3 (out of 3 items)

3. Make sure the Provider Chain is configured as shown below

Provider Chain

Available		Selected
geonodeAuthProvider	⇒	default geofence
	⇐	
	↑↑	
	↓↓	

Aviso: Every time you modify an Authentication Providers, **don't forget to save** the Authentication settings. This **must** be done for **each** change.

☐ geofence org.geoserver.geoserver.auth

☒ geonodeAuthProvider org.geonode.security.GeoNode

<< < 1 > >> Results 1 to 3 (out of 3 items)

Provider Chain

Available	
geonodeAuthProvider	⇒
	⇐
	↑↑
	↓↓

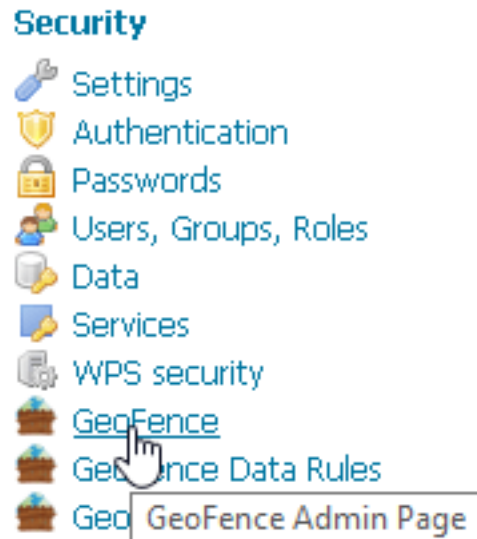
Save

Cancel

Setup of the GeoFence Server and Rules

1. Make sure GeoFence server works and the default settings are correctly configured

- Access the Security > GeoFence section



- Make sure the Options are configured as follows and the server works well when performing a Test Connection
 - Allow remote and inline layers in SLD; Set it to True
 - Allow SLD and SLD_BODY parameters in requests; Set it to True
 - Authenticated users can write; Set it to True
 - Use GeoServer roles to get authorizations; Set it to False

2. Check the GeoFence default Rules

- Access the Security > GeoFence Data Rules section
- Make sure the DENY ALL Rule is present by default, otherwise your data will be accessible to everyone

Nota: This rule is **always** the last one

Aviso: If that rule does not exists **at the very bottom** (this rule is **always** the last one), add it manually.

- Access the Security > GeoFence Admin Rules section
- No Rules needed here

Connection successful

GeoFence Admin Page

GeoFence options Administration Page

General settings

GeoServer Instance name for GeoFence

default-gs

GeoFence services URL (GeoServer restart is required if changed)

internal:/

Test Connection

Options

- ☒ Allow remote and inline layers in SLD
- ☒ Allow SLD and SLD_BODY parameters in requests
- ☒ Authenticated users can write
- ☐ Use GeoServer roles to get authorizations



GeoFence Data Rules

Configure data rules for the internal GeoFence server.

- Add new rule
- Remove selected rules

<< < 1 > >>

Results 1 to 1 (out of 1 items)

Search

☐	P	Role	User	Service	Request	Workspace	Layer	Access	
☐	0	*	*	*	*	*	*	DENY	

<< < 1 > >>

Results 1 to 1 (out of 1 items)



GeoFence Admin Rules

Configure admin rules for the internal GeoFence server.

-  Add new rule
-  Remove selected rules

<<

<

>

>>

Results 0 to 0 (out of 0 items)

Search

	P	Role	User	Workspace	Access
------------------------	---	------	------	-----------	--------

<<

<

>

>>

Results 0 to 0 (out of 0 items)

Troubleshooting and Advanced Features

Common Issues and Fixes

- GeoServer/GeoNode OAuth2 does not authenticate as Administrator even using GeoNode admin users

Symptoms

When trying to authenticate with an `admin` user using OAuth2, the process correctly redirects to GeoServer page but I'm not a GeoServer Administrator.

Cause

That means that somehow GeoServer could not successfully complete the Authorization and Authentication process.

The possible causes of the problem may be the following ones:

1. The OAuth2 Authentication fails on GeoServer side

This is usually due to an exception while trying to complete the Authentication process.

- A typical cause is that GeoServer tries to use HTTPS connections but the GeoNode certificate is not trusted;

In that case please refer to the section below. Also take a look at the logs (in particular the GeoServer one) as explained in `debug_geonode`. The GeoServer logs should contain a detailed Exception explaining the cause of the problem. If no exception is listed here (even after raised the log level to *DEBUG*), try to check for the GeoNode Role Service as explained below.

- Another possible issue is that somehow the OAuth2 handshake cannot complete successfully;

1. Login into GeoServer as administrator through its WEB login form.

2. Double check that all the `geonode-oauth2 - Authentication` using a GeoNode OAuth2 parameters are correct. If everything is ok, take a look at the logs (in particular the GeoServer one) as explained in `debug_geonode`. The GeoServer logs should contain a detailed Exception explaining the cause of the problem. If no exception is listed here (even after raised the log level to *DEBUG*), try to check for the GeoNode Role Service as explained below.

2. GeoServer is not able to retrieve the user Role from a Role Service

Always double check both HTTP Server and GeoServer log as specified in section `debug_geonode`. This might directly guide you to the cause of the problem.

- Check that the GeoServer host is granted to access GeoNode Role Service REST APIs in the `AUTH_IP_WHITELIST` of the `settings.py`
- Check that the `geonode REST role service` is the default Role service and that the GeoServer OAuth2 Plugin has been configured to use it by default
- Check that the GeoNode REST Role Service APIs are functional and produce correct JSON.

This is possible by using simple `cUrl` GET calls like

```
curl http://localhost/api/adminRole
$> {"adminRole": "admin"}

curl http://localhost/api/users
$> {"users": [{"username": "AnonymousUser", "groups":
↪": ["anonymous"]}, {"username": "afabiani",
↪"groups": ["anonymous", "test"]}, {"username":
↪"admin", "groups": ["anonymous", "test", "admin"]}
↪]}

curl http://localhost/api/roles
$> {"groups": ["anonymous", "test", "admin"]}

curl http://localhost/api/users/admin
$> {"users": [{"username": "admin", "groups": [
↪"anonymous", "test", "admin"]}]}
```

How to setup HTTPS secured endpoints

In a production system it is a good practice to encrypt the connection between GeoServer and GeoNode. That would be possible by enabling HTTPS Protocol on the GeoNode REST Role Service APIs and OAuth2 Endpoints.

Most of the times you will rely on a self-signed HTTPS connection using a generated certificate. That makes the connection *untrusted* and you will need to tell to the GeoServer Java Virtual Machine to trust it.

This can be done by following the steps below.

For any issue take a look at the logs (in particular the GeoServer one) as explained in `debug_geonode`. The GeoServer logs should contain a detailed Exception explaining the cause of the problem.

SSL Trusted Certificates

When using a custom `Keystore` or trying to access a non-trusted or self-signed SSL-protected OAuth2 Provider from a non-SSH connection, you will need to add the certificates to the JVM `Keystore`.

In order to do this you can follow the next steps:

In this example we are going to

1. Retrieve SSL Certificate from GeoNode domain:

«Access Token URI» = https://<geonode_host_base_url>/o/token/ therefore we need to trust https://<geonode_host_base_url> or (<geonode_host_base_url>:443)

Nota: You will need to get and trust certificates from every different HTTPS URL used on OAuth2 Endpoints.

2. Store SSL Certificates on local hard-disk
 3. Add SSL Certificates to the Java Keystore
 4. Enable the JVM to check for SSL Certificates from the Keystore
1. Retrieve the SSL Certificate from GeoNode domain

Use the `openssl` command in order to dump the certificate

For `https://<geonode_host_base_url>`

```
openssl s_client -connect <geonode_host_base_url>:443
```

```

Loading 'screen' into random state - done
CONNECTED(00000188)
depth=3 C = US, O = Equifax, OU = Equifax Secure Certificate Authority
verify return:1
depth=2 C = US, O = GeoTrust Inc., CN = GeoTrust Global CA
verify return:1
depth=1 C = US, O = Google Inc, CN = Google Internet Authority G2
verify return:1
depth=0 C = US, ST = California, L = Mountain View, O = Google Inc, CN = accounts.google.com
verify return:1
---
Certificate chain
 0 s:/C=US/ST=California/L=Mountain View/O=Google Inc/CN=accounts.google.com
  i:/C=US/O=Google Inc/CN=Google Internet Authority G2
 1 s:/C=US/O=Google Inc/CN=Google Internet Authority G2
  i:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
 2 s:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
  i:/C=US/O=Equifax/OU=Equifax Secure Certificate Authority
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIEoTCCA4mgAwIBAgIIeEP870cN1RQwDQYJKoZIhvcNAQELBQAwSTELMAkGA1UE
BhMCVVMxExZARBgNVBAoTCkdvb2dsZSBSBjBmMxJTAjBgNVBAMTHEdvd2dsZSBSBjBnR1
cm5ldCBDbXR0b3JpdHkgRzIwHhcNMjYxMDE5MTUxMjUzWWhcNMjYxMDE5MTUxMjUzWWhc
UjBtMQswCQYDVQQGEwJVUzETMBEGA1UECAwKQ2FsaWZvcn5pYTEwMDQGA1UEBwwN
TW91bnRhaW4gVmllZzETMBEGA1UECgwKR29vZ2x2Z2x1IEluYzEwMDQGA1UEAwwTYWNj
b3VudHMud29vZ2x2Z2x1LmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
AK71x4dYCS7NcMqyp8UAq7067Wb7LhkgE5QJcUVdvFfLrXmG3PAIcfiLMvdIK+F
RODn5d79I1h8qgn90XgB/xyFgx381MR/HoglavblCTfgbiotvhaMxxyY/h55efbcn
i2GGOXCWVcx119xPONwFvd7aUaOLSutMeEZ7T4Hqj3o0WFYVJq+dQNGdkGwgtUap
bosQFuyWg+rKINM6Opvy6ay8fRU/4JeKNVca2UuIu12bPchBzIXiKtLkMJSIYZRz
ySjJmFLm++/ipMQ549xeyviB4pS244athBuIgeNHgE8Xb7eTS5TPuWfJ913nhdT
IDsalPMX2PpsinA1IF82+kECAwEAaOACWcwggFjMBOGA1UdJQQWMBQGCCSGAQUF
BzMBBggrBgEFBQcDAjA1BgNVHREELjAsghNhy2NvdW50cy5nb29nbGUuY29tcgUg
-----END CERTIFICATE-----

```

- ## 2. Store SSL Certificate on local hard-disk

Copy-and-paste the section `-BEGIN CERTIFICATE-`, `-END CERTIFICATE-` and save it into a `.cert` file

Nota: .cert file are plain text files containing the ASCII characters included on the -BEGIN CERTIFICATE-, -END CERTIFICATE- sections

geonode.cert (or whatever name you want with .cert extension)

```

1 -----BEGIN CERTIFICATE-----
2 MIIEoTCCA4mgAwIBAgIIeEP870cN1RQwDQYJKoZIhvcNAQELBQAwSTELMAkGA1UE
3 BhMCVVMxMzEzARBgNVBAMTCkdvb2dsZSBjb250aW50aW50aW50aW50aW50aW50
4 cm5ldCBkdXRob3RpdHkgRzIwHhcNMjYxMDE5MTcxNjU3WhcNMjYxMDE5MTcxMzAw
5 WjBtMQswCQYDVQGEwJVUzETMBEGA1UECgwwK2FsaWZlc3R5TEWMBQGA1UEBwwN
6 TW91bnRhaW4gVm1ldzETMBEGA1UECgwwK2FsaWZlc3R5TEWMBQGA1UEBwwN
7 b3VudHMud29vZ2x1LmNvbTCCASIwDQYJKoZIhvcNAQEBBQAdggEPADCCAQoCggEB
8 AK71x4DYCS7NcMqyp8Uaq7067Wb7LLhkgE5QJcUVdvfLrXmG3PAIcfiLMvdIK+F
9 RODn5d79I18qgn90XgB/xyFgx381MR/Hoglavb1CTfghiotvhAMsxyY/h55efbcn
10 i2GGOXCWVcx119xPONWfvd7aUa0LSutMeE27T4Hqj3o0WFYVJq+dQNGdkGwqtUap
11 bosQFuyWg+KINM6Opvy6ay8fRU/4JeKNVcaZUuIu12bPchBzIXiKtLkMJSIYzRz
12 ySjJmFlm++/ipMQ549xeyviB4pSz44athBuIgeNhgE8Xb7eTS5TPuWFjJ913nhdT
13 IDsa1PMX2PpsinA1IF8Z+kECaWAAaOCaWcgwGfjMBOGA1UdJQQWMBQGCCsGAQUF
14 BwMBBggrBgEFBQcDAjA1BgNVHREELjAsghNhy2NvdW50cy5nb29nbGUuY29tghUq
15 LnBhcnRuZXXIuYW5kcm9pZC5jb20waAYIKwYBBQUHAQEEXDBaMCsGCCsGAQUFBzAC
16 hh9odHRwOi8vcGtPlmdvb2dsZS5jb20vRO1BRzIuY3JOMCsGCCsGAQUFBzABhh9o
17 dHRwOi8vY2xpZW50cy5nb29vZ2x1LmNvbS9vY3NwMBQGA1UdDgQWBBsf3MI/auhw
18 EZz38qRdr+B1Le1iyTAMBgNVHRMBAf8EAjAAMB8GA1UdIwQYMBaAFerdBhYbvPZo
19 tXb1gba7Yhq6WoEvMCEGA1UdIAQaMBgwDAYKKwYBBAHWeQIFATAIBgzngQwBAGIw
20 MAYDVROfBCkwIzAlcOqIVYfaHR0cDovL3BraS5nb29nbGUuY29tLjQdJQIcwlNw

```

3. Add SSL Certificates to the Java Keystore

You can use the Java command `keytool` like this

geonode.cert (or whatever name you want with .cert extension)

```
keytool -import -noprompt -trustcacerts -alias geonode -
  ↪ file geonode.cert -keystore ${KEYSTOREFILE} -storepass $
  ↪ {KEYSTOREPASS}
```

or, alternatively, you can use some graphic tool which helps you managing the SSL Certificates and Keystores, like [Portecle](#)

```
java -jar c:\apps\portecle-1.9\portecle.jar
```

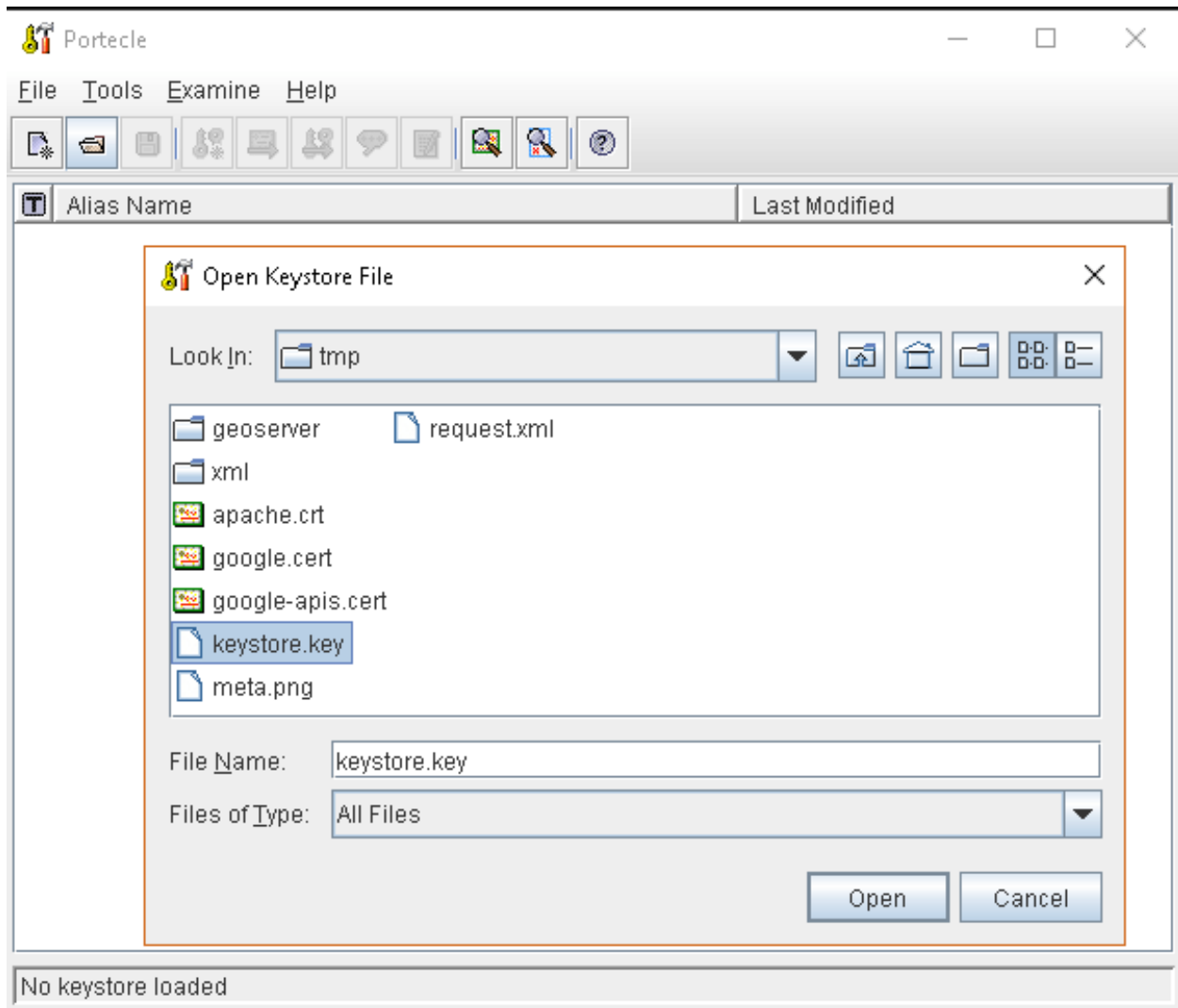
4. Enable the JVM to check for SSL Certificates from the Keystore

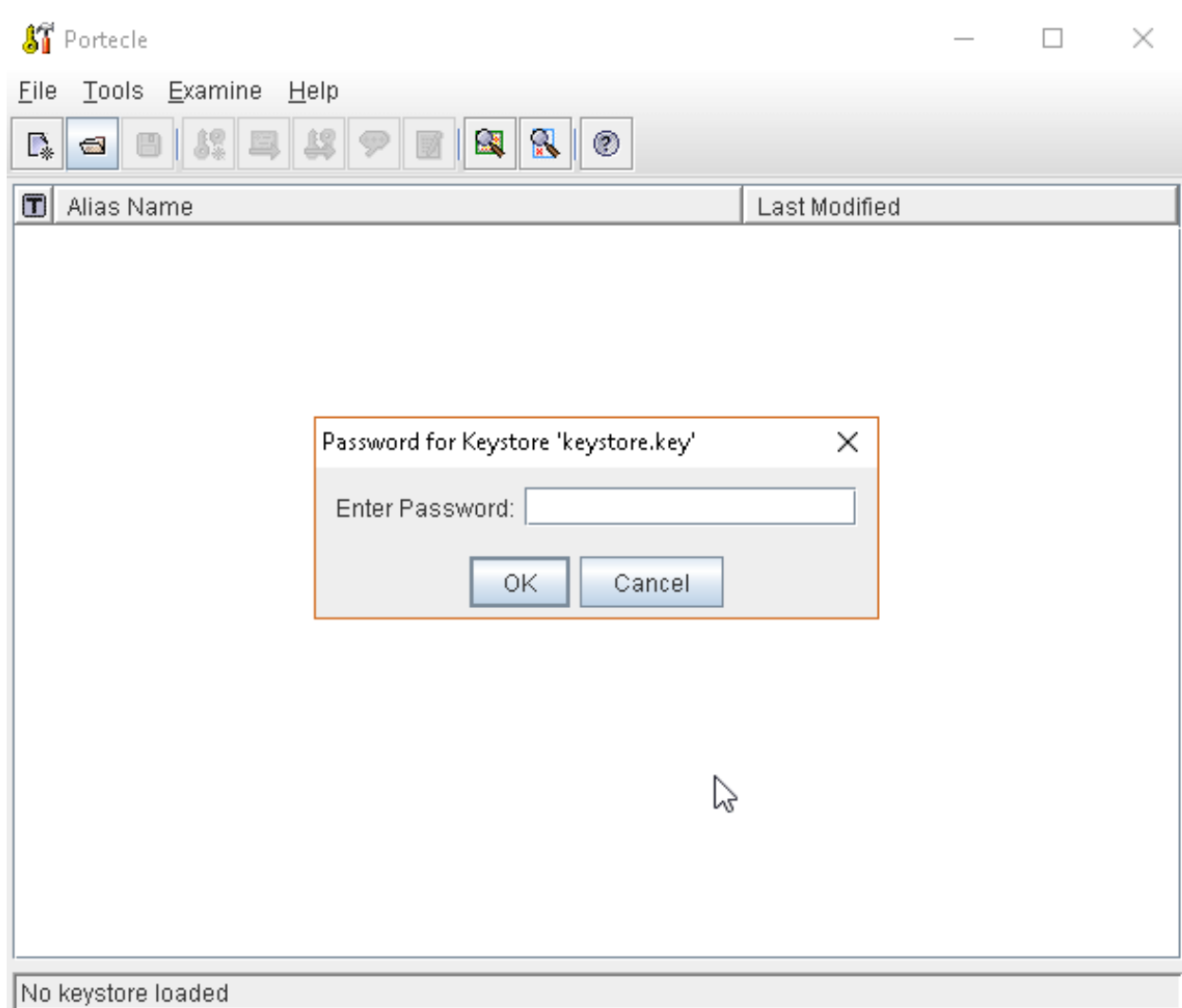
In order to do this, you need to pass a `JAVA_OPTION` to your JVM:

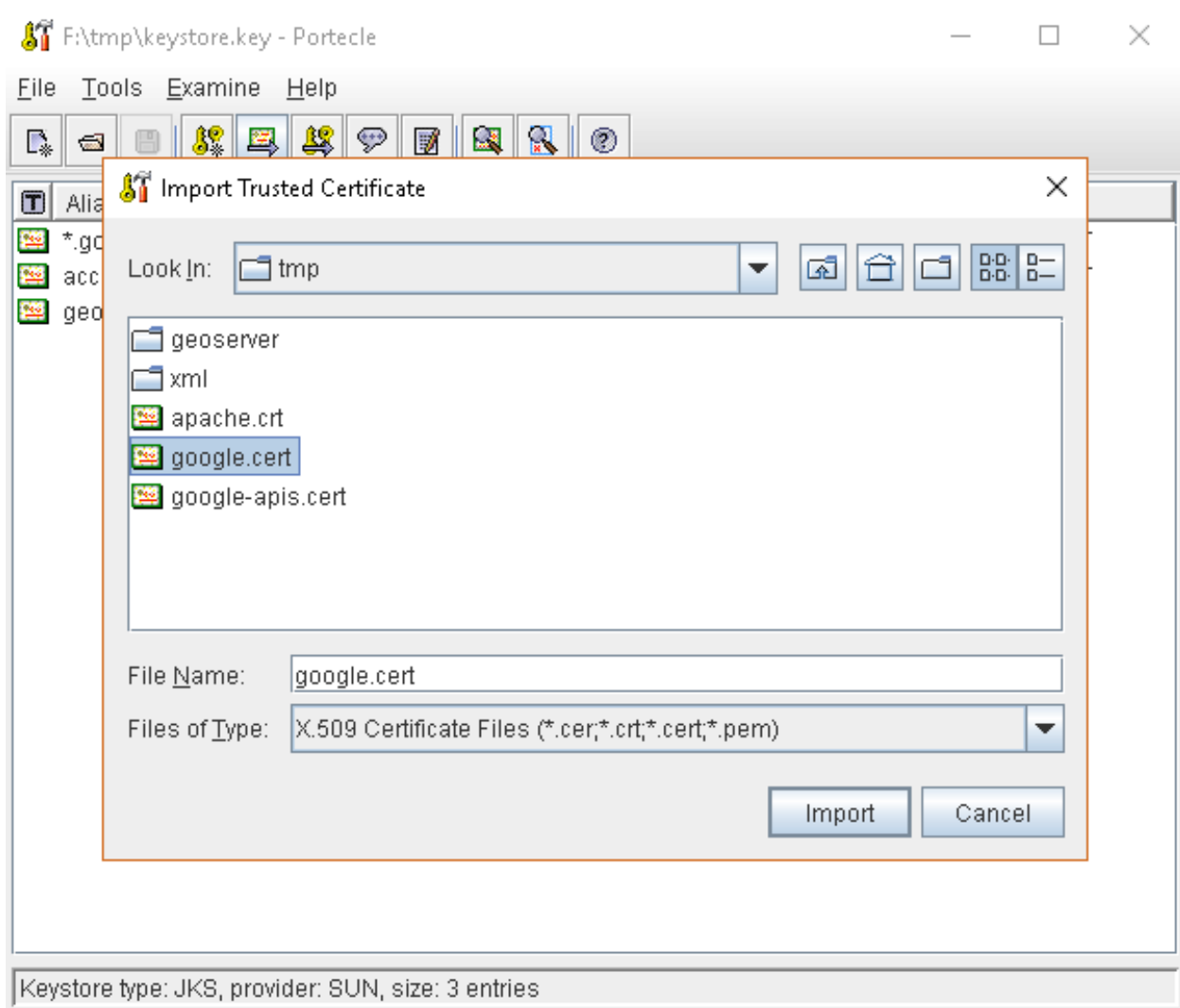
```
-Djavax.net.ssl.trustStore=F:\tmp\keystore.key
```

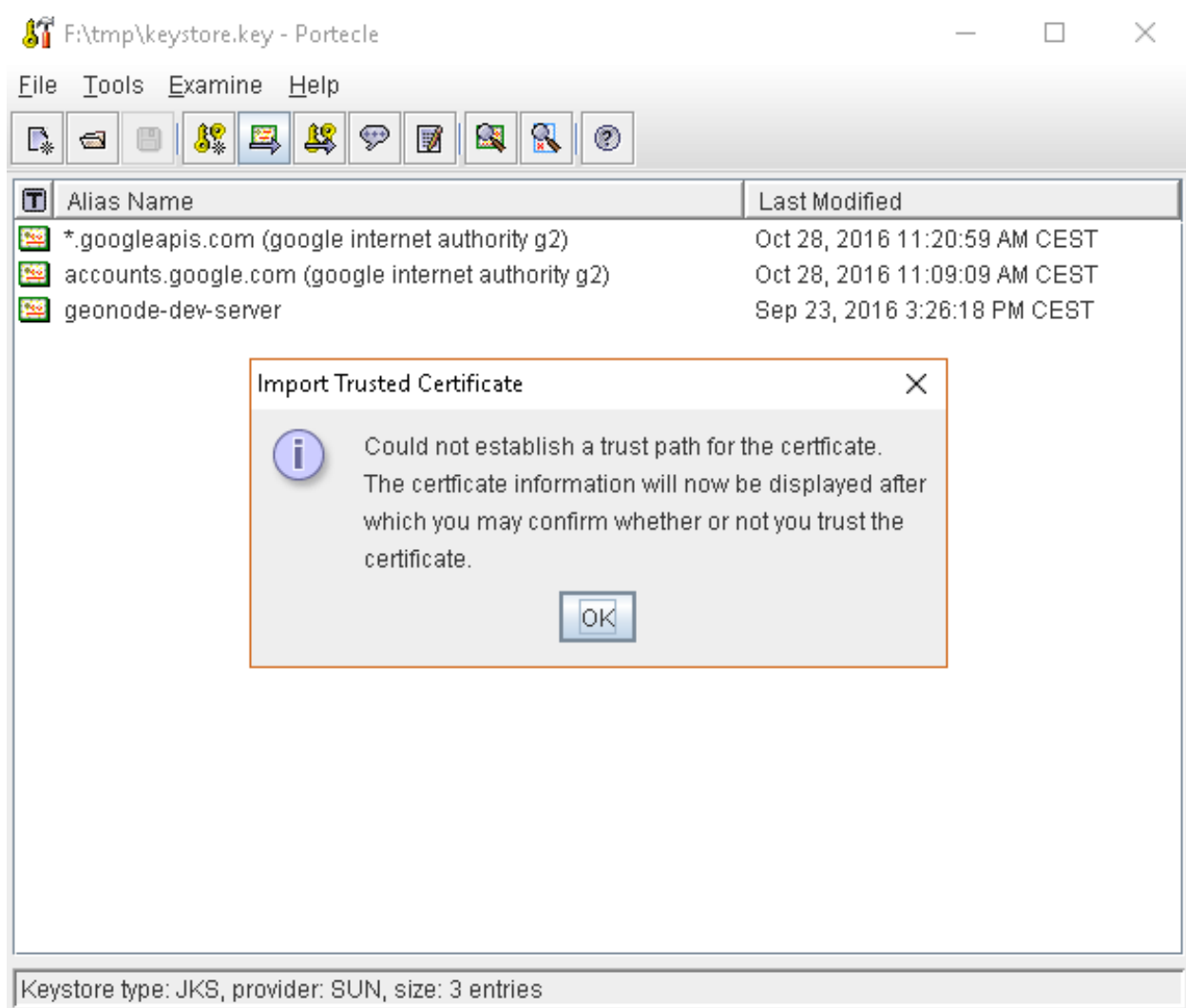
5. Restart your server

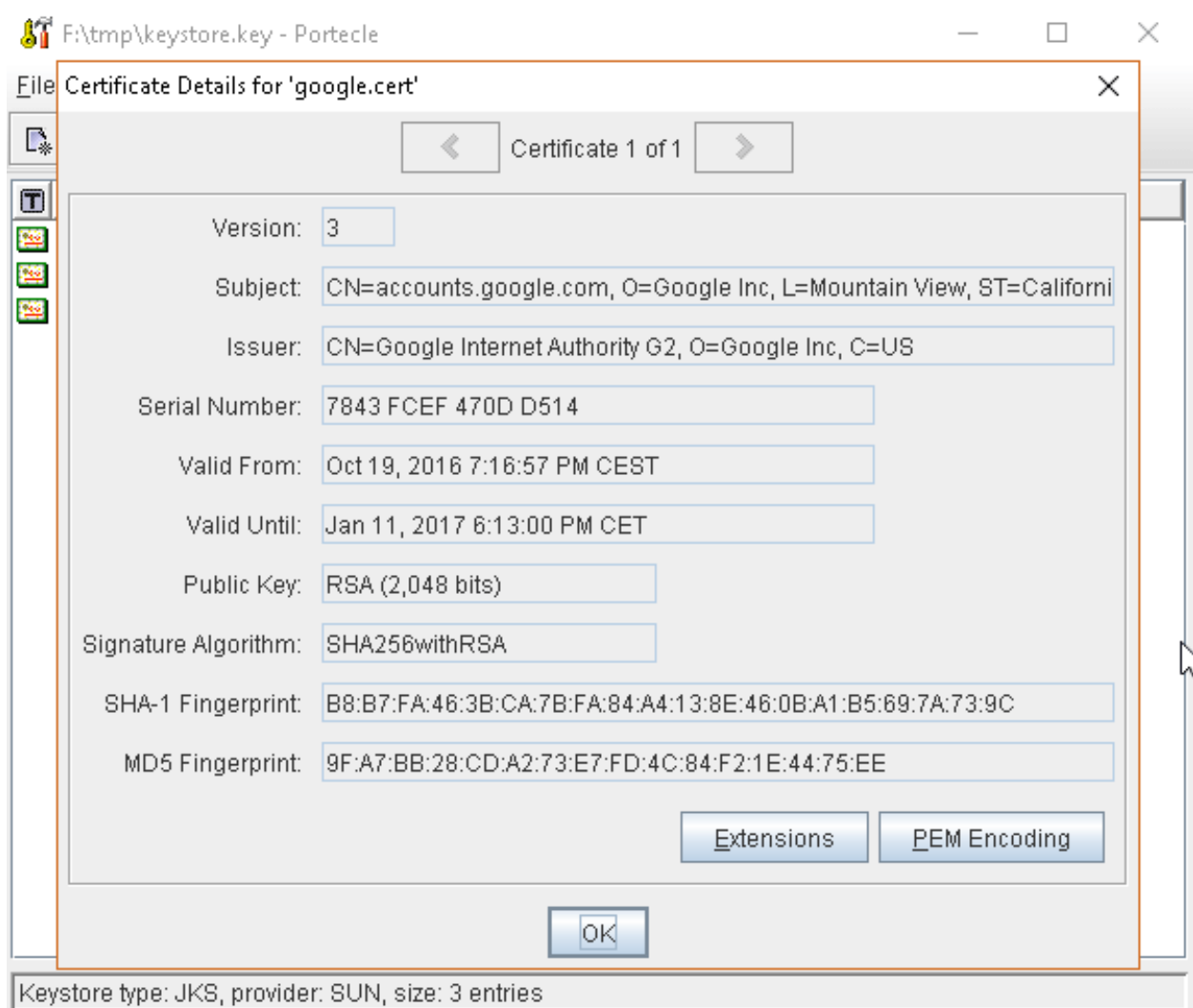
Nota: Here below you can find a bash script which simplifies the Keystore SSL Certificates importing. Use it at your convenience.

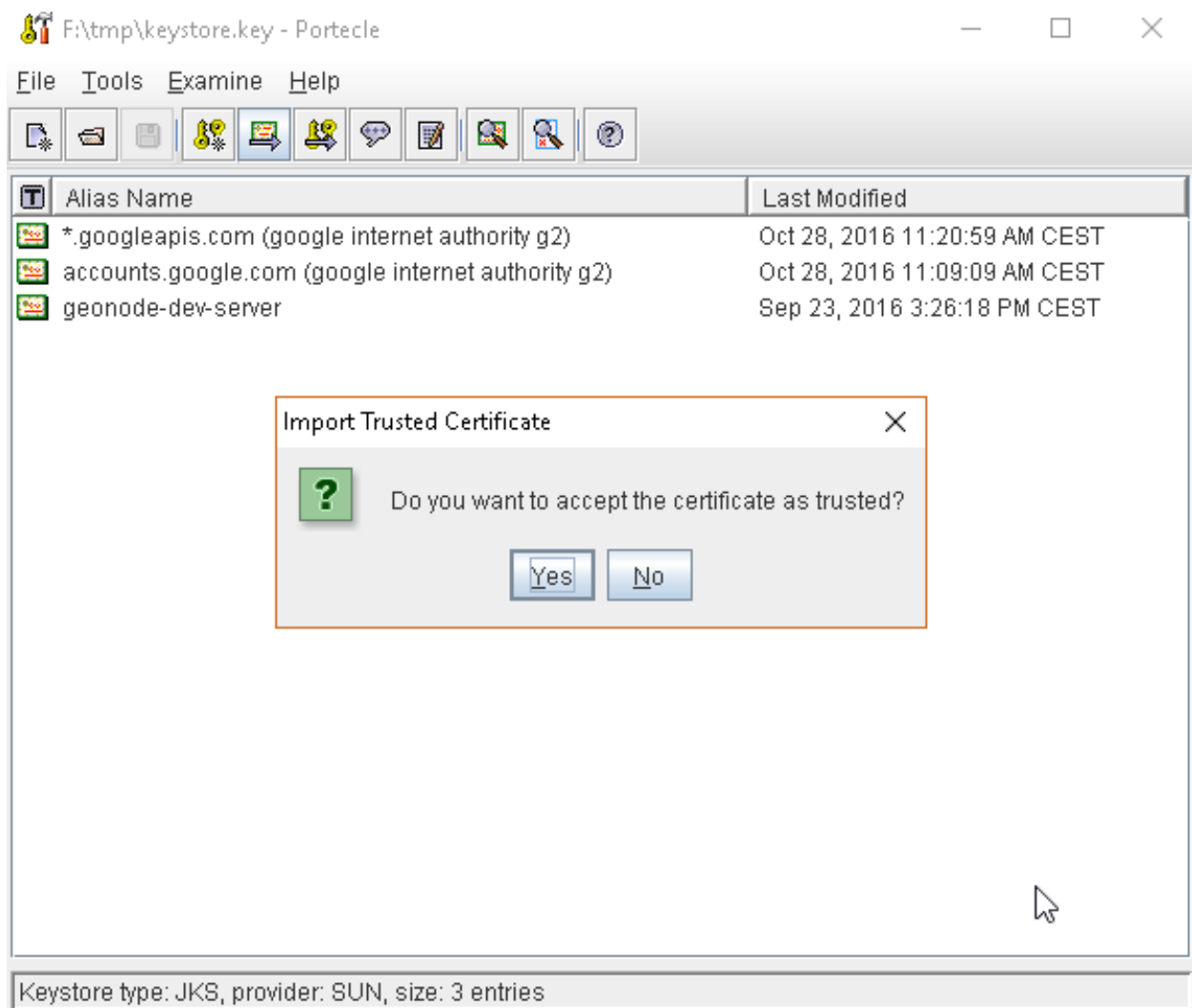


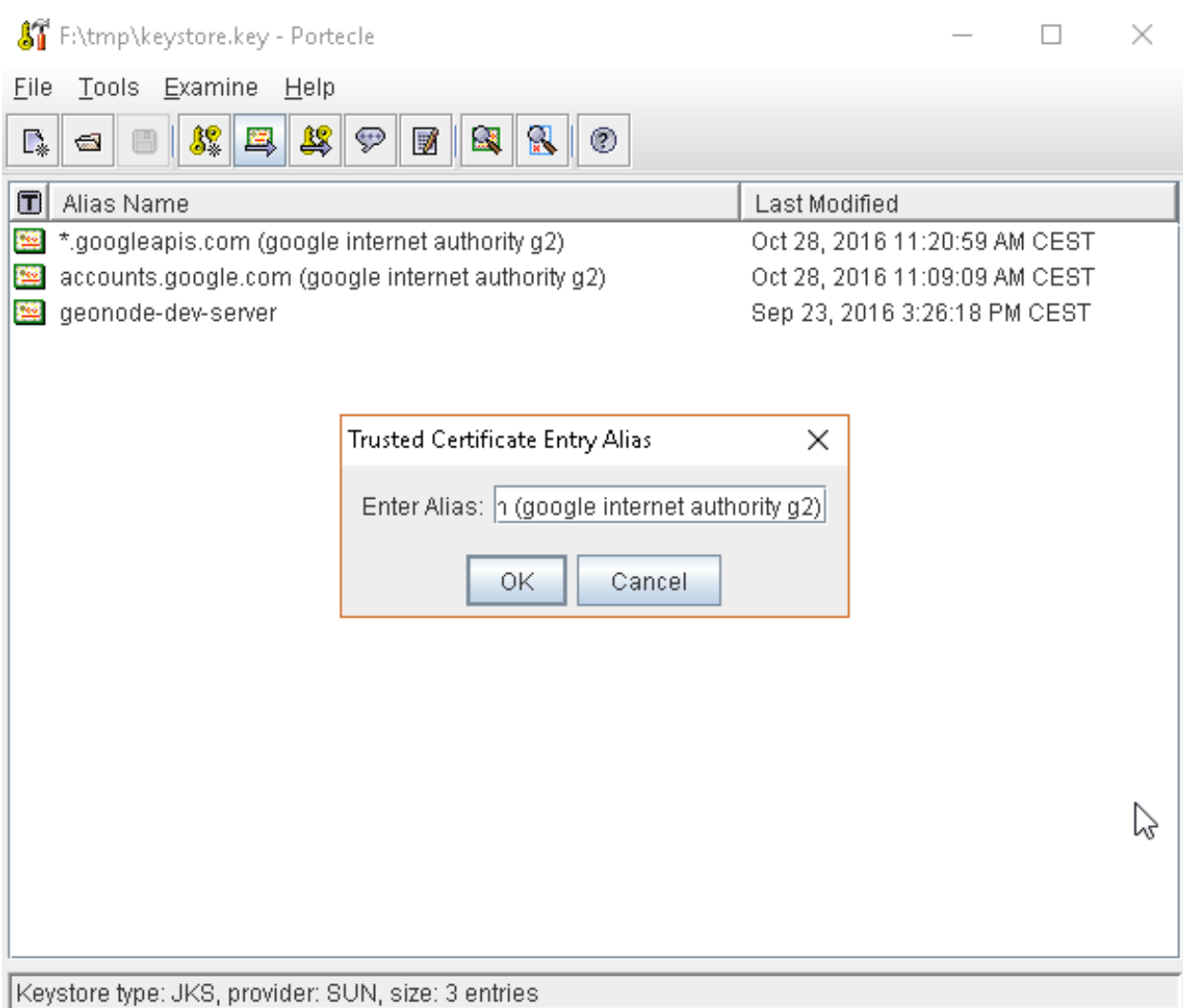


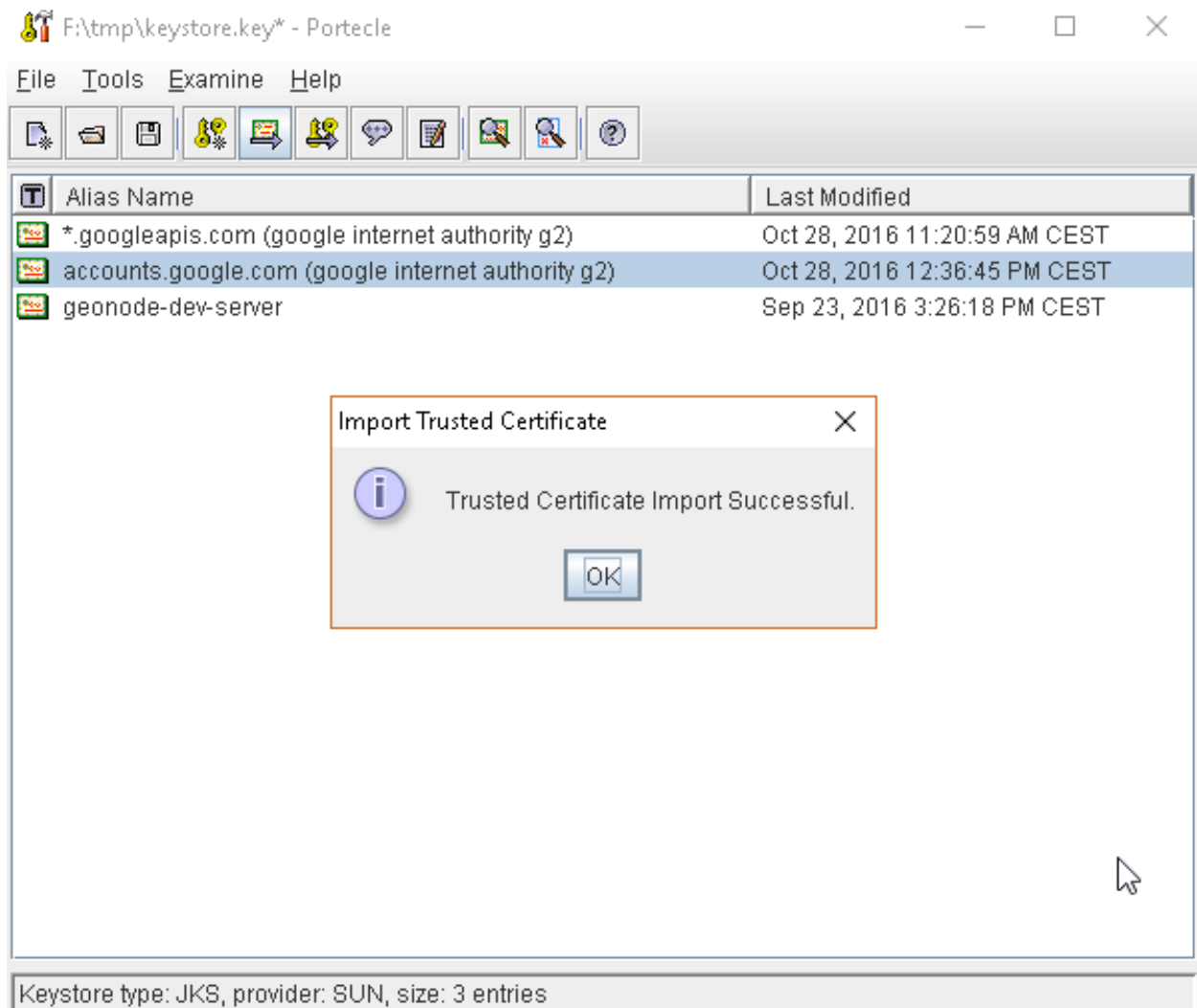


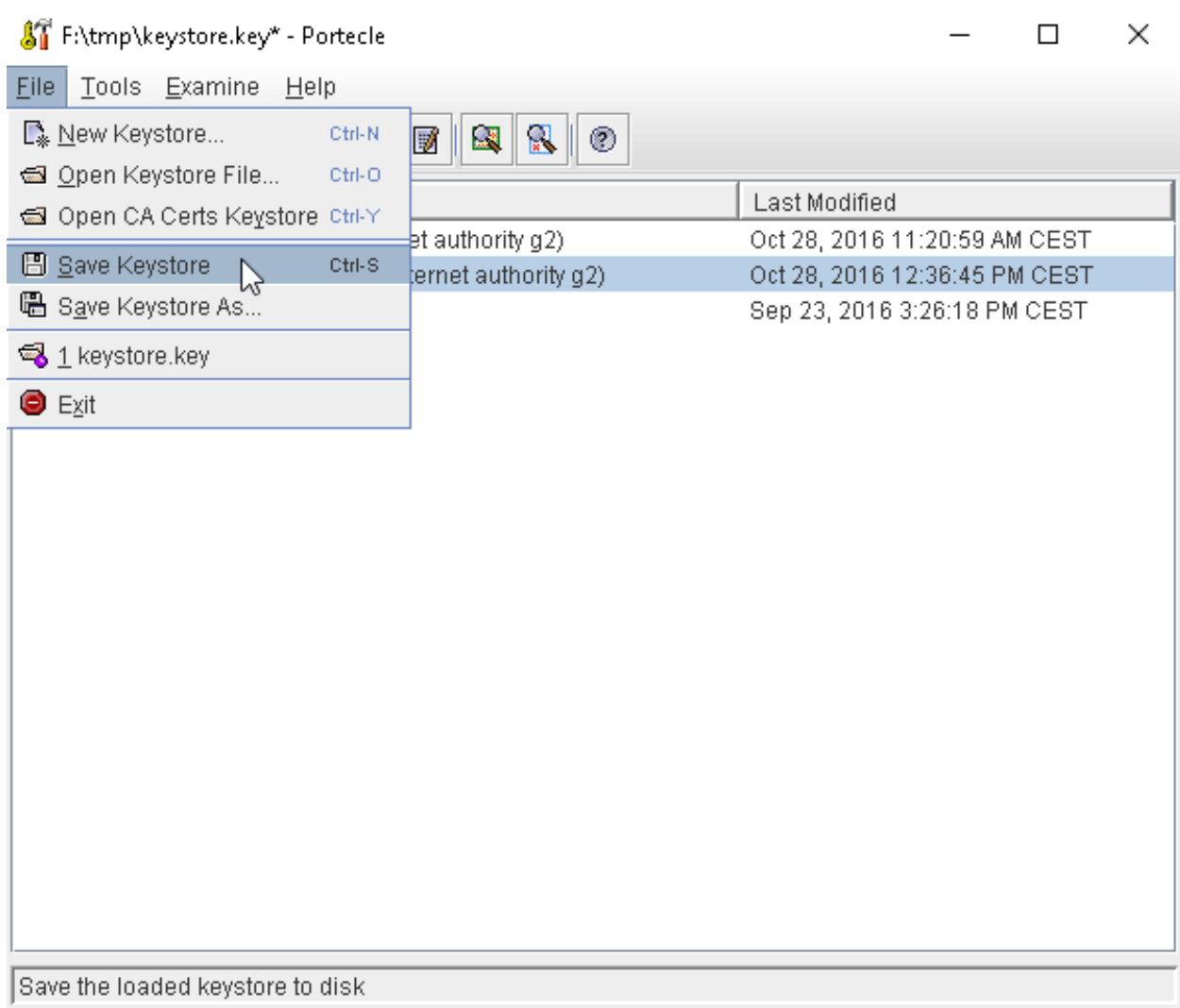












```
HOST=myhost.example.com
PORT=443
KEYSTOREFILE=dest_keystore
KEYSTOREPASS=changeme

# get the SSL certificate
openssl s_client -connect ${HOST}:${PORT} </dev/null \
    | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > ${HOST}.cert

# create a keystore and import certificate
keytool -import -noprompt -trustcacerts \
    -alias ${HOST} -file ${HOST}.cert \
    -keystore ${KEYSTOREFILE} -storepass ${KEYSTOREPASS}

# verify we've got it.
keytool -list -v -keystore ${KEYSTOREFILE} -storepass ${KEYSTOREPASS} -alias ${HOST}
```

GeoFence Advanced Features

GeoFence Rules Management and Tutorials

- This [tutorial](#) shows how to install and configure the Geofence Internal Server plug-in. It shows how to create rules in two ways: using the GUI and REST methods.
- GeoFence Rules can be created / updated / deleted through a REST API, accessible only by a GeoServer Admin user. You can find more details on how the GeoFence REST API works [here](#).

GeoFence Rules Storage Configuration

By default GeoFence is configured to use a filesystem based DB stored on the GeoServer Data Dir <GEOSERVER_DATA_DIR/geofence.

- It is possible also to configure GeoFence in order to use an external PostgreSQL / PostGIS Database. For more details please refer to the official GeoFence documentation [here](#).

1. Add Java Libraries to GeoServer

```
wget --no-check-certificate "https://www.dropbox.com/s/ilowulvd27j2cs1/
↳hibernate-spatial-postgis-1.1.3.2.jar?dl=1" -O hibernate-spatial-
↳postgis-1.1.3.2.jar
wget --no-check-certificate "https://www.dropbox.com/s/psolxleimaft0t7/
↳postgis-jdbc-1.3.3.jar?dl=1" -O postgis-jdbc-1.3.3.jar

cp hibernate-spatial-postgis-1.1.3.2.jar <GEOSERVER_WEBAPP_DIR>/WEB-INF/
↳lib
cp postgis-jdbc-1.3.3.jar <GEOSERVER_WEBAPP_DIR>/WEB-INF/lib

restart geoserver
```

- #### 2. Either create a DB with the updated schema here https://github.com/geoserver/geofence/blob/master/doc/setup/sql/002_create_schema_postgres.sql or enable the hbm2ddl auto creation through the configuration file (see step 3)

Nota: Notice that «update» also creates the tables if they do not exist. In production, however, I would suggest to change it to «validate»

```
# If you want to create a new DB for GeoFence
sudo -u postgres createdb -O geonode geofence; \
sudo -u postgres psql -d geofence -c 'CREATE EXTENSION postgis;'; \
sudo -u postgres psql -d geofence -c 'GRANT ALL ON geometry_columns TO_
↳PUBLIC;'; \
sudo -u postgres psql -d geofence -c 'GRANT ALL ON spatial_ref_sys TO_
↳PUBLIC;'; \
sudo -u postgres psql -d geofence -c 'GRANT ALL PRIVILEGES ON ALL TABLES_
↳IN SCHEMA public TO geonode;'
```

3. Add configuration similar to geofence-datasource-ovr.properties sample below (if loaded as GeoServer extension)

<GEOSERVER_DATA_DIR>/geofence/geofence-datasource-ovr.properties

```
# /* (c) 2019 Open Source Geospatial Foundation - all rights reserved
# * This code is licensed under the GPL 2.0 license, available at the_
↳root
# * application directory.
# */
#
geofenceVendorAdapter.databasePlatform=org.hibernate.spatial.postgis.
↳PostgisDialect
geofenceDataSource.driverClassName=org.postgresql.Driver
geofenceDataSource.url=jdbc:postgresql://localhost:5432/geofence
geofenceDataSource.username=postgres
geofenceDataSource.password=postgres
geofenceEntityManagerFactory.jpaPropertyMap[hibernate.default_
↳schema]=public

#####
↳#####
## Other setup entries
#####
↳#####
## hbm2ddl.auto may assume one of these values:
## - validate: validates the DB schema at startup against the internal_
↳model. May fail on oracle spatial.
## - update: updates the schema, according to the internal model._
↳Updating automatically the production DB is dangerous.
## - create-drop: drop the existing schema and recreates it according to_
↳the internal model. REALLY DANGEROUS, YOU WILL LOSE YOUR DATA.
## You may want not to redefine the property entirely, in order to leave_
↳the default value (no action).

geofenceEntityManagerFactory.jpaPropertyMap[hibernate.hbm2ddl.
↳auto]=update
geofenceEntityManagerFactory.jpaPropertyMap[javax.persistence.validation.
↳mode]=none
geofenceEntityManagerFactory.jpaPropertyMap[hibernate.validator.apply_to_
↳ddl]=false
geofenceEntityManagerFactory.jpaPropertyMap[hibernate.validator.
↳autoregister_listeners]=false
```

(continues on next page)

(continuação da página anterior)

```
##
## ShowSQL is set to true in the configuration file; putting
↳ showsql=false in
## this file, you can easily check that this override file has been
↳ properly applied.

# geofenceVendorAdapter.generateDdl=false
# geofenceVendorAdapter.showSql=false

## Set to "true" in specific use cases
# workspaceConfigOpts.showDefaultGroups=false

#####
↳ #####
## Disable second level cache.
## This is needed in a geofence-clustered environment.

#geofenceEntityManagerFactory.jpaPropertyMap[hibernate.cache.use_second_
↳ level_cache]=false

#####
↳ #####
## Use external ehcache configuration file.
## Useful to change cache settings, for example diskStore path.
#geofenceEntityManagerFactory.jpaPropertyMap[hibernate.cache.provider_
↳ configuration_file_resource_path]=file:/path/to/geofence-ehcache-
↳ override.xml
```

1.21 Hardening GeoNode

1.21.1 Publish on HTTPS

TBD

1.21.2 OAuth2 Fixtures Update and Base URL Migration

TBD

1.21.3 GeoNode Security Subsystem

TBD

1.21.4 OAuth2 Tokens and Sessions

TBD (ref to *OAuth2 Access Tokens*)

1.22 Social Login

1.22.1 GeoNode Social Accounts

Contents

- *GeoNode Social Accounts*
 - *Allow GeoNode to Login throug Social Accounts (Facebook and Linkedin)*
 - * *Base concepts and objects*
 - * *Installation*
 - * *Configuration*
 - * *Usage*
 - *LinkedIn Application*
 - *Facebook Application*
 - * *Login by using Existing Accounts on GeoNode*

Allow GeoNode to Login throug Social Accounts (Facebook and Linkedin)

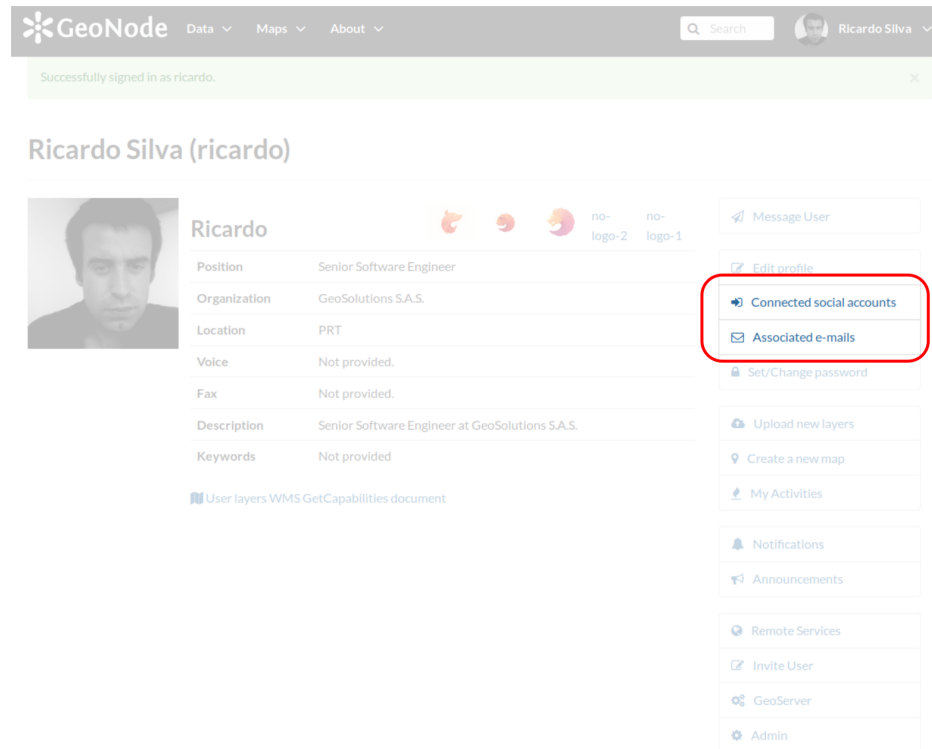
Base concepts and objects

In order to harmonize the various authentication flows between local accounts and remote social accounts, the whole user registration and authentication codebase has been refactored.

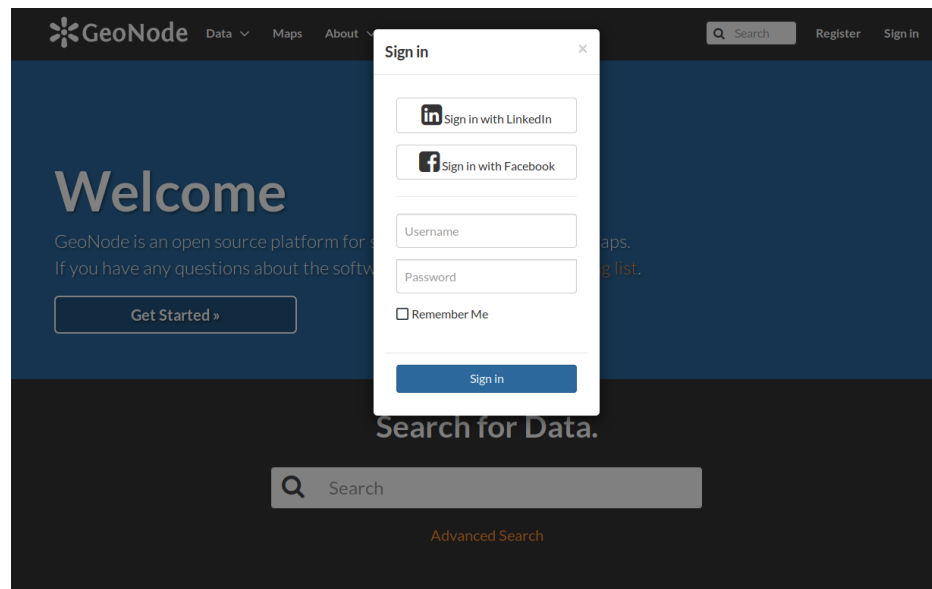
Major changes:

- `geonode-user-accounts` has been retired and is not used anymore. This app was only capable of managing local accounts;
- `django-allauth` has been integrated as a dependency of geonode. It provides a solution for managing both local and remote user accounts. It is now used for dealing with most user registration and auth flows;
- `django-invitations` has also been integrated as a dependency of geonode and is used for managing invitations to new users. This functionality was previously provided by `geonode-user-accounts`;
- `django-allauth` has been extended in order to provide the following additional features:
 - Automatically registering an e-mail with a user when the e-mail is used to connect to a social account;
 - Automatically extract information from the user's social account and use that to enhance the user's profile fields on geonode. This was implemented in a pluggable way, allowing custom installs to configure it for other providers;

- Allow approval of new registrations by staff members before allowing new users to login. This functionality was previously provided by geonode-user-accounts.
- There are now extra sections on the user's profile to manage connected social accounts and e-mail accounts



- When properly configured, the login and register pages now display the possibility to login with social accounts



Installation

- Install the new allauth plugin and remove any of the old dependencies

```
pip install -r requirements.txt --upgrade
pip install -e . --upgrade --no-cache
pip uninstall geonode-user-accounts -y
pip uninstall django-user-accounts -y
```

- ensure sure the Django model is updated and the templates updated to the static folder

```
DJANGO_SETTINGS_MODULE=geonode.local_settings python -W ignore manage.py_
↳makemigrations
DJANGO_SETTINGS_MODULE=geonode.local_settings python -W ignore manage.py_
↳migrate
DJANGO_SETTINGS_MODULE=geonode.local_settings python -W ignore manage.py_
↳collectstatic --noinput
```

- ensure that Social Providers are enabled in your settings:

```
# prevent signing up by default
ACCOUNT_OPEN_SIGNUP = True
ACCOUNT_EMAIL_REQUIRED = True
ACCOUNT_EMAIL_VERIFICATION = 'optional'
ACCOUNT_EMAIL_CONFIRMATION_EMAIL = True
ACCOUNT_EMAIL_CONFIRMATION_REQUIRED = True
ACCOUNT_CONFIRM_EMAIL_ON_GET = True
ACCOUNT_APPROVAL_REQUIRED = True

SOCIALACCOUNT_ADAPTER = 'geonode.people.adapters.SocialAccountAdapter'

SOCIALACCOUNT_AUTO_SIGNUP = False

INSTALLED_APPS += (
    'allauth.socialaccount.providers.linkedin_oauth2',
    'allauth.socialaccount.providers.facebook',
)

SOCIALACCOUNT_PROVIDERS = {
    'linkedin_oauth2': {
        'SCOPE': [
            'r_emailaddress',
            'r_basicprofile',
        ],
        'PROFILE_FIELDS': [
            'emailAddress',
            'firstName',
            'headline',
            'id',
            'industry',
            'lastName',
            'pictureUrl',
            'positions',
            'publicProfileUrl',
            'location',
            'specialties',
            'summary',
        ],
    },
}
```

(continues on next page)

(continuação da página anterior)

```

    ]
  },
  'facebook': {
    'METHOD': 'oauth2',
    'SCOPE': [
      'email',
      'public_profile',
    ],
    'FIELDS': [
      'id',
      'email',
      'name',
      'first_name',
      'last_name',
      'verified',
      'locale',
      'timezone',
      'link',
      'gender',
    ]
  },
}

# Comment out this in case you want to diable Social login
SOCIALACCOUNT_PROFILE_EXTRACTORS = {
    "facebook": "geonode.people.profileextractors.FacebookExtractor",
    "linkedin_oauth2": "geonode.people.profileextractors.
→LinkedInExtractor",
}

```

Configuration

1. Go to GeoNode/Django Admin Dashboard and add the Social Apps you want to configure:

admin/socialaccount/socialapp/

Django administration

Home > Social Accounts > Social applications

Select social application to change

Action: 0 of 2 selected

☐	Name	Provider
☐	Facebook	Facebook
☐	LinkedIn	LinkedIn

2 social applications

- LinkedIn

Change social application

Provider:	LinkedIn ▾
Name:	LinkedIn
Client id:	 App ID, or consumer key
Secret key:	 API secret, client secret, or consumer secret
Key:	 Key
Sites:	Available sites ⓘ Filter Choose all ⓘ Chosen sites ⓘ Remove all ⓘ Hold down "Control", or "Command" on a Mac, to select more than one.

- Facebook

Change social application

Provider:	Facebook ▾
Name:	Facebook
Client id:	 App ID, or consumer key
Secret key:	 API secret, client secret, or consumer secret
Key:	 Key
Sites:	Available sites ⓘ Filter Choose all ⓘ Chosen sites ⓘ Remove all ⓘ Hold down "Control", or "Command" on a Mac, to select more than one.

Aviso: Make sure to add the sites you want to enable.

Usage

You need first to create and configure OAuth2 Applications on your Social Providers.

This will require a personal or business account, which can access to the developers sections of LinkedIn and Facebook and create and configure new Applications.

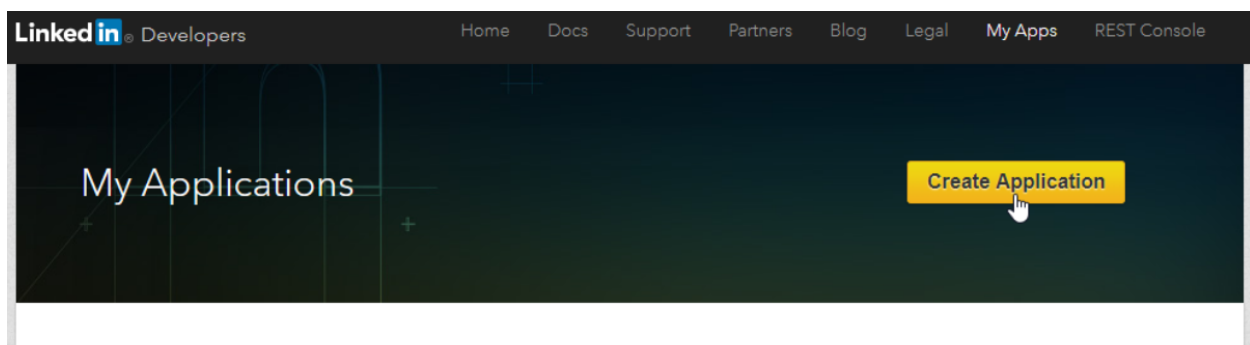
That account won't be visible to the GeoNode users. This is needed only to generate OAuth2 Client ID and Client Secret Authorization Keys.

In the following sections we will see in details how to configure them for both LinkedIn and Facebook.

LinkedIn Application

(ref.: <http://django-allauth.readthedocs.io/en/latest/providers.html>)

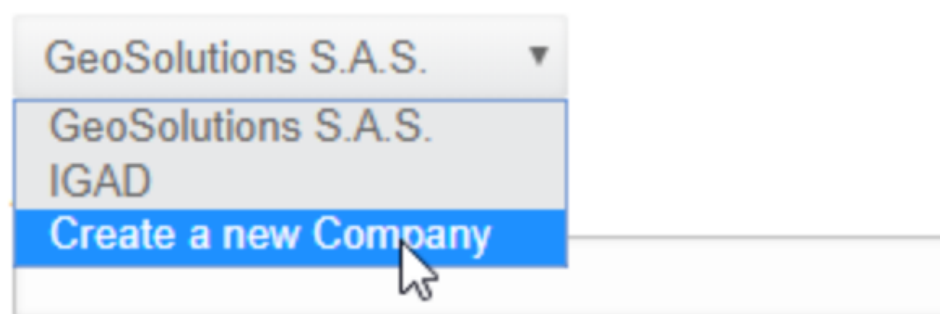
1. Go to <https://www.linkedin.com/developer/apps> and select Create Application



2. Create a new Company

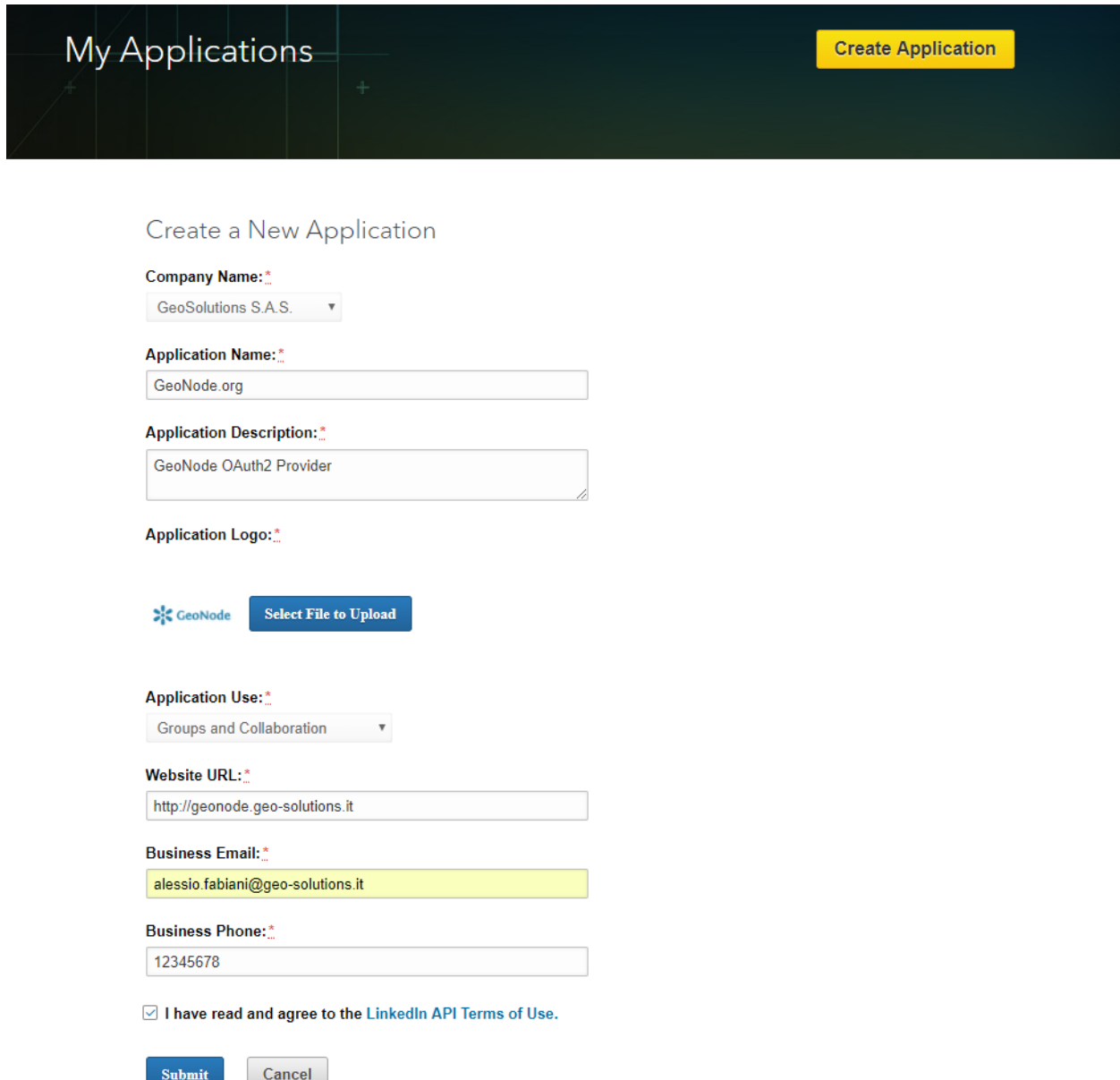
Create a New Application

Company Name: *



3. Fill the informations

Nota: The logo must have precise square dimensions



The screenshot shows the 'My Applications' section of the GeoNode interface. At the top, there's a dark header with the text 'My Applications' and a yellow 'Create Application' button. Below this, the 'Create a New Application' form is displayed. The form includes several fields: 'Company Name' (a dropdown menu showing 'GeoSolutions S.A.S.'), 'Application Name' (a text input field with 'GeoNode.org'), 'Application Description' (a text input field with 'GeoNode OAuth2 Provider'), 'Application Logo' (a section with the GeoNode logo and a 'Select File to Upload' button), 'Application Use' (a dropdown menu showing 'Groups and Collaboration'), 'Website URL' (a text input field with 'http://geonode.geo-solutions.it'), 'Business Email' (a text input field with 'alessio.fabiani@geo-solutions.it'), and 'Business Phone' (a text input field with '12345678'). At the bottom of the form, there is a checkbox labeled 'I have read and agree to the [LinkedIn API Terms of Use](#)' which is checked. Finally, there are 'Submit' and 'Cancel' buttons.

4. Select the following Default Application Permissions

Aviso: Be sure to select the `r_basicprofile` and `r_emailaddress` application permissions.

Default Application Permissions

☒ r_basicprofile
☐ w_share

☒ r_emailaddress

5. Add OAuth 2.0 Authorized Redirect URLs:

```
http://geonode.geo-solutions.it/account/linkedin_oauth2/login/callback/  
http://geonode.geo-solutions.it/account/linkedin/login/callback/
```

Default Application Permissions

☒ r_basicprofile
☐ w_share

☒ r_emailaddress

☐ rw_c

OAuth 2.0

Authorized Redirect URLs:

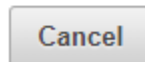
6. Save

<http://geonode.geo-solutions.it/account/linkedin/login>

OAuth 1.0a

Default "Accept" Redirect URL:

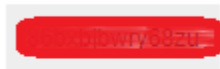
Default "Cancel" Redirect URL:



7. Take note of the Authentication Keys

Authentication Keys

Client ID:

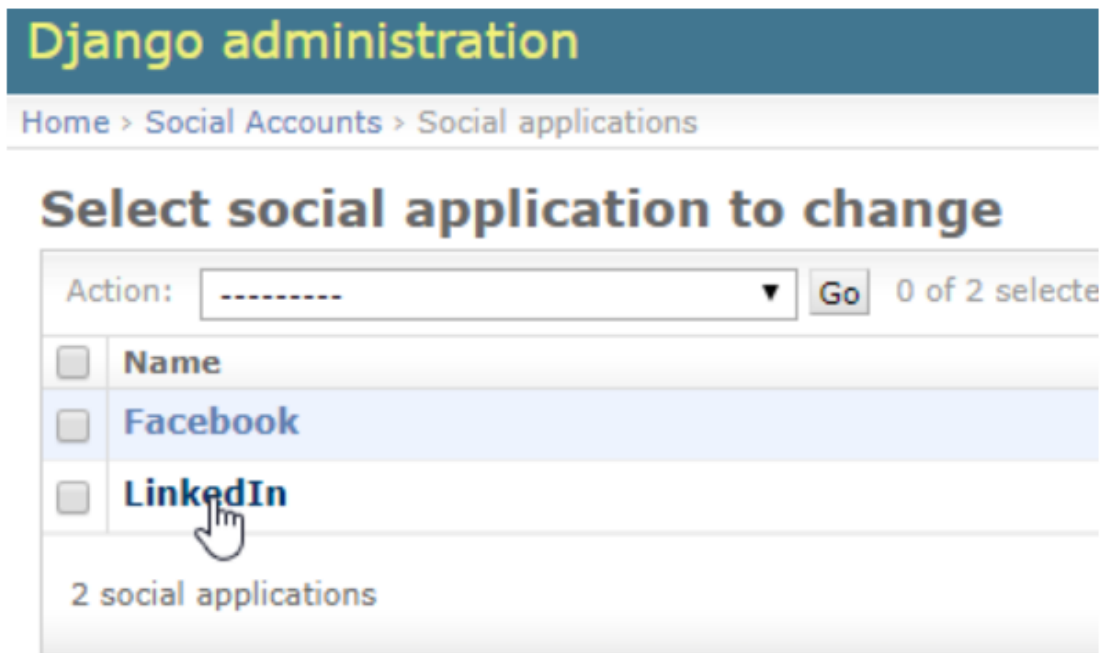


Client Secret:



Default Application Permissions

8. Go to GeoNode/Django admin, Social Applications and select the LinkedIn one
(/admin/socialaccount/socialapp/)



Django administration

Home > Social Accounts > Social applications

Select social application to change

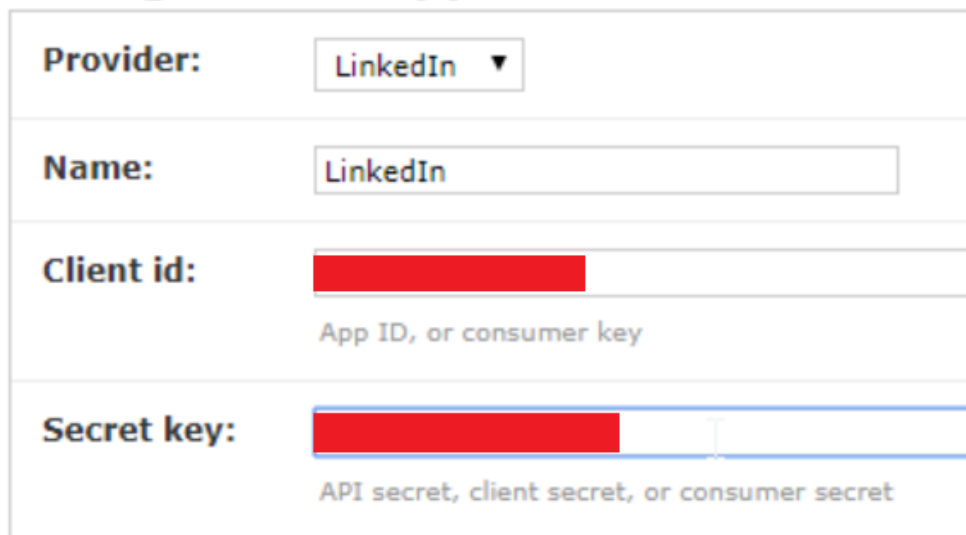
Action: 0 of 2 selected

☐	Name
☐	Facebook
☐	LinkedIn

2 social applications

9. Cut and Paste the Client ID and Client Secret on the related fields

Change social application



Provider:

Name:

Client id:

App ID, or consumer key

Secret key:

API secret, client secret, or consumer secret

10. Save

Facebook Application

(ref.: <http://django-allauth.readthedocs.io/en/latest/providers.html>) (ref.: <https://www.webforefront.com/django/setupdjangosocialauthentication.html>)

1. Go to <https://developers.facebook.com/apps> and Add a New Application



2. Create the App ID and go to the Dashboard

Crea un nuovo ID app

Inizia l'integrazione di Facebook nella tua app o nel sito Web

Nome visualizzato

Indirizzo e-mail di contatto

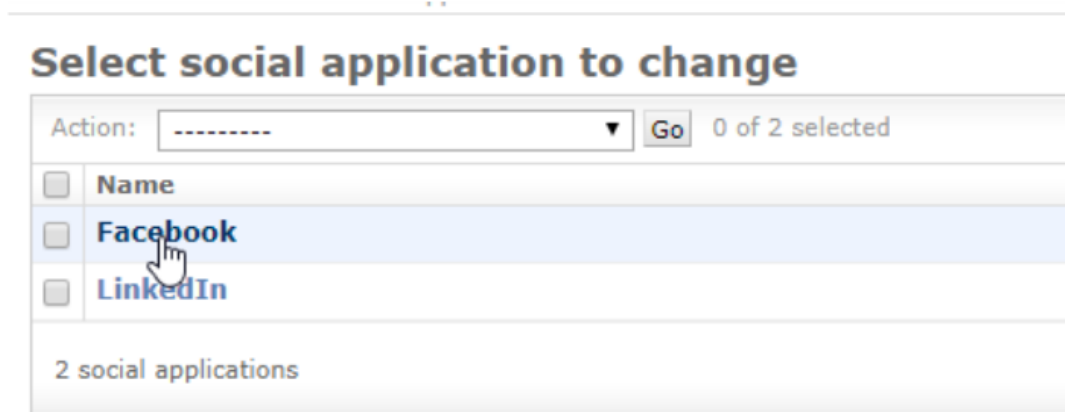
Procedendo, accetti le [Normative della Piattaforma Facebook](#)



3. Take note of the Authentication Keys



4. Go to GeoNode/Django admin, Social Applications and select the LinkedIn one
(/admin/socialaccount/socialapp/)



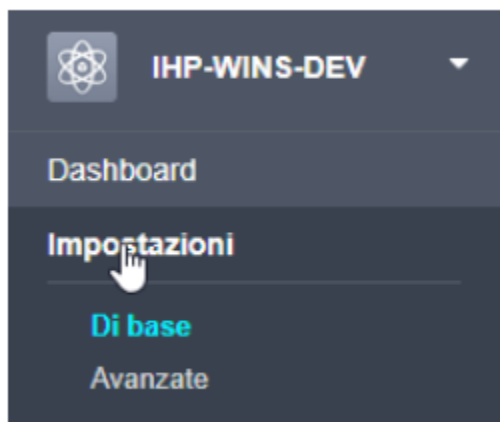
5. Cut and Paste the App ID and Secret Key on the related fields

ClientID	<-->	App Id
Client Secret	<-->	Secret Key

Change social application

Provider:	Facebook ▼
Name:	Facebook
Client id:	 App ID, or consumer key
Secret key:	 API secret, client secret, or consumer secret

6. Save
7. Go back to the Facebook Application Dashboard and select Settings



8. Add your App Domain

ID app

Nome visualizzato

Domini app

<http://igad-dev.geo-solutions.it/>

Normativa sulla privacy per la finestra di dialogo di accesso e per

9. Click on Add Platform

Domini app

Indirizzo e-mail

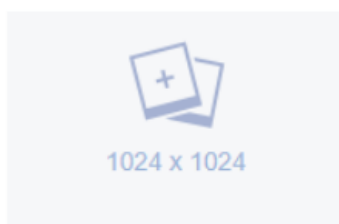
alessio.fabiani

URL Normativa sulla privacy

URL delle Cond

Condizioni d'u

Icona dell'app (1024 x 1024)



Categoria

Scegli una cate

Ottieni maggiori

+ Aggiungi piattaforma



10. Select Web Site

Seleziona piattaforma



Giochi Web di
Facebook



Sito Web



iO



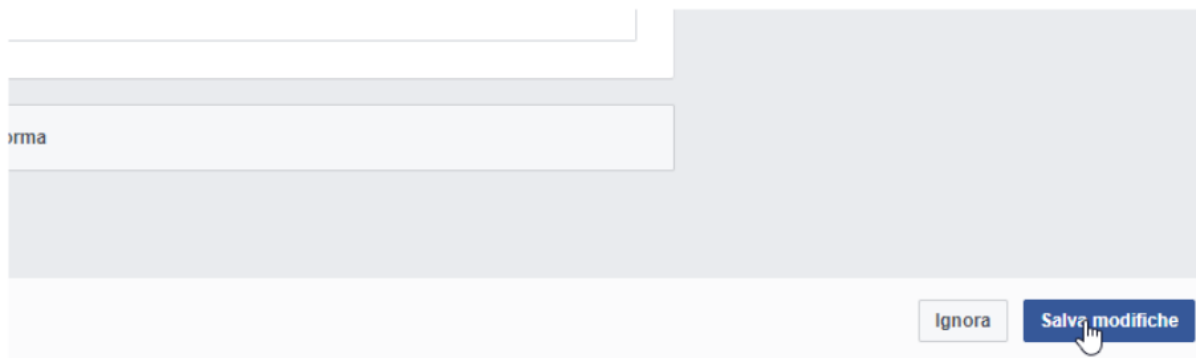
11. Add the URL

Sito Web

URL del sito

http://igad-dev.geo-solutions.it/

12. And Save



13. Go to Add Product



14. Select Facebook Login

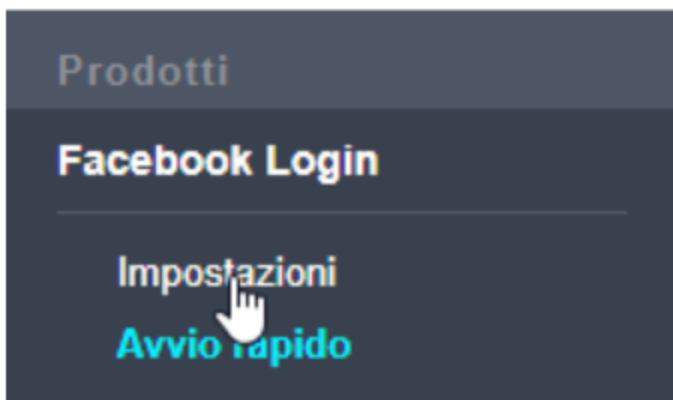


15. Select Web

questa app.



16. Go to Settings



17. Make sure Allow client OAuth and Access via OAuth Web are enabled

☒ Sì	Accesso client OAuth Abilita il flow standard del token client OAuth. Proteggi la tua applicazione con il controllo del browser per l'accesso al client. [?]
☒ Sì	Accesso a OAuth Web Abilita l'accesso al client OAuth basato sul web. [?]
☐ No	Accesso OAuth al browser incorporato Abilita l'URI di reindirizzamento per il controllo del browser per l'accesso al client. [?]

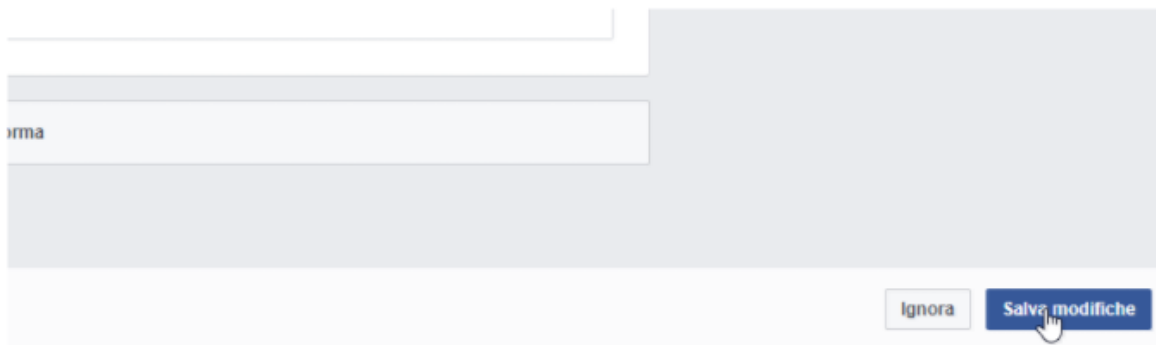
18. Add the valid redirect URIs:

```
http://geonode.geo-solutions.it/account/facebook/login/callback/
http://geondoe.geo-solutions.it/account/login/
```

URI di reindirizzamento OAuth validi

☐ No	Accesso dai dispositivi Abilita il flusso di accesso del client OAuth per dispositivi come smart TV [?]
------------------------------------	---

19. Save

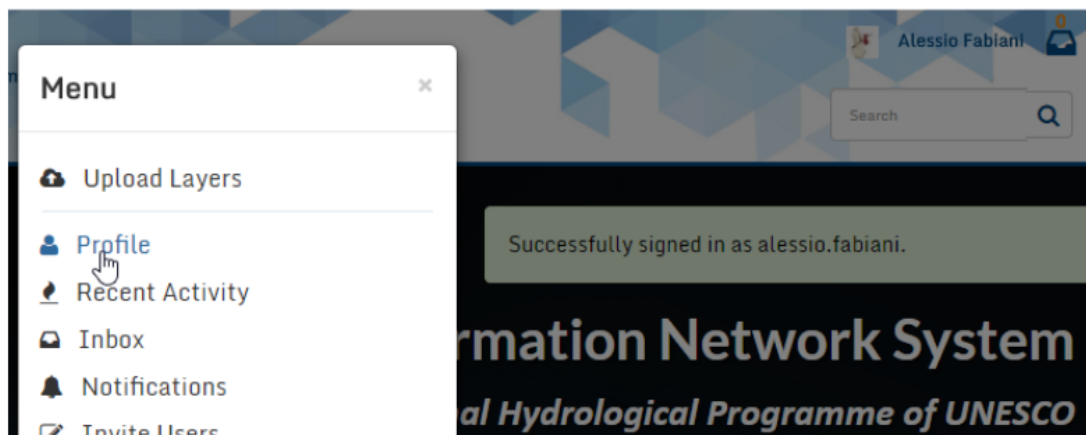


Login by using Existing Accounts on GeoNode

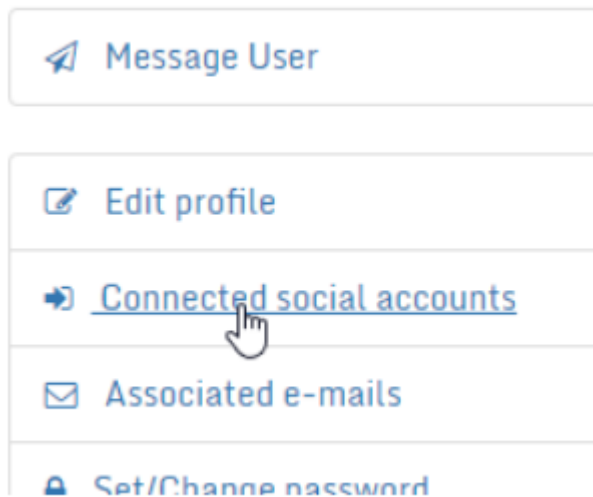
If you want to enable an already existing user account to login through social apps, you need to associate it to social accounts.

Usually this could be done only by the current user, since this operation requires authentication on its social accounts.

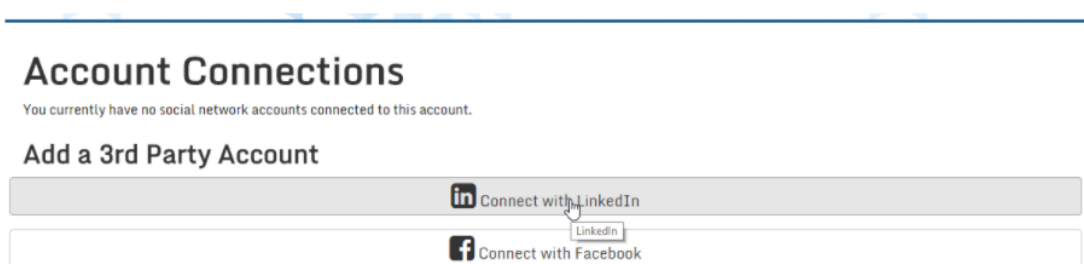
In order to do that you need to go to the User Profile Settings



Click on “Connected social accounts”



And actually connect them



Account Connections

You can sign in to your account using any of the following already connected third party accounts:

- ☐  Facebook account: Alessio Fabiani
- ☐  LinkedIn account: Alessio Fabiani

[Remove](#)

1.23 GeoNode Django Contrib Apps

1.23.1 Geonode auth via LDAP

This package provides utilities for using LDAP as an authentication and authorization backend for geonode.

The `django_auth_ldap` package is a very capable way to add LDAP integration with django projects. It provides a lot of flexibility in mapping LDAP users to geonode users and is able to manage user authentication.

However, in order to provide full support for mapping LDAP groups with geonode's and enforce group permissions on resources, a custom geonode authentication backend is required. This contrib package provides such a backend, based on `django_auth_ldap`.

Installation

Installing this contrib package is a matter of:

1. Installing geonode
2. Installing system LDAP libraries (development packages needed)
3. Cloning this repository locally
4. Change to the `ldap` directory and install this contrib package

```
# 1. install geonode (not shown here for brevity)
# 2. install systemwide LDAP libraries
sudo apt install \
    libldap2-dev \
    libsasl2-dev

# 3. get geonode/contribs code
git clone https://github.com/GeoNode/geonode-contribs.git

# 4. install geonode ldap contrib package
cd geonode-contribs/ldap
pip install .
```

Configuration

1. Add `geonode_ldap.backend.GeonodeLdapBackend` as an additional auth backend.

```
# e.g. by updating your settings.py or local_settings.py
AUTHENTICATION_BACKENDS += (
    "geonode_ldap.backend.GeonodeLdapBackend",
)
```

You may use additional auth backends, the django authentication framework tries them all according to the order listed in the settings. This means that geonode can be setup in such a way as to permit internal organization users to login with their LDAP credentials, while at the same time allowing for casual users to use their facebook login (as long as you enable facebook social auth provider).

Nota: The django's `django.contrib.auth.backends.ModelBackend` must also be used in order to provide full geonode integration with LDAP. However this is included by default on GeoNode settings

```
# The GeoNode default settings are the following
AUTHENTICATION_BACKENDS = (
    'oauth2_provider.backends.OAuth2Backend',
    'django.contrib.auth.backends.ModelBackend',
    'guardian.backends.ObjectPermissionBackend',
    'allauth.account.auth_backends.AuthenticationBackend',
)
```

2. Set some additional configuration values. Some of these variables are prefixed with `AUTH_LDAP` (these are used directly by `django_auth_ldap`) while others are prefixed with `GEONODE_LDAP` (these are used by `geonode_ldap`). The geonode custom variables are:

- `GEONODE_LDAP_GROUP_PROFILE_FILTERSTR` - This is an LDAP search fragment with the filter that allows querying for existing groups. See example below
- `GEONODE_LDAP_GROUP_NAME_ATTRIBUTE` - This is the name of the LDAP attribute that will be used for deriving the geonode group name. If not specified it will default to `cn`, which means that the LDAP object's *common name* will be used for generating the name of the geonode group
- `GEONODE_LDAP_GROUP_PROFILE_MEMBER_ATTR` - This is the name of the LDAP attribute that will be used for deriving the geonode membership. If not specified it will default to `member`

Example configuration:

```
# add these import lines to the top of your geonode settings file
from django_auth_ldap import config as ldap_config
from geonode_ldap.config import GeonodeNestedGroupOfNamesType
import ldap

# enable logging
import logging
logger = logging.getLogger('django_auth_ldap')
logger.addHandler(logging.StreamHandler())
logger.setLevel(logging.DEBUG)

# add both standard ModelBackend auth and geonode.contrib.ldap auth
AUTHENTICATION_BACKENDS += (
    'geonode_ldap.backend.GeonodeLdapBackend',
)

# django_auth_ldap configuration
AUTH_LDAP_SERVER_URI = os.getenv("LDAP_SERVER_URL")
AUTH_LDAP_BIND_DN = os.getenv("LDAP_BIND_DN")
AUTH_LDAP_BIND_PASSWORD = os.getenv("LDAP_BIND_PASSWORD")
AUTH_LDAP_USER_SEARCH = ldap_config.LDAPSearch(
    os.getenv("LDAP_USER_SEARCH_DN"),
    ldap.SCOPE_SUBTREE,
    os.getenv("LDAP_USER_SEARCH_FILTERSTR")
)

# should LDAP groups be used to spawn groups in GeoNode?
AUTH_LDAP_MIRROR_GROUPS = strtobool(os.getenv("LDAP_MIRROR_GROUPS", 'True'))
AUTH_LDAP_GROUP_SEARCH = ldap_config.LDAPSearch(
    os.getenv("LDAP_GROUP_SEARCH_DN"),
    ldap.SCOPE_SUBTREE,
    os.getenv("LDAP_GROUP_SEARCH_FILTERSTR")
)
AUTH_LDAP_GROUP_TYPE = GeonodeNestedGroupOfNamesType()
```

(continues on next page)

(continuação da página anterior)

```

AUTH_LDAP_USER_ATTR_MAP = {
    "first_name": "givenName",
    "last_name": "sn",
    "email": "mailPrimaryAddress"
}
AUTH_LDAP_FIND_GROUP_PERMS = True
AUTH_LDAP_MIRROR_GROUPS_EXCEPT = [
    "test_group"
]

# these are not needed by django_auth_ldap - we use them to find and match
# GroupProfiles and GroupCategories
GEONODE_LDAP_GROUP_NAME_ATTRIBUTE = os.getenv("LDAP_GROUP_NAME_ATTRIBUTE", default="cn
↪")
GEONODE_LDAP_GROUP_PROFILE_FILTERSTR = os.getenv("LDAP_GROUP_SEARCH_FILTERSTR", ↪
↪default='(ou=research group)')
GEONODE_LDAP_GROUP_PROFILE_MEMBER_ATTR = os.getenv("LDAP_GROUP_PROFILE_MEMBER_ATTR", ↪
↪default='member')

```

Example environment variables:

```

LDAP_SERVER_URL=ldap://<the_ldap_server>
LDAP_BIND_DN=uid=ldapinfo,cn=users,dc=ad,dc=example,dc=org
LDAP_BIND_PASSWORD=<something_secret>
LDAP_USER_SEARCH_DN=dc=ad,dc=example,dc=org
LDAP_USER_SEARCH_FILTERSTR=( & (uid=%(user)s) (objectClass=person) )
LDAP_MIRROR_GROUPS=True
LDAP_GROUP_SEARCH_DN=cn=groups,dc=ad,dc=example,dc=org
LDAP_GROUP_SEARCH_FILTERSTR=( | (cn=abt1) (cn=abt2) (cn=abt3) (cn=abt4) (cn=abt5) (cn=abt6) )
LDAP_GROUP_PROFILE_MEMBER_ATTR=uniqueMember

```

The configuration seen in the example above will allow LDAP users to login to geonode with their LDAP credentials.

On first login, a geonode user is created from the LDAP user and its LDAP attributes `cn` and `sn` are used to populate the geonode user's `first_name` and `last_name` profile fields.

Any groups that the user is a member of in LDAP (under the `cn=groups,dc=ad,dc=example,dc=org` search base and belonging to one of `( | (cn=abt1) (cn=abt2) (cn=abt3) (cn=abt4) (cn=abt5) (cn=abt6) )` groups) will be mapped to the corresponding geonode groups, even creating these groups in geonode in case they do not exist yet. The geonode user is also made a member of these geonode groups.

Upon each login, the user's geonode group memberships are re-evaluated according to the information extracted from LDAP. The `AUTH_LDAP_MIRROR_GROUPS_EXCEPT` setting can be used to specify groups whose memberships will not be re-evaluated.

If no LDAP groups shall be mirrored `LDAP_MIRROR_GROUPS` and `LDAP_MIRROR_GROUPS_EXCEPT` must be set to `False`.

Nota: Users mapped from LDAP will be marked with an `ldap` tag. This will be used to keep them in sync.

Aviso: If you remove the `ldap` tag, the users will be threaten as pure internal GeoNode ones.

You may also manually generate the geonode groups in advance, before users login. In this case, when a user logs in and the mapped LDAP group already exists, the user is merely added to the geonode group

Be sure to check out `django_auth_ldap` for more information on the various configuration options.

Keep Users and Groups Synchronized

In order to constantly keep the remote LDAP Users and Groups **synchronized** with GeoNode, you will need to run periodically some specific management commands.

```
* /10 * * * * /opt/geonode/my-geonode/manage.sh updateldapgroups >> /var/log/cron.log_
↪ 2>&1
* /10 * * * * /opt/geonode/my-geonode/manage.sh updateldapusers >> /var/log/cron.log_
↪ 2>&1
```

Where the `manage.sh` is a bash script similar to the following one:

`manage.sh`

```
export $(grep -v '^#' /opt/geonode/my-geonode/.env | xargs -d '\n'); /home/<my_user>/.
↪ virtualenvs/geonode/bin/python /opt/geonode/my-geonode/manage.py $@
```

and the `/opt/geonode/my-geonode/.env` is something similar to the following one:

`/opt/geonode/my-geonode/.env`

```
DEBUG=False
DJANGO_ALLOWED_HOSTS=<geonode_public_host>,localhost,127.0.0.1
DJANGO_DATABASE_URL=postgis://my_geonode:*****@localhost:5432/my_geonode_db
DEFAULT_BACKEND_UPLOADER=geonode.importer
DEFAULT_FROM_EMAIL=geonode@example.org
DJANGO_EMAIL_HOST=smtp.example.org
DJANGO_EMAIL_HOST_PASSWORD=*****
DJANGO_EMAIL_HOST_USER=geonode
DJANGO_EMAIL_PORT=465
DJANGO_EMAIL_USE_SSL=True
DJANGO_SETTINGS_MODULE=my_geonode.settings
DJANGO_SECRET_KEY=*****
OAUTH2_API_KEY=*****
PROXY_URL=/proxy/?url=
EXIF_ENABLED=True
EMAIL_ENABLE=True
TIME_ENABLED=True
ACCOUNT_OPEN_SIGNUP=True
ACCOUNT_APPROVAL_REQUIRED=True
ACCOUNT_EMAIL_REQUIRED=True
ACCOUNT_EMAIL_VERIFICATION=optional
AVATAR_GRAVATAR_SSL=True
GEONODE_DB_URL=postgis://my_geonode:*****@localhost:5432/my_geonode_data
GEOSERVER_ADMIN_PASSWORD=*****
GEOSERVER_LOCATION=https://<geonode_public_host>/geoserver/
GEOSERVER_PUBLIC_HOST=<geonode_public_host>
GEOSERVER_PUBLIC_LOCATION=https://<geonode_public_host>/geoserver/
GEOSERVER_WEB_UI_LOCATION=https://<geonode_public_host>/geoserver/
LDAP_SERVER_URL=ldap://<the_ldap_server>
LDAP_BIND_DN=uid=ldapinfo,cn=users,dc=ad,dc=example,dc=org
LDAP_BIND_PASSWORD=<something_secret>
LDAP_USER_SEARCH_DN=dc=ad,dc=example,dc=org
LDAP_USER_SEARCH_FILTERSTR=( & (uid=%(user)s) (objectClass=person) )
LDAP_MIRROR_GROUPS=True
```

(continues on next page)

(continuação da página anterior)

```

LDAP_GROUP_SEARCH_DN=cn=groups,dc=ad,dc=example,dc=org
LDAP_GROUP_SEARCH_FILTERSTR=(|(cn=abt1)(cn=abt2)(cn=abt3)(cn=abt4)(cn=abt5)(cn=abt6))
LDAP_GROUP_PROFILE_MEMBER_ATTR=uniqueMember
OGC_REQUEST_MAX_RETRIES=3
OGC_REQUEST_POOL_CONNECTIONS=100
OGC_REQUEST_POOL_MAXSIZE=100
OGC_REQUEST_TIMEOUT=60
SITEURL=https://<geonode_public_host>/
SITE_HOST_NAME=<geonode_public_host>
FREETEXT_KEYWORDS_READONLY=False
# Advanced Workflow Settings
ADMIN_MODERATE_UPLOADS=False
GROUP_MANDATORY_RESOURCES=False
GROUP_PRIVATE_RESOURCES=False
RESOURCE_PUBLISHING=False

```

Nota: You might want to use the same `/opt/geonode/my-geonode/.env` for your UWSGI configuration too:

```

[uwsgi]
socket = 0.0.0.0:8000
uid = <my_user>
gid = www-data

plugins = python3
virtualenv = /home/<my_user>/virtualenvs/geonode

# set environment variables from .env file
env LANG=en_US.utf8
env LC_ALL=en_US.UTF-8
env LC_LANG=en_US.UTF-8

for-readline = /opt/geonode/my-geonode/.env
    env = %(_)
endfor =

chdir = /opt/geonode/my-geonode
module = my_geonode.wsgi:application

processes = 12
threads = 2
enable-threads = true
master = true

# logging
# path to where uwsgi logs will be saved
logto = /storage/my_geonode/logs/geonode.log
daemonize = /storage/my_geonode/logs/geonode.log
touch-reload = /opt/geonode/my-geonode/my_geonode/wsgi.py
buffer-size = 32768
max-requests = 500
harakiri = 300 # respawn processes taking more than 5 minutes (300 seconds)
# limit-as = 1024 # avoid Errno 12 cannot allocate memory
harakiri-verbose = true
vacuum = true
thunder-lock = true

```

1.23.2 Geonode Logstash for centralized monitoring/analytics

This contrib app, along with the GeoNode internal monitoring app, lets administrators to configure a service for sending metrics data to a **centralized server** which comes with [Logstash](#).

So it will be possible to visualize stats and charts about one or more GeoNode instances outside the application. Having a server configured with the [ELK stack](#), it is possible to visualize those information on a Kibana dashboard for example.

If you manage more than one GeoNode instances, that server can receive data from many GeoNode(s) so it can make available both *single-instance dashboards* (referred to individual instances) and *global dashboards* (stats calculated on the whole set of instances).

Aviso: The centralized monitoring service cannot be active if the settings variables [USER_ANALYTICS_ENABLED](#) and `monitoring-enabled` are set to *False*.

Overview

By default, GeoNode will send data to the centralized server every **3600 seconds** (1 hour) so, if enabled, the monitoring app will collect 1-hour-aggregated data. This time interval can be configured, see the next paragraphs to know how.

Formatted and compressed data will be sent on a **TCP** connection (on the **443** standard port by default) through a **scheduled celery task** which basically logs information via [python-logstash-async](#).

Aviso: This feature requires [python-logstash-async](#).

Data and events formats

Each time the centralized monitoring service is called, 4 types of *JSON* formatted events are sent to the server:

1. Instance overview

```
{
  "format_version": "1.0",
  "instance": {
    "name": geonode instance HOSTNAME,
    "ip": geonode instance IP
  },
  "time": {
    "startTime": UTC now - 1 hour (default)
    "endTime": UTC now
  },
  "hits": total number of requests,
  "unique_visits": total number of unique sessions,
  "unique_visitors": total number of unique users,
  "registered_users": total number of registered users at the end time,
  "layers": total number of layers at the end time,
  "documents": total number of documents at the end time,
  "maps": total number of maps at the end time,
  "errors": total number of errors
}
```

2. Resources details


```
{
  "format_version": "1.0",
  "instance": {
    "name": geonode instance HOSTNAME,
    "ip": geonode instance IP
  },
  "time": {
    "startTime": UTC now - 1 hour (default)
    "endTime": UTC now
  },
  "resources": [
    ...
    {
      "type": resource type,
      "name": resource name,
      "url": resource URL,
      "hits": total number of requests about this resource,
      "unique_visits": total number of unique sessions about this resource,
      "unique_visitors": total number of unique users about this resource,
      "downloads": total number of resource downloads,
      "ogcHits": total number of OGC service requests about this resource,
      "publications": total number of publication events
    },
    ...
  ]
}
```

3. Countries details

```
{
  "format_version": "1.0",
  "instance": {
    "name": geonode instance HOSTNAME,
    "ip": geonode instance IP
  },
  "time": {
    "startTime": UTC now - 1 hour (default)
    "endTime": UTC now
  },
  "countries": [
    ...
    {
      "name": country name,
      "hits": total number of requests about the country
    },
    ...
  ]
}
```

4. UA (User Agent) Family details

```
{
  "format_version": "1.0",
  "instance": {
    "name": geonode instance HOSTNAME,
    "ip": geonode instance IP
  },
  ...
}
```

(continues on next page)

(continuação da página anterior)

```
"time": {
  "startTime": UTC now - 1 day
  "endTime": UTC now
},
"ua_families": [
  ...
  {
    "name": UA family name
    "hits": total number of requests about the UA family
  },
  ...
]
```

These messages will be [gzip](#) compressed in order to improve transport performances and they should be parsed by a [logstash filter](#) on the server side (see [Logstash configuration](#)).

Configuration

The centralized monitoring service is disabled by default because it needs the internal monitoring to be active and service-specific configurations.

GeoNode configuration

On the GeoNode side, all needed configurations can be set up from the Django admin interface. If enabled, the **GEONODE LOGSTASH** section will show the **Centralized servers** feature:

GEONODE_LOGSTASH

Centralized servers

 [Change](#)

Let's add one:

Django administration
WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home · Geonode_Logstash · Centralized servers · CentralizedServer object

Change centralized server
HISTORY

Host: 192.168.1.91
Centralized Server IP address.

Port: 5000
Centralized Server TCP port number.

Interval: 3600
Data aggregation time interval (in seconds).

Db path: logstash.db
The local SQLite database to cache log events between emitting and transmission to the Logstash server. This way log events are cached even across process restarts (and crashes).

Socket timeout: 5
Timeout in seconds for TCP connections.

Queue check interval: 2
Interval in seconds to check the internal queue for new messages to be cached in the database.

Queue events flush interval: 0.1
Interval in seconds to send cached events from the database to Logstash.

Queue events flush count: 50
Count of cached events to send from the database to Logstash; events are sent to Logstash whenever QUEUED_EVENTS_FLUSH_COUNT or QUEUED_EVENTS_FLUSH_INTERVAL is reached, whatever happens first.

Queue events batch size: 50
Maximum number of events to be sent to Logstash in one batch. Depending on the transport, this usually means a new connection to the Logstash is established for the event batch.

Logstash db timeout: 5
Timeout in seconds to 'connect' the SQLite database.

Last successful deliver: Oct 17, 2019, 2:39 p.m.
Timestamp of the last successful deliver.

Next scheduled deliver: Oct 17, 2019, 3:39 p.m.
Timestamp of the next scheduled deliver.

Last failed deliver: -
Timestamp of the last failed deliver.

Delete Save and continue editing SAVE

Test connection

The **Host** IP address and the **Port** number are mandatory as well as the time **Interval** (3600 seconds by default) which defines the service invocation polling (so the time range on which data should be aggregated).

Nota: Once the service configured, the user can test the configuration by clicking on **Test connection**. It will test the connection with the centralized server without saving the configuration.

Other settings come with a default value:

- **Db path** → the local Spatialite database to cache events between emitting and transmission to the Logstash server (log events are cached even across process restarts and crashes);
- **Socket timeout** → timeout in seconds for TCP connections;
- **Queue check interval** → interval in seconds to check the internal queue for new messages to be cached in the database;
- **Queue events flush interval** → interval in seconds to send cached events from the database to Logstash;
- **Queue events flush count** → count of cached events to send from the database to Logstash;

- **Queue events batch size** -> maximum number of events to be sent to Logstash in one batch;
- **Logstash db timeout** -> timeout in seconds to “connect” the Spatialite database.

To better understand what these variables mean, it is recommended to read the [python-logstash-async options for the asynchronous processing and formatting](#).

Other three read-only fields will be visible:

- **Last successful deliver** -> timestamp of the last successful deliver (if exists);
- **Next scheduled deliver** -> timestamp of the next scheduled deliver;
- **Last failed deliver** -> timestamp of the last failed deliver (if exists).

Logstash configuration

On the server side, a proper Logstash configuration should be set up.

Some events formats contain arrays (see [Data and events formats](#)) so Logstash should be able to retrieve a single event for each element of the array. The [Split filter plugin](#) helps to correctly parse those messages.

As mentioned above, events messages will be gzip compressed so the [Gzip_lines codec plugin](#) should be installed along with Logstash and the «gzip_lines» codec should be used for the *tcp* input.

An example of the logstash configuration:

```
input {
  tcp {
    port => <logstash_port_number>
    codec => "gzip_lines"
  }
}

filter {
  json {
    source => "message"
  }
  if [format_version] == "1.0" {
    if [countries] {
      split {
        field => "countries"
      }
    }
    if [resources] {
      split {
        field => "resources"
      }
    }
    if [ua_families] {
      split {
        field => "ua_families"
      }
    }
    mutate {
      remove_field => "message"
    }
  }
}

geoip {
```

(continues on next page)

(continuação da página anterior)

```

    source => "[instance][ip]"
  }
}

output {
  elasticsearch {
    hosts => "elasticsearch:<elastic_port_number>"
    index => "logstash-%{[instance][name]}-%{+YYYY.MM.dd}"
    user => "elastic"
    password => "changeme"
  }
  stdout { codec => rubydebug }
}

```

Usage

When saving the service configuration, if monitoring enabled, GeoNode will create/update a celery [Periodic Task](#) which will be executed at regular intervals based on the *interval* configured.

You can check this behavior on the *Periodic Tasks* section of the admin UI:

PERIODIC TASKS	
Crontabs	+ Add ✎ Change
Intervals	+ Add ✎ Change
Periodic tasks	+ Add ✎ Change
Solar events	+ Add ✎ Change

The *dispatch-metrics-task* task:

Django administration
WELCOME, ADMIN. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home · Periodic Tasks · Periodic tasks

Select periodic task to change
ADD PERIODIC TASK +

Action: Go 0 of 3 selected

☐	PERIODIC TASK	ENABLED
☐	dispatch-metrics-task: every 3600 seconds	✓
☐	delayed-security-sync-task: every 60 seconds	✓
☐	celery.backend_cleanup: 0 4 * * * (m/h/d/dM/MY)	✓

3 periodic tasks

The task details:

Django administration
WELCOME, ADMIN . VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Periodic Tasks › Periodic tasks › dispatch-metrics-task: every 3600 seconds

Change periodic task
HISTORY

Name: dispatch-metrics-task
Useful description

Task (registered):

Task (custom): toring.tasks.dispatch_metrics

☒ Enabled

Schedule

Interval: every 3600 seconds

Crontab: -----
Use one of interval/crontab

Solar: -----
Use a solar schedule

Arguments (Show)

Execution Options (Show)

Delete Save and add another Save and continue editing SAVE

Aviso: When disabling monitoring is a **good practice** to disable the corresponding Periodic Task too.

Management command

In addition to the scheduled task, this contrib app makes also available the **dispatch_metrics** command to manually send metrics to the server.

Obviously the time interval considered will start at the last successful delivery and will finish at the current time.

When the monitoring plugin is enabled (*USER_ANALYTICS_ENABLED* and *monitoring-enabled* are set to *True*) and a *Geonode Logstash for centralized monitoring/analytics* configured, Geonode sends (hourly by default) metrics data to an external server (which comes with Logstash) for stats visualization and analysis.

The command can be launched using the `manage.py` script. No options are required.

```
$ DJANGO_SETTINGS_MODULE=<your_settings_module> python manage.py dispatch_metrics
```

Possible exceptions raised during the execution will be reported to GeoNode log.

1.24 GeoNode Admins Guide

GeoNode has an administration panel, based on the Django admin, which can be used to do some database operations. Although most of the operations can and should be done through the normal GeoNode interface, the admin panel provides a quick overview and management tool over the database.

The following sections will explain more in depth what functionalities the admin panel makes you available. It should be highlighted that the sections not covered in this guide are meant to be managed through GeoNode UI.

1.24.1 Accessing the panel

The *Admin Panel* is a model-centric interface where trusted users can manage content on GeoNode. Only the staff users can access the admin interface.

Nota: The “staff” flag, which controls whether the user is allowed to log in to the admin interface, can be set by the admin panel itself.

The panel can be reached from *Admin* link of the *User Menu* in the navigation bar (see the picture below) or through this URL: `http://<your_geonode_host>/admin`.

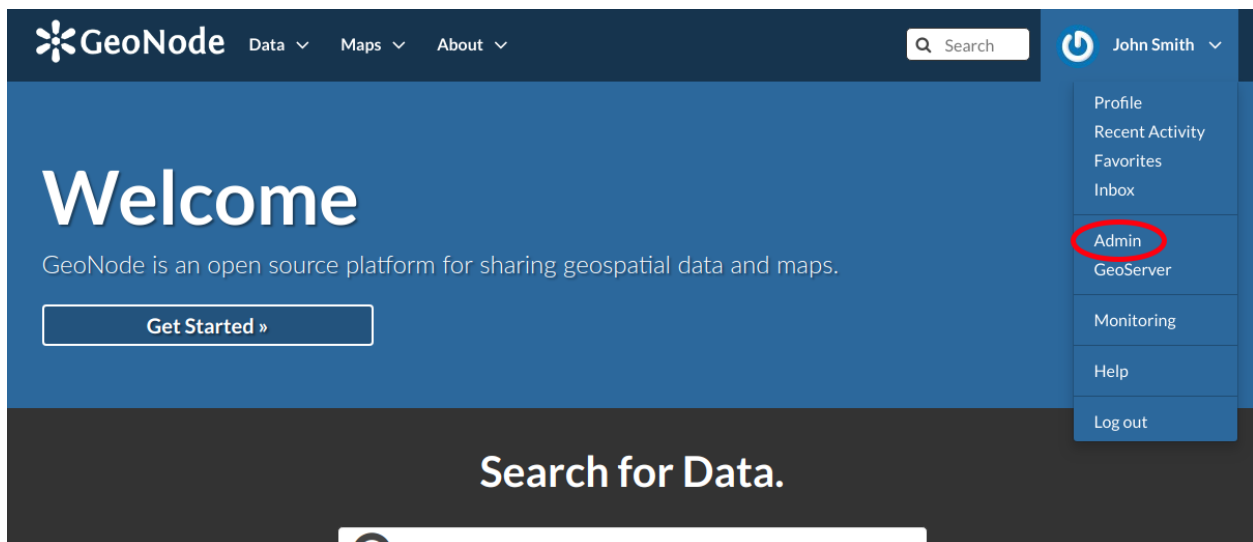


Fig. 289: The Admin Link of the User Menu

When clicking on that link the Django-based *Admin Interface* page opens and shows you all the Django models registered in GeoNode.

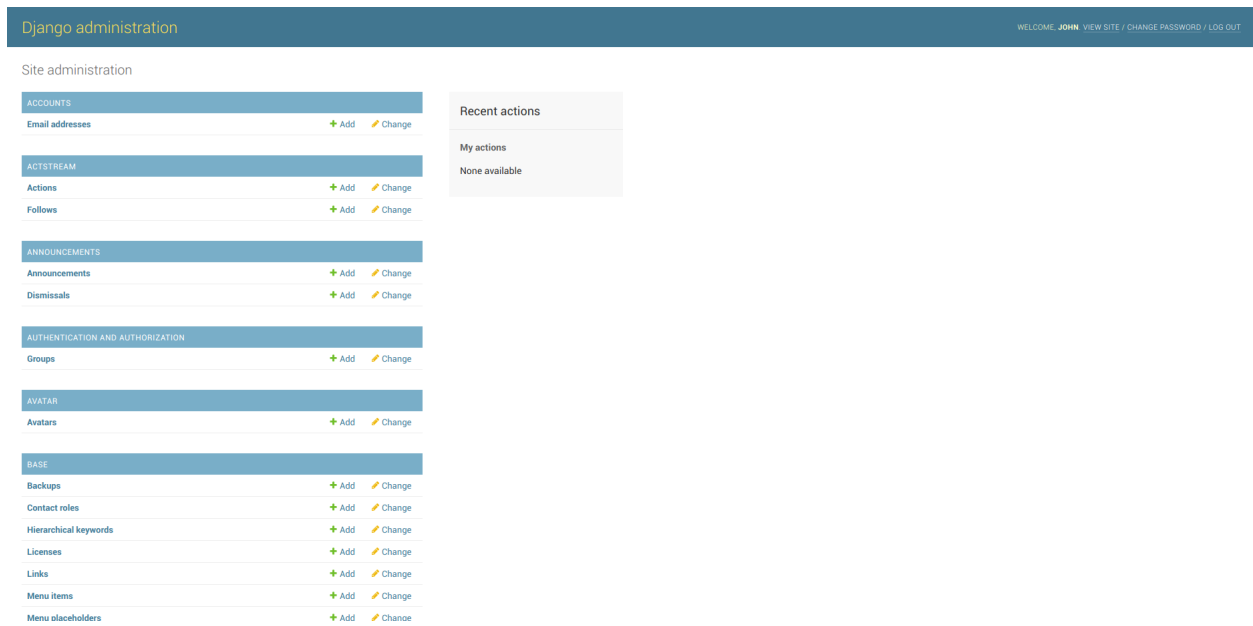


Fig. 290: The GeoNode Admin Interface

1.24.2 Reset or Change the admin password

From the *Admin Interface* you can access the *CHANGE PASSWORD* link on the right side of the navigation bar.

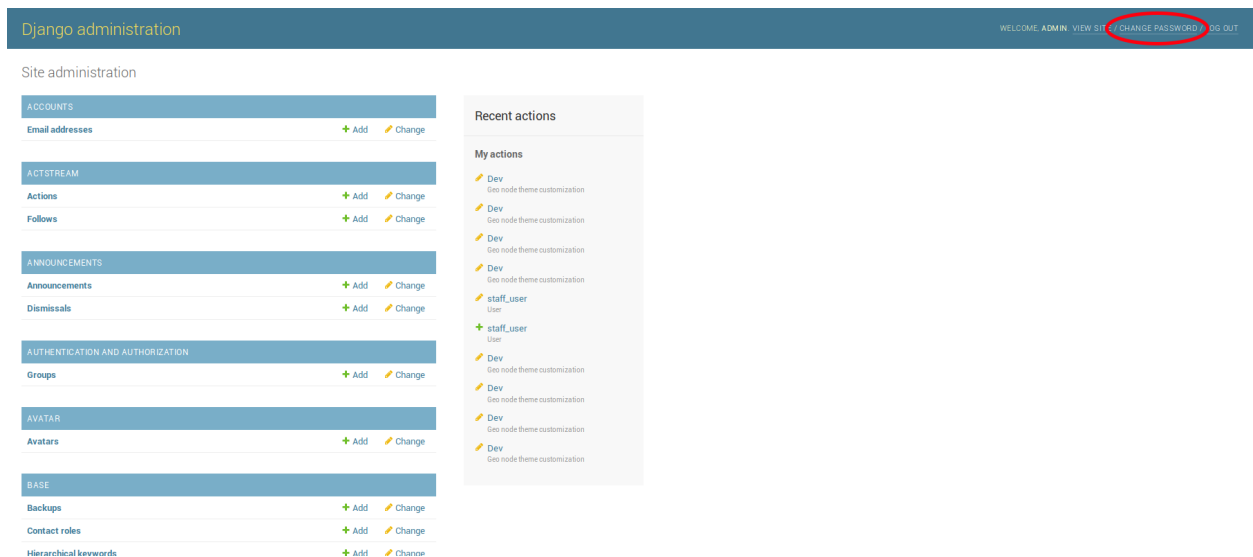


Fig. 291: The Change Password Link

It allows you to access the *Change Password Form* through which you can change your password.

Once the fields have been filled out, click on *CHANGE MY PASSWORD* to perform the change.

Django administration

WELCOME ADMIN / CHANGE PASSWORD / LOG OUT

Home / Password change

Password change

Please enter your old password, for security's sake, and then enter your new password twice so we can verify you typed it in correctly.

Old password:

New password:

New password confirmation:

CHANGE MY PASSWORD

Fig. 292: *The Change Password Form*

1.24.3 Simple Theming

GeoNode provides by default some theming options manageable directly from the Administration panel. Most of the times those options allows you to easily change the GeoNode look and feel without touching a single line of *HTML* or *CSS*.

As an *administrator* go to `http://<your_geonode_host>/admin/geonode_themes/geonodethemecustomization/`.

Django administration

WELCOME ADMIN / VIEW SITE / CHANGE PASSWORD / LOG OUT

Home / GeoNode Themes Library / Themes

Select geo node theme customization to change

ADD GEO NODE THEME CUSTOMIZATION +

Action: 0 of 1 selected

ID	IS ENABLED	NAME	DATE	DESCRIPTION
2	☒	Dev	May 3, 2019, 6:01 p.m.	

1 geo node theme customization

Fig. 293: *List of available Themes*

The panel shows all the available GeoNode themes, if any, and allows you to create new ones.

Aviso: Only one theme at a time can be **activated** (aka *enabled*). By disabling or deleting all the available themes, GeoNode will turn the gui back to the default one.

Editing or creating a new Theme, will actually allow you to customize several properties.

At least you'll need to provide a Name for the Theme. Optionally you can specify also a Description, which will allow you to better identify the type of Theme you created.

Just below the Description field, you will find the Enabled checkbox, allowing you to toggle the Theme.

Change geo node theme customization

Name:
This will not appear anywhere.

Description:
This will not appear anywhere.

Fig. 294: Theme Name and Description

Description:
This will not appear anywhere.

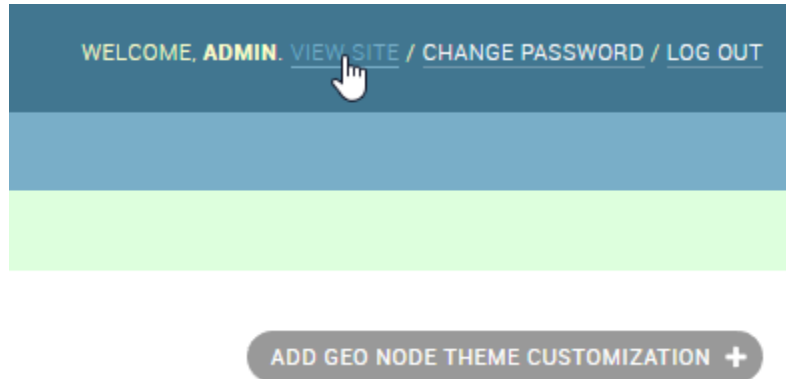
☒ **Is enabled**
Enabling this theme will disable the current enabled theme (if any).



Fig. 295: Theme Name and Description

Jumbotron and Get Started link

Nota: Remember, everytime you want to apply some changes to the Theme, you **must** save the Theme and reload the GeoNode browser tab. In order to quickly switch back to the Home page, you can just click the `VIEW SITE` link on the top-right corner of the Admin dashboard.



The next section, allows you to define the first important Theme properties. This part involves the GeoNode main page sections.

Logo	Choose file 6-logo.png
Jumbotron background	Choose file No file chosen
	☐ Hide text in the jumbotron Check this if the jumbotron background image already contains text
Welcome theme	jumbotron background ▼ Choose between using jumbotron background and slide show
Jumbotron slide show	+ Hold down "Control", or "Command" on a Mac, to select more than one.
Jumbotron title	Jumbotron title
Jumbotron content	Jumbotron content
	☐ Hide call to action
Call to action text	Call to action
Call to action link	www.call_to_action_link

Fig. 296: Jumbotron and Logo options

By changing those properties as shown above, you will easily change your default home page from this

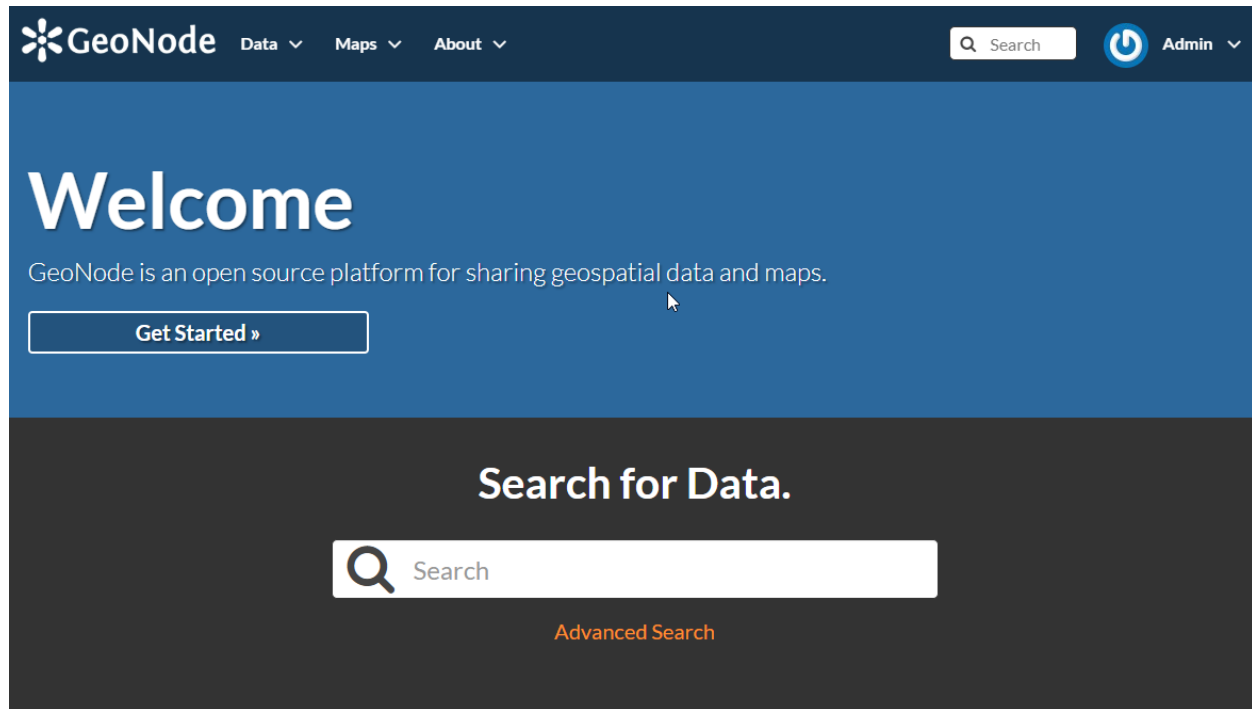


Fig. 297: *GeoNode Default Home*

to this

It is possible to optionally **hide** the Jumbotron text and/or the Call to action button

Slide show

To switch between a slide show and a jumbotron, flip the value of the welcome theme from «slide show» to «jumbotron» and vice versa to either display a jumbotron with a «get started» link or a slide show in the home page

For example, to display a slide show, change the welcome theme from jumbotron background

to slide show

Before creating a slide show, make sure you have slides to select from (in the multi-select widget) to make up the slide show.

If no slides exist, click the plus (+) button beside the slide show multi-select widget to add a new slide.

Fill in the slide name, slide content using markdown formatting, and upload a slide image (the image that will be displayed when the slide is in view).

For slide images that already contain text, hide slide content by checking the checkbox labeled «Hide text in the jumbotron slide» as shown below, then save the slide.

It is also possible to hide a slide from all slide show themes that use it by unchecking the checkbox labeled «Is enabled» as shown below.

Selecting the above slide in a slide show and enabling slide show (using the «welcome theme» configuration) will create a slide show with a slide as shown below:

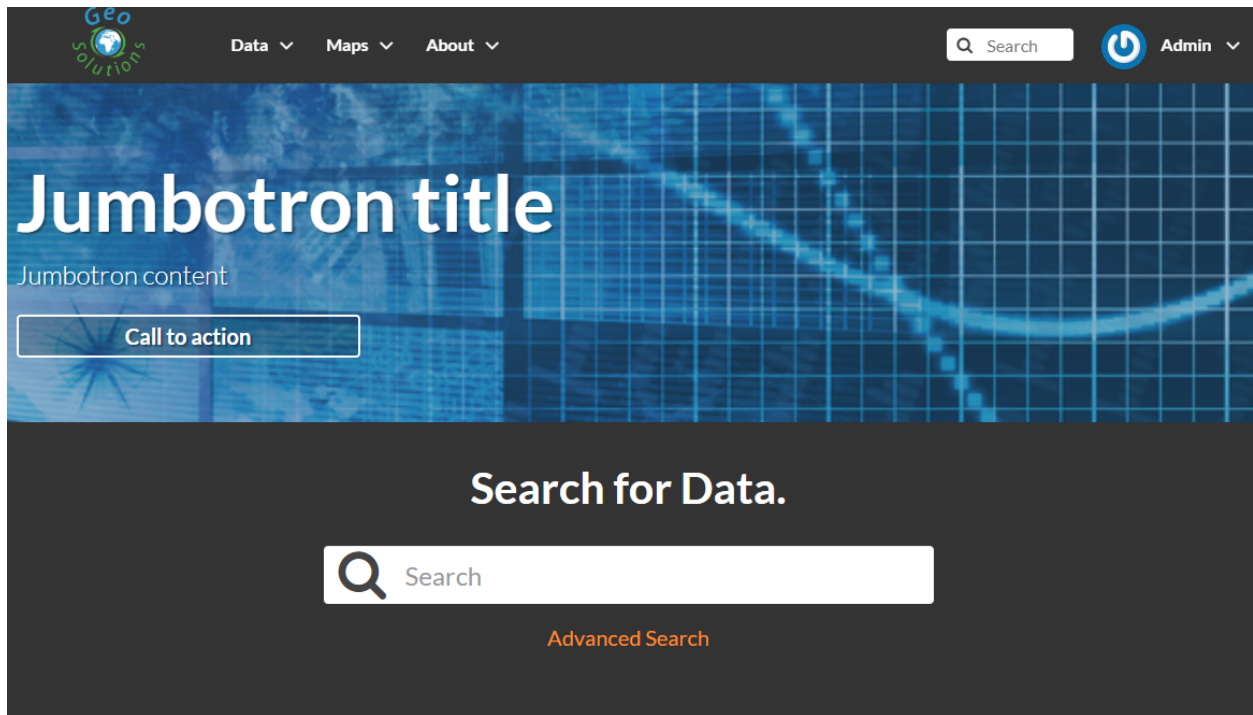


Fig. 298: Updating Jumbotron and Logo

☒ Hide text in the jumbotron 
Check this if the jumbotron background image already contains text

Jumbotron title:

Jumbotron content:

Jumbotron content

☒ Hide call to action 

Call to action text:

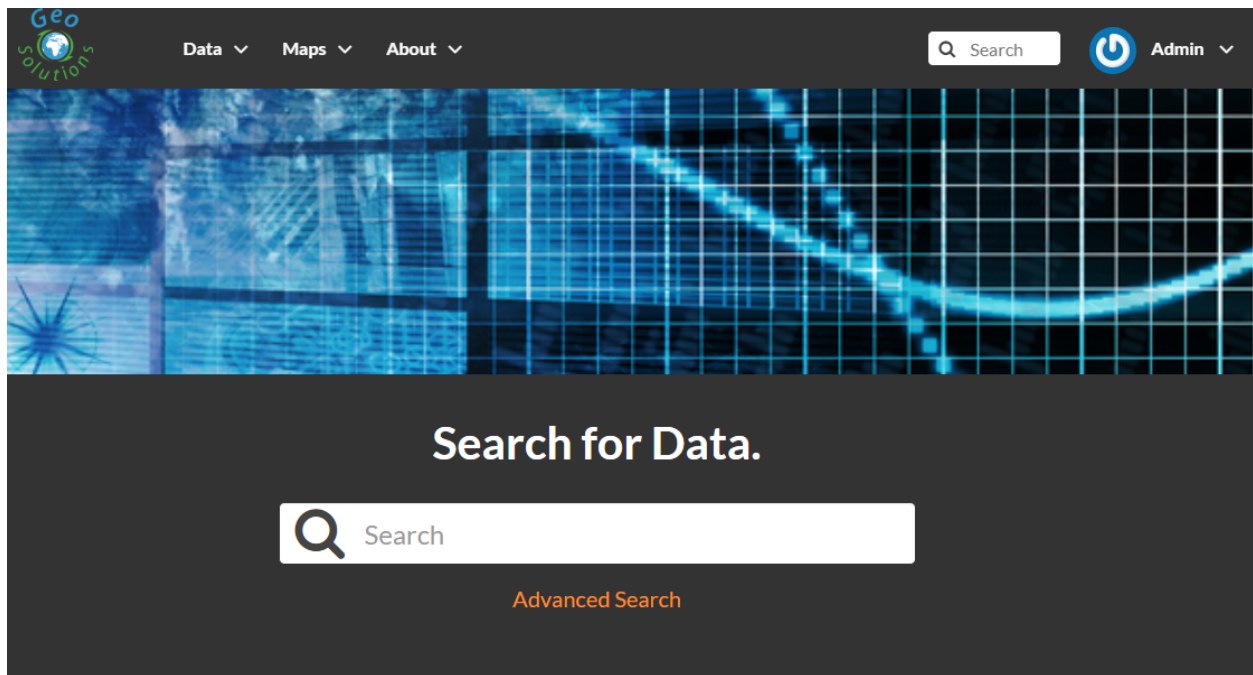


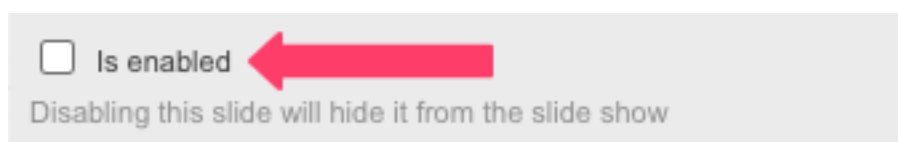
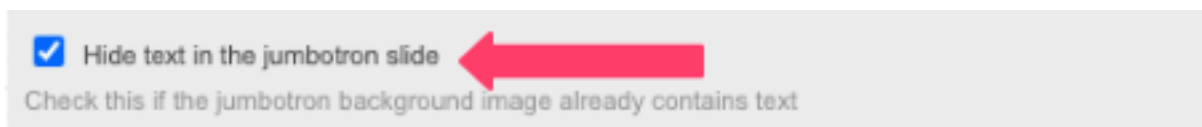
Fig. 299: Hide Jumbotron text and Call to action button

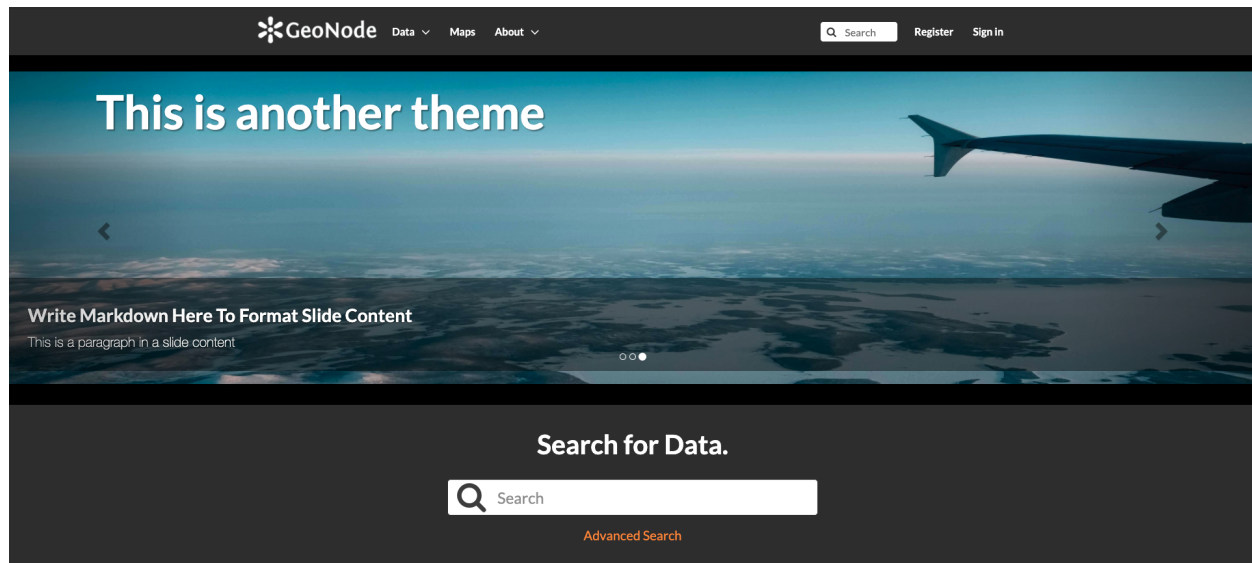




Add jumbotron theme slide

Slide name	New Slide
Jumbotron slide background	Choose file No file chosen
Jumbotron slide content	### Write Markdown Here To Format Slide Content This is a paragraph in a slide content Fill in this section with markdown
☐ Hide text in the jumbotron slide Check this if the jumbotron background image already contains text	
☒ Is enabled Disabling this slide will hide it from the slide show	





Copyright and contact info footer

The default GeoNode footer does not present any type of contact info.

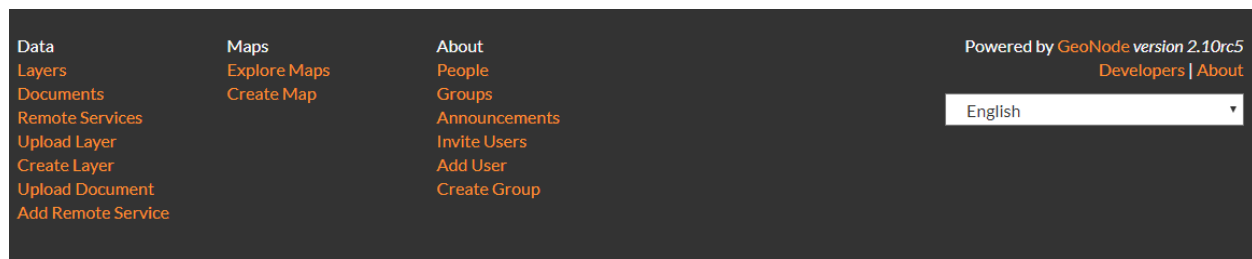


Fig. 300: *Default GeoNode Footer*

By enabling and editing the `contact_us_box` fields

it will be possible to show a simple *Contact Us* info box on the GeoNode footer section.

Similarly, by editing the `Copyright` text box and/or background color

it will be possible to show the Copyright statement to the bottom of the page

Partners

GeoNode simple theming, allows also a `Partners` section, in order to easily list links to third-party institutions collaborating to the project.

The example below shows the `Partners` section of [WorldBank CHIANG MAI URBAN FLOODING](#) GeoNode instance made through integrating theming options.

The `Partners` items can be managed through the `http://<your_geonode_host>/admin/geonode_themes/partner/` Admin section

From here it is possible to add, modify or delete partners items.

A new partner is defined by few elements, a `Logo`, a `Name`, a `Display Name` and a `Website`

☒ Enable contact us box	
Contact name:	Mr. Pibody
Contact position:	Chief Executive Officer
Contact administrative area:	Paperopoli
Contact street:	113th street
Contact postal code:	113117
Contact city:	Topolinia
Contact country:	Disney
Contact delivery point:	
Contact voice:	+00 000.000.113
Contact facsimile:	+00 000.000.117
Contact email:	pibody@paperopoli.dis

Fig. 301: *Enable contact us box*

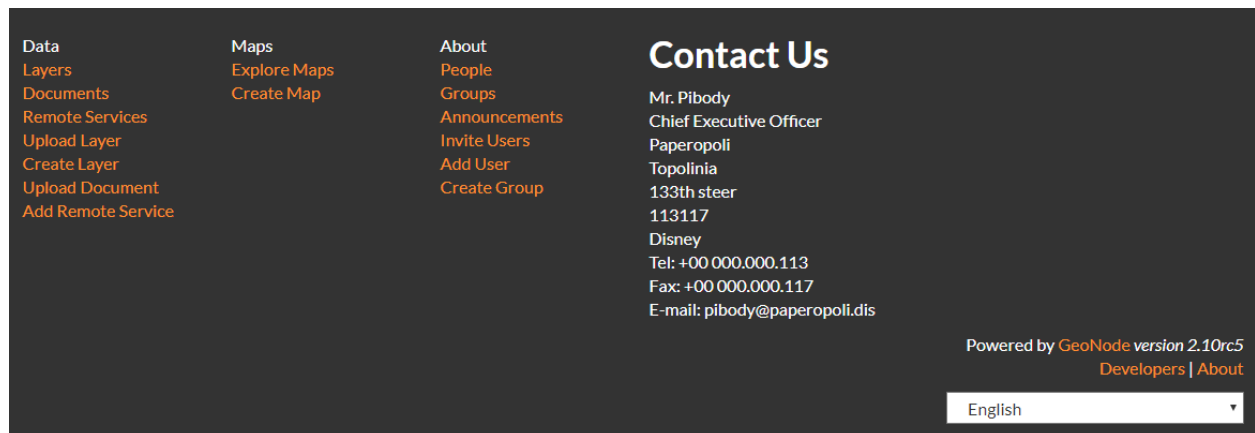


Fig. 302: *Contact Us Footer*

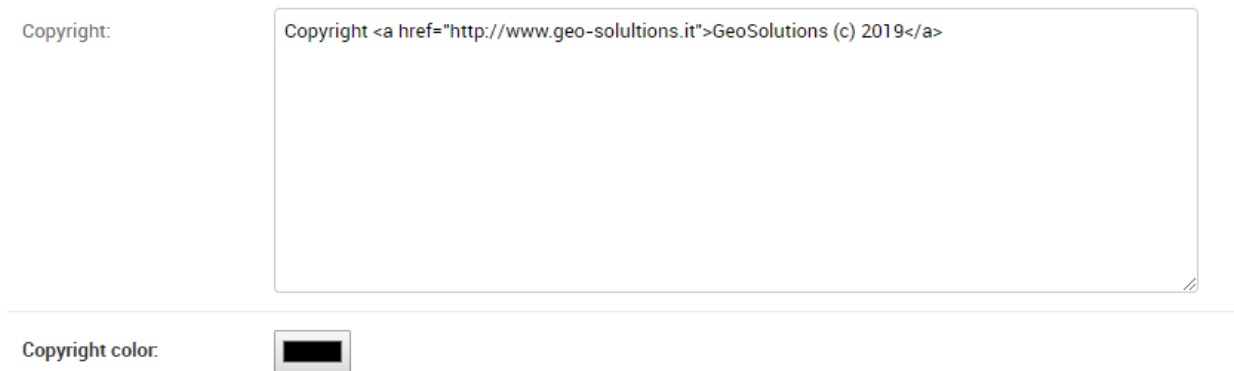


Fig. 303: *Copyright Text and Color*



Fig. 304: *Copyright*

With support from:



Data
Layers
Documents
Remote Services

Maps
Explore Maps

About
People
Groups
Invite Users

Powered by **GeoNode version 2.10rc5**
Developers | About

English

GEOSOLUTIONS S.A.S. 2019 (C)

Fig. 305: *Urban-flooding GeoNode Partners Section*

Django administration

WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home : GeoNode Themes Library : Partners

Select partner to change

Action: Go 0 of 8 selected

ID	NAME	DISPLAY NAME	WEBSITE
8	co-risk	co-risk	http://co-risk.org/
1	Disaster Risk Financing & Insurance Program	Disaster Risk Financing & Insurance Program	http://www.worldbank.org/en/programs/disaster-risk-financing-and-insurance-program
5	EOS	EOS	https://earthobservatory.sg/
3	GFDRR	GFDRR	https://www.gfdr.org/en
2	Humanitarian OpenStreetMap Team	Humanitarian OpenStreetMap Team	https://www.hotosm.org/what-we-do
7	NATCAP	NATCAP	http://www.naturalcapitalproject.org/
6	NTU	NTU	https://ase.ntu.edu.sg/
4	WORLD BANK GROUP	WORLD BANK GROUP	https://www.worldbank.org/

8 Partners

Fig. 306: *GeoNode Partners Admin Section*

Add partner

Logo:	Choose File No file chosen
Name:	
	This will not appear anywhere.
Display name:	
Website:	

Fig. 307: Add a Partner

In order to attach or detach a Partner to an existing Theme on GeoNode, you will need to edit the Theme and go to the Partners section

From here you will be able to either to change the Partners title text and/or select/deselect Partners from the multi-select box.

Nota: In order to select/deselect elements from the multi-select box, you **must** use the CTRL+CLICK button combination.

Privacy Policies and Cookie settings

By enabling the Cookies Law Info Bar checkbox (True by default)

it will be possible to allow GeoNode presenting the *Privacy Policies and Cookie settings* pop-ups and links at the bottom of the home page

There are plenty of options available, allowing you to customize contact info as long as colors of the bar and page.

One of the most important to consider it is for sure the Cookie law info bar text

The default text contained in this section is the following one

This website uses cookies to improve your experience,
check **this page
→ for details.
We'll assume you're ok with this, but you can opt-out if you wish.**

The text can be changed and customized, of course. Nevertheless it points by default to the following page

/privacy_cookies/

aka `http://<your_geonode_host>/privacy_cookies/`

Contact email:

Partners title: With support from:

Partners:

co-risk
Disaster Risk Financing & Insurance Program
EOS
GFDRR
Humanitarian OpenStreetMap Team
NATCAP
NTU
WORLD BANK GROUP

+

Hold down "Control", or "Command" on a Mac, to select more than one.

Copyright: `<strong><a style="color:#ffffff" href="http://www.geo-solutions.it">GeoSolutions S`

Fig. 308: Theme Partners Section

Copyright color:

☒ Cookies Law Info Bar 

Cookie law info bar head:

This website uses cookies

Cookie law info bar text:

This website uses cookies to improve your experience, check `<strong><a style="color:#000000" href="/privacy_cookies/">this page</a></strong>` for details. We'll assume you're ok with this, but you can opt out if you wish.

Fig. 309: Cookies Law Info Bar checkbox

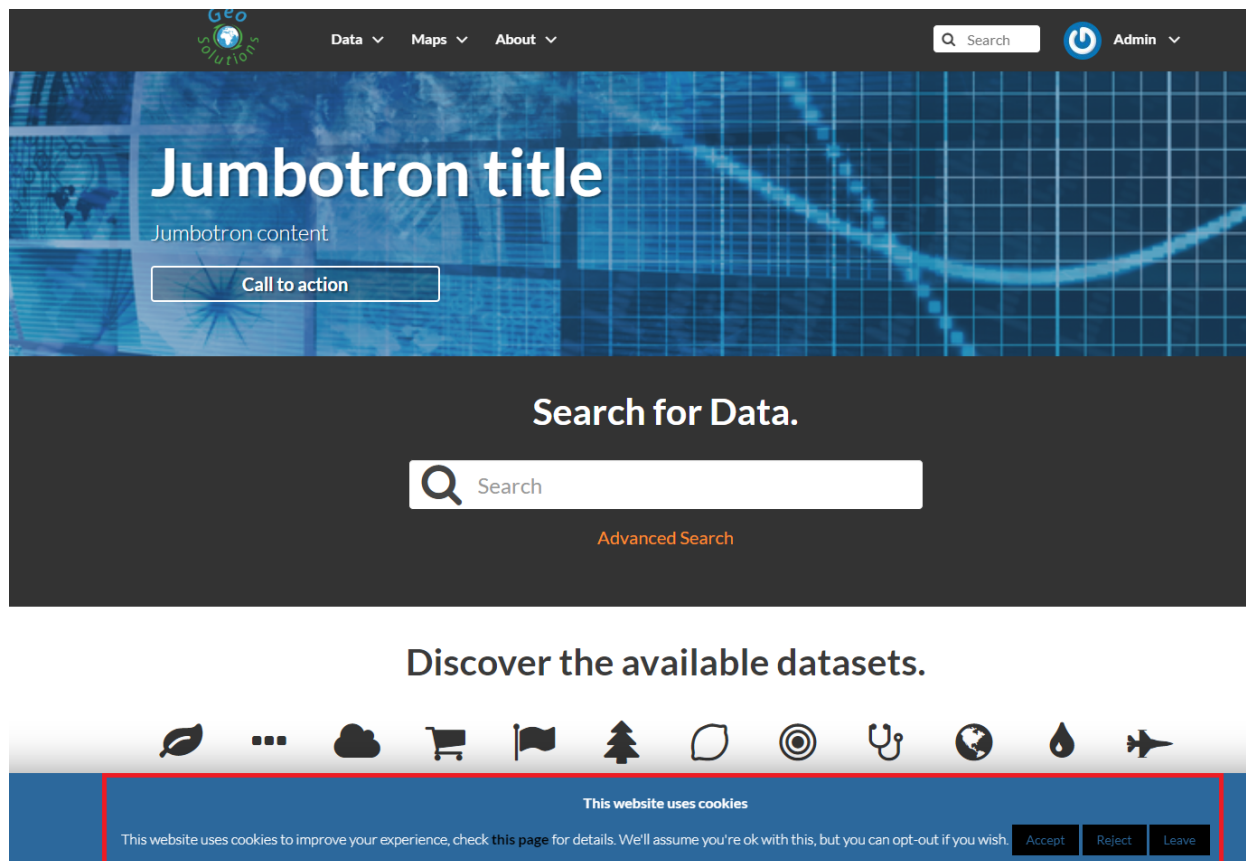


Fig. 310: Cookies Law Info Bar

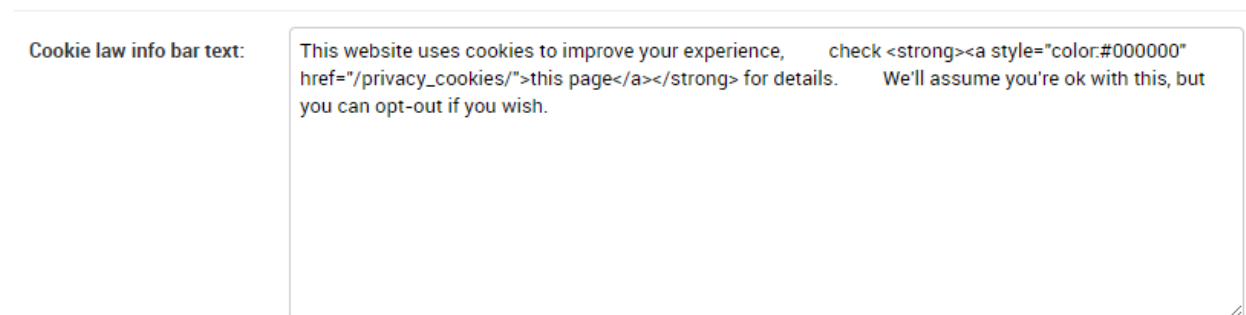


Fig. 311: Cookie law info bar text



Privacy & Cookies Policy

Privacy & Cookies Policy.

This site agrees to respect the privacy of the Website user in accordance with the applicable regulations on the protection of personal data and in particular EU Regulation 2016/679 (hereinafter the "Regulation", "GDPR").

This document ("Privacy & Cookies Policy") provides information on the processing of personal data collected through this Website (hereinafter "Website") and therefore constitutes information to the data subjects in accordance with the aforementioned regulations. Within the specific area of the Website, which collects the personal information of the user, a specific policy is normally published. The following information applies only to this Website and not to other websites accessed via links.

Pursuant to Article 13 of the Regulation, we hereby provide the following information:

DATA CONTROLLER

The Data Controller is **GEOSOLUTIONS DI GIANNECCHINI SIMONE & C.**, #DATA_CONTROLLER_ADDRESS; Tel. ##DATA_CONTROLLER_PHONE e-mail: #DATA_CONTROLLER_EMAIL.

WHAT DATA DO WE PROCESS?

The following data may be subject to processing:

Browsing Data

The processing of personal data of users who visit only the Website (i.e. without sending communications or using reserved areas) is limited to the navigation data, i.e. those for which the transmission to the Website is necessary for the operation of IT systems responsible for the management of the Website and the Internet communication protocols. This category includes the IP addresses or domain of the computer used to visit the Website and other parameters relative to the operating system used by the user to connect to the Website. The Company collects these and other data (such as, for example, the number of visits and the time spent on the Website) only for statistical purposes and in anonymous form in order to control the operation of the Website and improve its functionality. This is information that is not collected for the association with other information about users and to identify the latter; however, by their very nature, these data can allow the identification of users through processing and association with data held by third parties.

The legal basis for this processing is the legitimate interest of the Data Controller in the technical management related to the functionality and safety of the Website as defined by Art. 6.1. (f) of the Regulation

Cookies

Cookies are small text files, which the Web site places on the devices in use, such as computers or mobile devices, stored in directories used by the user's web browser. There are various types of cookies, some make the Website experience more efficient, others to enable certain functions.

The Website uses "technical" cookies, such as navigation or session cookies, or tools to make functional and optimize the navigation and use of the Website.

Fig. 312: */privacy_cookies/ Default Page*

The page contains a default generic text along with some placeholders, which, most probably, won't meet your needs.

In order to change this you have two options:

1. Change the link reported into the `Cookie law info bar` text section, to make it pointing to an external/static page.
2. Change the contents of `/geonode/templates/privacy-cookies.html` Django template accordingly to your needs; this is basically a plain HTML page which can be easily customized by using a standard text editor.

Switching between different themes

In the case you have defined more Themes, switching between them is as easy as enabling one and disabling the others.

Remember to save the Themes everytime and refresh the GeoNode home page on the browser to see the changes.

It is also important that there is **only one** Theme enabled **at a time**.

In order to go back to the standard GeoNode behavior, just disable or delete all the available Themes.

1.24.4 Add a new user

In GeoNode, administrators can manage other users. For example, they can *Add New Users* through the following form.

Fig. 313: Adding New Users

The form above can be reached from the *Admin Panel* at the following path: *Home > People > Users*. Click on *ADD USER +* to open the form page.

It is also available, in the GeoNode UI, the *Add User* link of the *About* menu in the navigation bar.

To perform the user creation fill out the required fields (*username* and *password*) and click on *SAVE*. You will be redirected to the *User Details Page* which allows to insert further information about the user.

The user will be visible into the *Users List Page* of the *Admin Panel* and in the *People Page* (see *Visualização de informações de outros utilizadores*).

Django administration

WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home · People · Users

Select user to change

Q Search

Action: Go 0 of 22 selected

☐	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS	ACTIVE
☐	admin	alessio.fabiani@geo-solutions.it	Admin		✓	✓
☐	AnonymousUser				✗	✓
☐	capoti	pcorti@gmail.com	Paolo	Corti	✗	✓
☐	cfuhrer	cfuhrer@ohchr.org	Caroline	Fuhrer	✗	✓
☐	dimitris.karakostis	dkarakostis@gmail.com			✗	✓
☐	dpadron	diegopadron@gmail.com			✗	✓
☐	fbartoli	xbartolone@gmail.com	Francesco	Bartoli	✗	✓
☐	federico	gianoli.federico@gmail.com			✗	✗
☐	fgianoli	alpacagis@gmail.com	Federico	Gianoli	✗	✓
☐	giohappy	giovanni.allegri@geo-solutions.it	Giovanni	Allegri	✓	✓
☐	gioscarda	giovanni.scardapane@geo-solutions.it			✗	✓
☐	gtrash	gaptrash@gmail.com			✗	✗
☐	tzehvor	sinopceknob5@hotmail.com			✗	✗

ADD USER +

FILTER

By staff status

All
Yes
No

By superuser status

All
Yes
No

By active

All
Yes
No

By groups

All
anonymous
Test-Group
john-smith-foundation-team
-

Fig. 314: The Add User button in the Users List page

Geo Solutions

Data ▾ Maps ▾ About ▾

Q Search Admin ▾

People
Groups
Group Categories
Announcements
Invite Users
Add User
Create Group

Fig. 315: Add User Link

The screenshot shows the 'Django administration' interface for user management. The breadcrumb trail is 'Home > People > Users > joe'. The page title is 'Change user'. There are two buttons in the top right: 'HISTORY' and 'VIEW ON SITE'. The form is divided into several sections:

- Username:** A text input field containing 'joe'. Below it, a note says 'Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.'
- Password:** A text input field containing a long alphanumeric string. Below it, a note says 'Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.'
- Personal info:** A section with three text input fields: 'First name' (containing 'Joe'), 'Last name' (containing 'Banana'), and 'Email address' (containing 'banana.joe@mail.com').
- Permissions:** A section with three checkboxes:
 - ☒ **Active**: Designates whether this user should be treated as active. Unselect this instead of deleting accounts.
 - ☐ **Staff status**: Designates whether the user can log into this admin site.
 - ☐ **Superuser status**: Designates that this user has all permissions without explicitly assigning them.
- Groups:** A section with two panels. The left panel, 'Available groups', has a search bar with 'Filter' and a list of groups including 'joe', 'smith', and 'foundation'. The right panel, 'Chosen groups', has a list with 'anonymous' and a green plus sign to add more groups.

Fig. 316: *The User Details Page*

The screenshot shows the 'GeoNode' 'Explore People' page. The top navigation bar includes the 'GeoNode' logo, 'Data', 'Maps', and 'About' menus, a search bar, and an 'Admin' button. The main heading is 'Explore People'. Below the heading, there is a search bar with a dropdown arrow and the word 'SEARCH'. The search input field contains the text 'joe' and is circled in red. To the right of the search bar, it says 'Total: 1'. On the far right, there is a button with a magnifying glass icon and a dropdown arrow. Below the search bar, there is a user profile card for 'Joe Banana'. The card has a blue circular icon with a white 'G' and the text 'Joe Banana', 'No Organization Info', and three icons (a diamond, a location pin, and a document) each followed by a '0'.

Fig. 317: *The User in the People page*

1.24.5 Activate/Disable a User

When created, new users are *active* by default. You can check that in the *User Details Page* from the *Admin Panel* (see the picture below).

Django administration

Home > People > Users > joe

Change user

Username:
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password: **algorithm: sha1 salt: lZ***** hash: 5f3e2b*******
Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using [this form](#).

Personal info

First name:

Last name:

Email address:

Permissions

☒ **Active**
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

☐ **Staff status**
Designates whether the user can log into this admin site.

☐ **Superuser status**
Designates that this user has all permissions without explicitly assigning them.

Groups:

Available groups ⓘ

Filter

infn_emith_foundation_team

Chosen groups ⓘ

anonymous

Fig. 318: New Users Active by default

Active users can interact with other users and groups, can manage resources and, more in general, can take actions on the GeoNode platform.

Untick the *Active* checkbox to disable the user. It will be not considered as user by the GeoNode system.

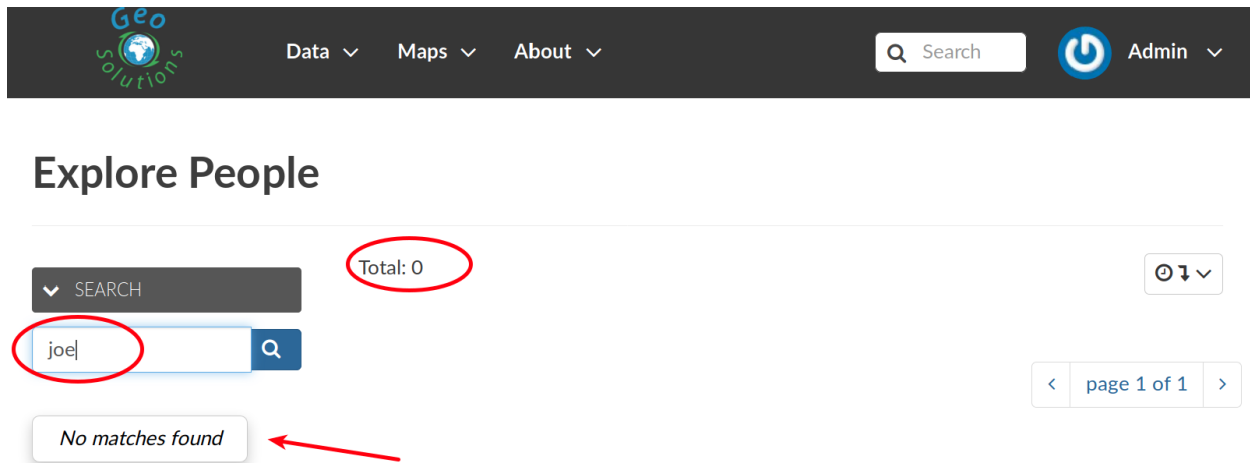


Fig. 319: Disabled Users

1.24.6 Change a User password

GeoNode administrators can also change/reset the password for those users who forget it. As shown in the picture below, click on [this](#) form link from the *User Details Page* to access the *Change Password Form*.

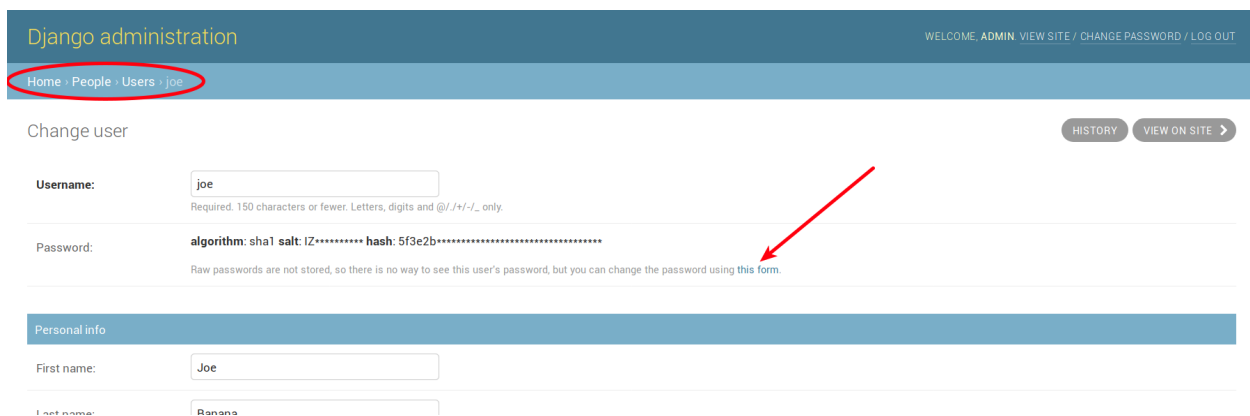


Fig. 320: Changing Users Passwords

The *Change User Password Form* should look like the following one. Insert the new password two times and click on **CHANGE PASSWORD**.

1.24.7 Promoting a User to Staff member or superuser

Active users have not access to admin tools. GeoNode makes available those tools only to *Staff Members* who have the needed permissions. *Superusers* are staff members with full access to admin tools (all permissions are assigned to them).

Administrators can promote a user to *Staff Member* by ticking the **Staff status** checkbox in the *User Details Page*. To make some user a *Superuser*, the **Superuser status** checkbox should be ticked. See the picture below.

GeoNode administration

Home › People › Users › joe › Change password

Change password: joe

Enter a new password for the user **joe**.

Password:

Password (again):
Enter the same password as before, for verification.

CHANGE PASSWORD

Fig. 321: *Changing Users Passwords*

Django administration WELCOME, ADMIN / VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › People › Users › joe

Change user HISTORY VIEW ON SITE

Username:
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password: **algorithm:** sha1 **salt:** IZ***** **hash:** 5f3e2b*****
Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

Personal info

First name:

Last name:

Email address:

Permissions

☒ **Active**
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

☒ **Staff status**
Designates whether the user can log into this admin site.

☒ **Superuser status**
Designates that this user has all permissions without explicitly assigning them.

Groups:

Available groups:

Chosen groups:

Fig. 322: *Staff and Superuser permissions*

1.24.8 Creating a Group

In GeoNode is possible to create new groups with set of permissions which will be inherited by all the group members.

The creation of a Group can be done both on the GeoNode UI and on the *Admin Panel*, we will explain how in this paragraph.

The *Create Groups* link of *About* menu in the navigation bar allows administrators to reach the *Group Creation Page*.

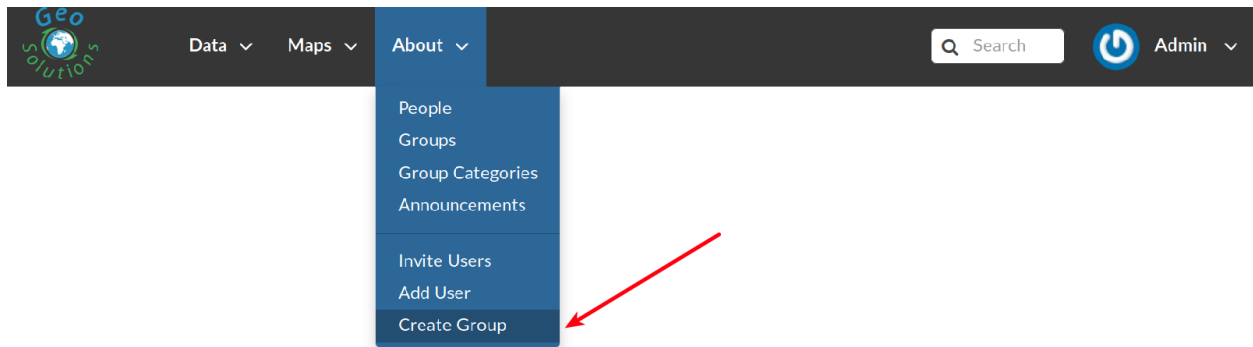


Fig. 323: The Create Group Link

The following form will open.

Fill out all the required fields and click *Create* to create the group. The *Group Details Page* will open.

The new created group will be searchable in the *Groups List Page*.

Nota: The *Create a New Group* button on the *Groups List Page* allows to reach the *Group Creation Form*.

As already mentioned above, groups can also be created from the Django-based *Admin Interface* of GeoNode.

The *Groups* link of the *AUTHENTICATION AND AUTHORIZATION* section allows to manage basic Django groups which only care about permissions.

To create a GeoNode group you should take a look at the *GROUPS* section.

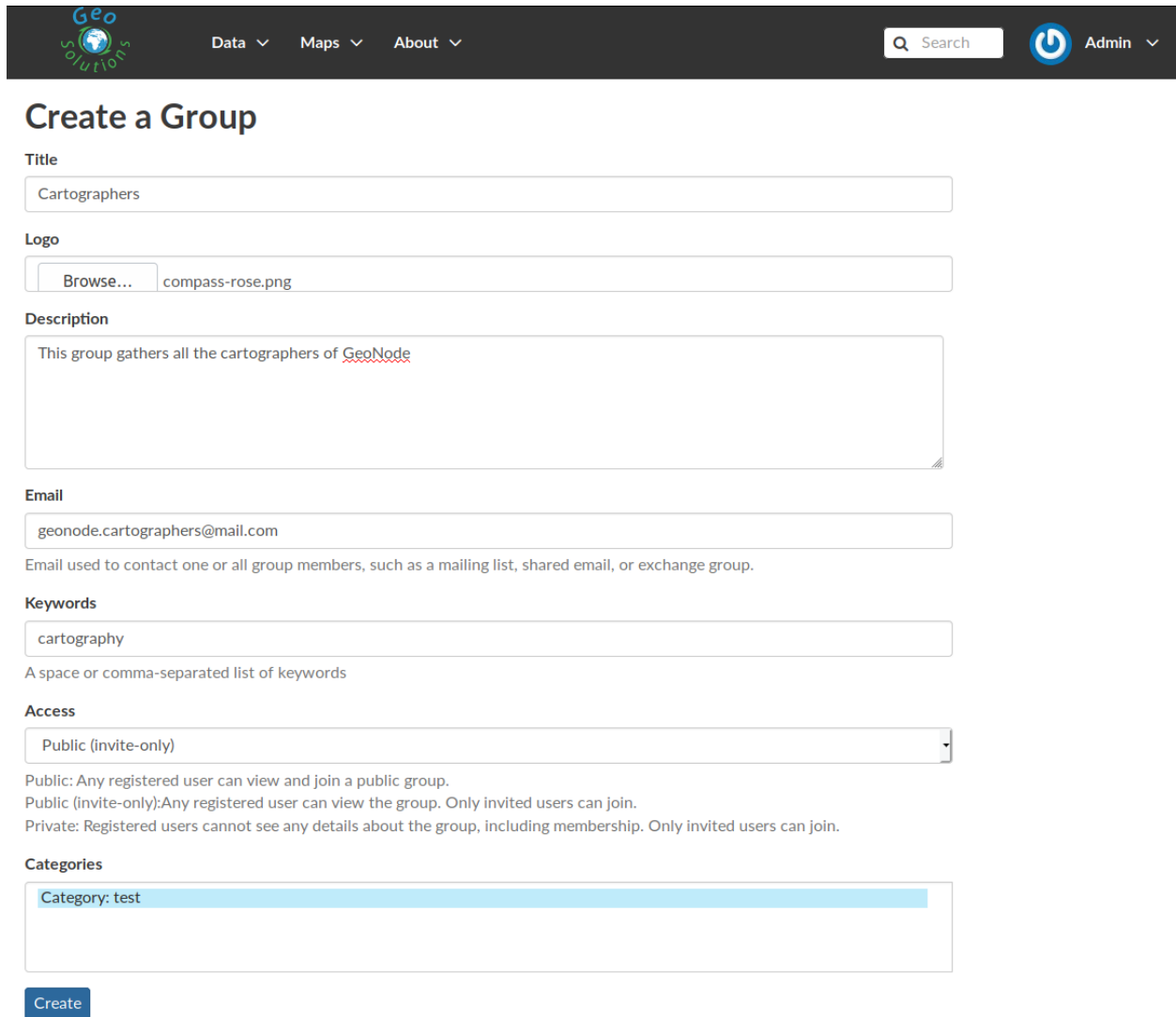
As you can see, GeoNode provides two types of groups. You will learn more about that in the next paragraph.

Types of Groups

In GeoNode users can be grouped through a *Group Profile*, an enhanced Django group which can be enriched with some further information such as a description, a logo, an email address, some keywords, etc. It also possible to define some *Group Categories* based on which those group profiles can be divided and filtered.

A new **Group Profile** can be created as follow:

- click on the *Group Profile + Add* button
- fill out all the required fields (see the picture below), *Group Profiles* can be explicitly related to group categories
- click on *SAVE* to perform the creation, the new created group profile will be visible in the *Group Profiles List*



Create a Group

Title
Cartographers

Logo
Browse... compass-rose.png

Description
This group gathers all the cartographers of GeoNode

Email
geonode.cartographers@mail.com
Email used to contact one or all group members, such as a mailing list, shared email, or exchange group.

Keywords
cartography
A space or comma-separated list of keywords

Access
Public (invite-only)
Public: Any registered user can view and join a public group.
Public (invite-only): Any registered user can view the group. Only invited users can join.
Private: Registered users cannot see any details about the group, including membership. Only invited users can join.

Categories
Category: test

Create

Fig. 324: *The Group Creation Form*


Data ▾
Maps ▾
About ▾


Admin ▾

Cartographers

Last Modified: June 26, 2019, 12:41 p.m.



This group gathers all the cartographers of GeoNode

cartography

✉ geonode.cartographers@mail.com

test

[Edit Group Details](#)

[Manage Group Members](#)

[Delete this Group](#)

[Group Activities](#)

Permissions

This group is **Public (invite-only)**. Anyone may view this group but membership is by invitation only.

Managers



admin

GeoSolutions

Members

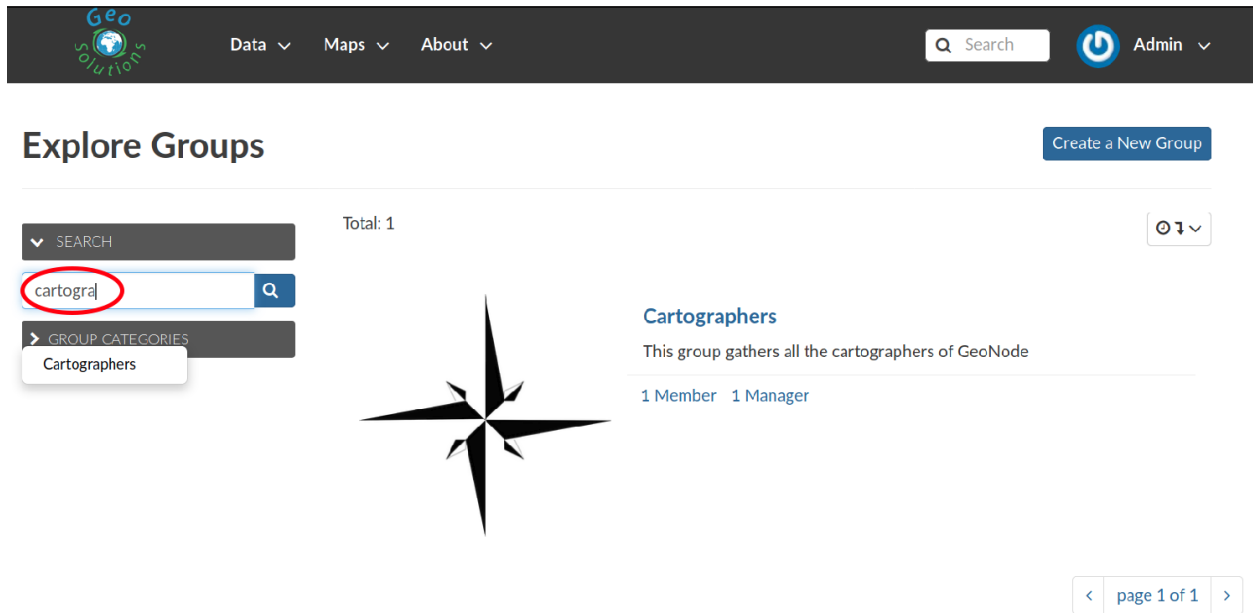


Admin

GeoSolutions

◇ 39
📍 10
📄 1

Fig. 325: *The Group Details Page*

Fig. 326: *The Groups List Page*Fig. 327: *The Groups Section on the Admin Panel*

Django administration

WELCOME, ADMIN / [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home / Groups / Group profiles / Add group profile

Add group profile

Title:

Transportation Planners

Title [en]:

Slug:

transportation-planners

Logo:

Browse...

transport_logo.png

Description:

Users interested in transports

Description [en]:

Email:

transport.group@mail.com

Email used to contact one or all group members, such as a mailing list, shared email, or exchange group.

Keywords:

railways, roads

A space or comma-separated list of keywords

Access:

Public

Public: Any registered user can view and join a public group.
Public (invite-only): Any registered user can view the group. Only invited users can join.
Private: Registered users cannot see any details about the group, including membership. Only invited users can join.

Categories:

Category: Transport

+

Hold down "Control", or "Command" on a Mac, to select more than one.

GROUP MEMBERS

USER	ROLE	JOINED	DELETE?
johnsmith	Manager	Date: 2019-06-26 Today Time: 15:20:58 Now	
joe	Member	Date: 2019-06-26 Today Time: 15:20:58 Now	
-----	-----	Date: 2019-06-26 Today Time: 15:20:58 Now	

+ Add another Group member

Save and add another

Save and continue editing

SAVE

Fig. 328: A new Group Profile

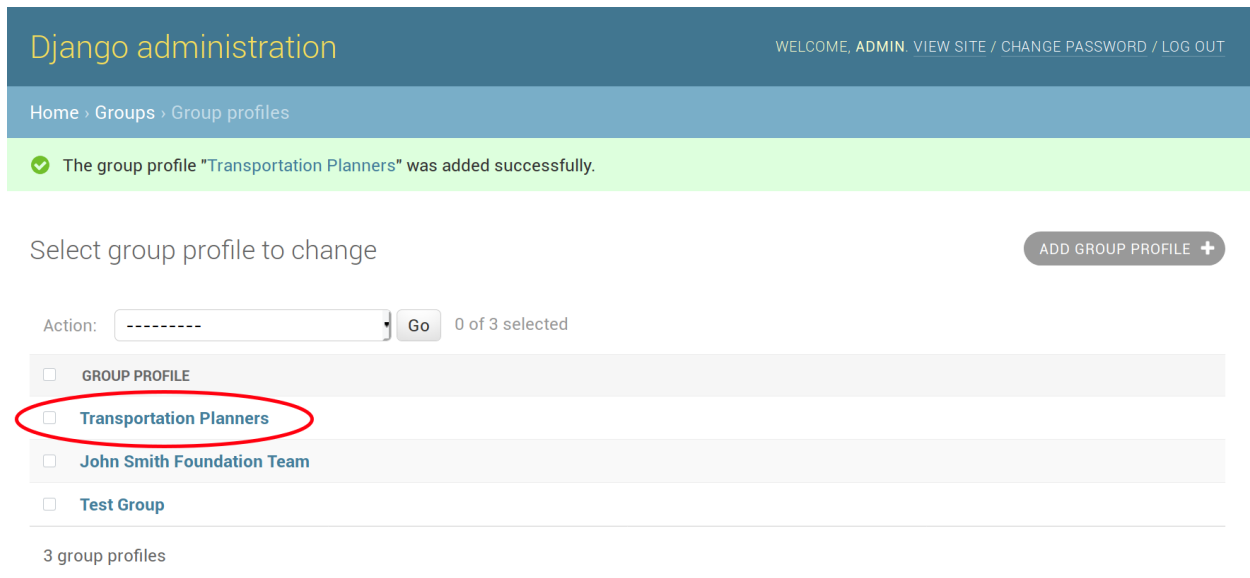


Fig. 329: The Group Profiles List

Group Categories

Group Profiles can also be related to *Group Categories* which represents common topics between groups. In order to add a new **Group Category** follow these steps:

- click on the *Group Categories + Add* button
- fill out the creation form (type *name* and *description*)
- click on *SAVE* to perform the creation, the new created category will be visible in the *Group Categories List*

When a GeoNode resource (layer, document or maps) is associated to some *Group Profile*, it is also possible to retrieve the *Group Category* it belongs to.

So when searching for resources (see [Procurando Dados](#)) you can also filter the data by group category.

1.24.9 Managing a Group

Through the *Groups* link of *About* menu in the navigation bar, administrators can reach the *Groups List Page*.

In that page all the GeoNode *Group Profiles* are listed.

For each group some summary information (such as the *title*, the *description*, the number of *members* and *managers*) are displayed near the *Group Logo*.

Administrators can manage a group from the *Group Profile Details Page* which is reachable by clicking on the *title* of the group.

As shown in the picture above, all information about the group are available on that page:

- the group *Title*;
- the *Last Editing Date* which shows a timestamp corresponding to the last editing of the group properties;
- the *Keywords* associated with the group;

Django administration WELCOME, ADMIN [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

[Home](#) > [Groups](#) > [Group Categories](#) > Add group category

Add group category

Name [en]:

Description:

All about transport

Slug:

[Save and add another](#) [Save and continue editing](#) [SAVE](#)

Fig. 330: A new Group Category

Django administration WELCOME, ADMIN [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

[Home](#) > [Groups](#) > [Group Categories](#)

Select group category to change [ADD GROUP CATEGORY +](#)

Action: [Go](#) 0 of 1 selected

☐	NAME	SLUG
☐	Transport	transport

1 group category

Fig. 331: The Group Categories List

The screenshot shows the 'Explore Layers' interface. On the left, under 'Filters', the 'GROUP CATEGORIES' filter is expanded, and 'Transport' is selected and circled in red. A red arrow points from this selection to the 'TRANSPORTATION PLANNERS' group category in the search results. The search results show '2 Layers found'. The first layer is 'roads' (Berlin Roads) by johnsmith, dated 10 Jun 2019, with 5 views and 0 shares. The second layer is 'railways' by johnsmith, dated 7 Jun 2019, with 2 views and 0 shares. Both layers are categorized under 'TRANSPORTATION PLANNERS', which is circled in red in the results. The page includes a search bar, navigation tabs (Data, Maps, About), and an 'Admin' button.

Fig. 332: Filtering Layers by Group Category

The screenshot shows the navigation bar of the GeoNode interface. The 'About' dropdown menu is open, displaying options: People, Groups, Group Categories, Announcements, Invite Users, Add User, and Create Group. A red arrow points to the 'Groups' link in this menu. The navigation bar also includes a search bar and an 'Admin' button.

Fig. 333: The Groups Link in the navigation bar


Data ▾
Maps ▾
About ▾




Admin ▾

Explore Groups

Create a New Group

▼ SEARCH



► GROUP CATEGORIES

Total: 4






Test Group

Test

1 Member 1 Manager



John Smith Foundation Team

Members of the Team of John Smith

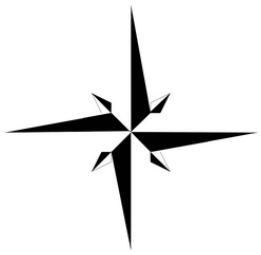
1 Member 1 Manager



Transportation Planners

Users interested in transports

3 Members 1 Manager



Cartographers

This group gathers all the cartographer of GeoNode

1 Member 1 Manager

<

page 1 of 1

>

Fig. 334: Group Profiles List Page



[Data](#) [Maps](#) [About](#)

 [Admin](#)

John Smith Foundation Team

Last Modified: June 10, 2019, 2:51 p.m.

Members of the Team of John Smith

OSM john smith transport

[Edit Group Details](#)

[Manage Group Members](#)

[Delete this Group](#)

[Group Activities](#)

Permissions

This group is **Public**. Anyone may join this group.

Managers

 [admin](#)
GeoSolutions

Members



Admin
GeoSolutions

 39

 10

 1

[<](#) [page 1 of 1](#) [>](#)

Fig. 335: Group Profile Details Page

- *Permissions* on the group (Public, Public(invite-only), Private);
- *Members* who join the group;
- *Managers* who manage the group.

There are also four links:

- The *Edit Group Details* link opens the *Group Profile Form* through which the following properties can be changed:
 - *Title*.
 - *Logo* (see next paragraphs).
 - *Description*.
 - *Email*, to contact one or all group members.
 - *Keywords*, a comma-separated list of keywords.
 - *Access*, which regulates permissions:
 - * *Public*: any registered user can view and join a public group.
 - * *Public (invite-only)*: only invited users can join, any registered user can view the group.
 - * *Private*: only invited users can join the group, registered users cannot see any details about the group, including membership.
 - *Categories*, the group categories the group belongs to.
- *Managing Group Members* (see next paragraphs).
- the *Delete this Group*, click on it to delete the Group Profile. GeoNode requires you to confirm this action.
- the *Group Activities* drives you to the *Group Activities Page* where you can see all layers, maps and documents associated with the group. There is also a *Comments* tab which shows comments on those resources.

Group Logo

Each group represents something in common between its members. So each group should have a *Logo* which graphically represents the idea that identify the group.

On the *Group Profile Form* page you can insert a logo from your disk by click on *Browse...*

Click on *Update* to apply the changes.

Take a look at your group now, you should be able to see that logo.



[Data](#) [Maps](#) [About](#)

 [Admin](#)

John Smith Foundation Team

Last Modified: June 10, 2019, 2:51 p.m.

Members of the Team of John Smith

OSM john smith transport

[Edit Group Details](#)

[Manage Group Members](#)

[Delete this Group](#)

[Group Activities](#)

Permissions

This group is **Public**. Anyone may join this group.

Managers

 admin
GeoSolutions

Members



Admin
GeoSolutions

39 10 1

[<](#)

page 1 of 1

[>](#)

Fig. 336: Group Profile Details Page

Are you sure you want to remove the group: John Smith Foundation Team?

[Yes, I am sure](#) [No, don't remove it](#)

Fig. 337: Confirm Group Deletion

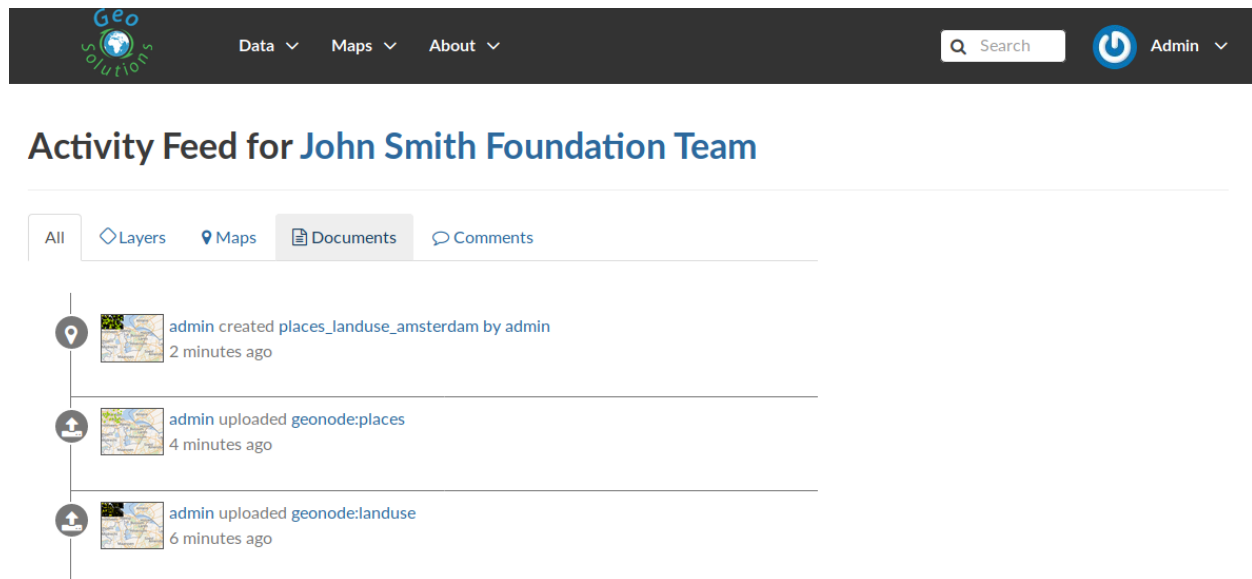


Fig. 338: *Group Activities*

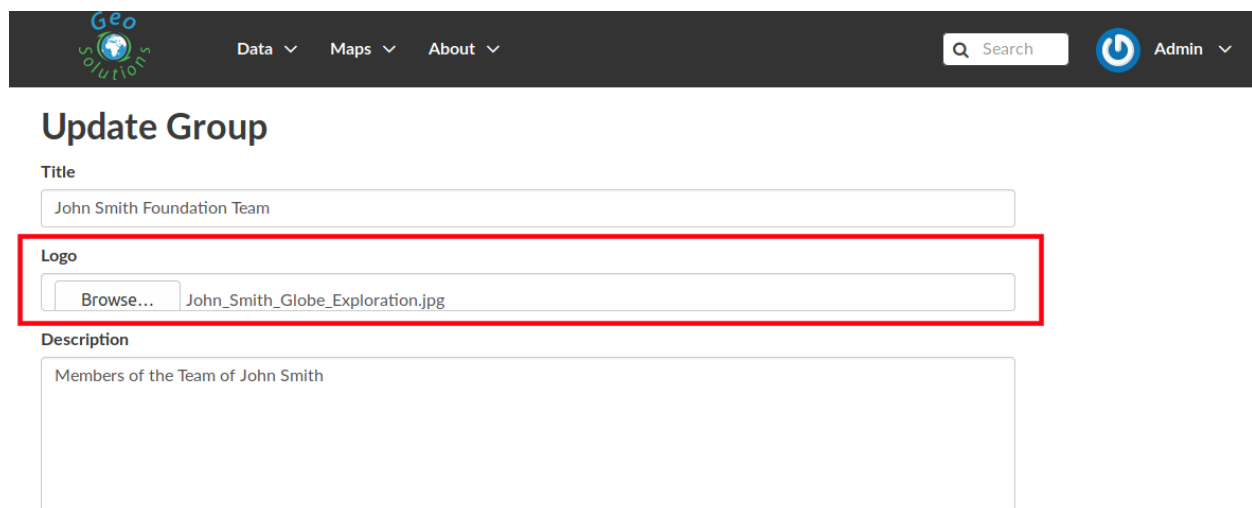


Fig. 339: *Editing the Group Logo*

Data ▾ Maps ▾ About ▾Search Admin ▾

John Smith Foundation Team

Last Modified: June 27, 2019, 9:14 a.m.



Members of the Team of John Smith

OSM john smith transport

[Edit Group Details](#)[Manage Group Members](#)[Delete this Group](#)[Group Activities](#)

Permissions

This group is **Public**. Anyone may join this group.

Managers

 admin
GeoSolutions

Members



Admin
GeoSolutions

 41  11  1

Fig. 340: *The Group Logo*

Managing Group members

The *Manage Group Members* link opens the *Group Members Page* which shows *Group Members* and *Group Managers*. **Managers** can edit group details, can delete the group, can see the group activities and can manage memberships. Other **Members** can only see the group activities.

In Public Groups, users can join the group without any approval. Other types of groups require the user to be invited by the group managers.

Only group managers can *Add new members*. In the picture below, you can see the manager can search for users by typing their names into the *User Identifiers* search bar. Once found, he can add them to the group by clicking the *Add Group Members* button. The *Assign manager role* flag implies that all the users found will become managers of the group.

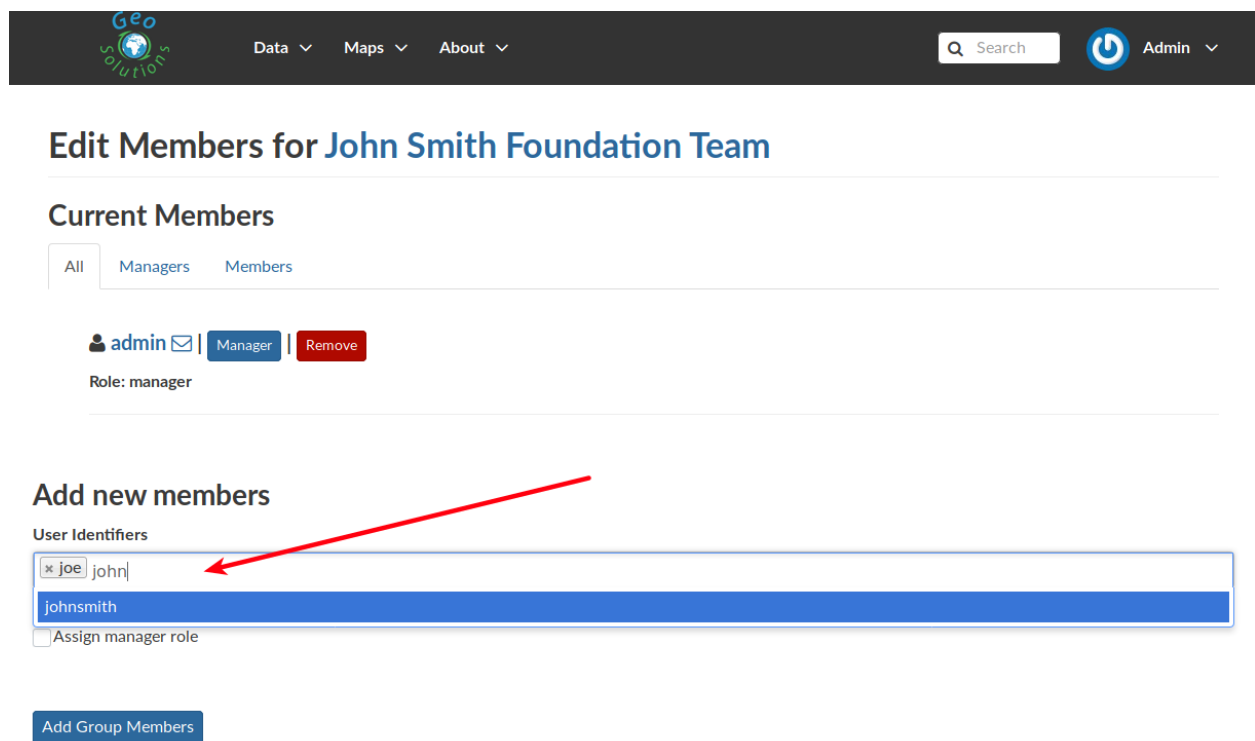
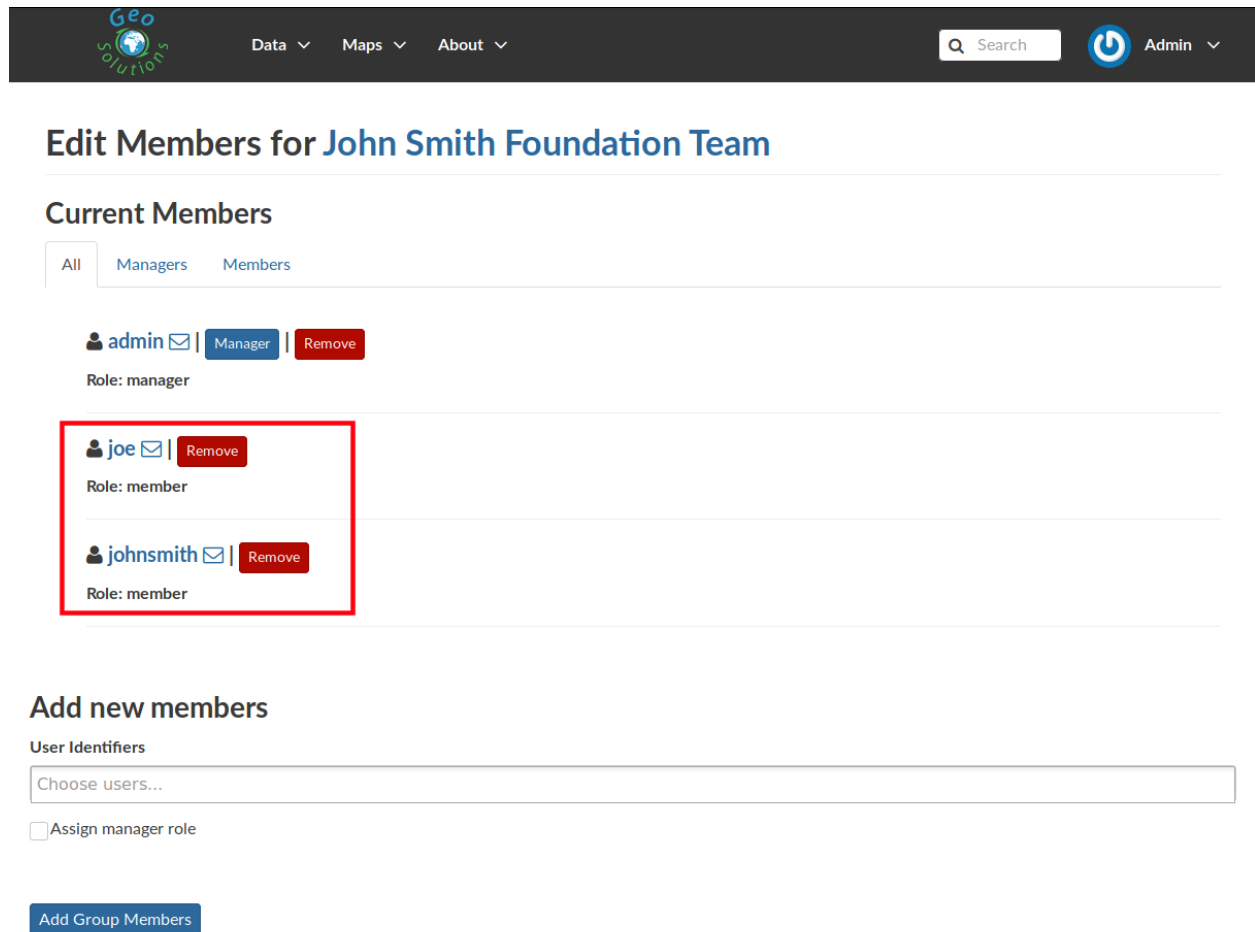


Fig. 341: Adding a new Member to the Group

The following picture shows you the results.

If you want to change the role of group members after adding them, you can use the «promote» button to make a member into a manager, and the «demote» button to make a manager into a regular member.



Edit Members for John Smith Foundation Team

Current Members

All Managers Members

admin | Manager | Remove
Role: manager

joe | Remove
Role: member

johnsmith | Remove
Role: member

Add new members

User Identifiers

Choose users...

☐ Assign manager role

Add Group Members

Fig. 342: *New Members of the Group*

1.24.10 Group based advanced data workflow

By default GeoNode is configured to make every resource (Layer, Document or Map) suddenly available to everyone, i.e. publicly accessible even from anonymous/non-logged in users.

It is actually possible to change few configuration settings in order to allow GeoNode to enable an advanced publication workflow.

With the advanced workflow enabled, your layer, document or map won't be automatically published (i.e. made visible and accessible for all, contributors or simple users).

For now, your item is only visible by yourself, the manager of the group to which the layer, document or map is linked (this information is filled in the metadata), the members of this group, and the GeoNode Administrators.

Before being published, the layer, document or map will follow a two-stage review process, which is described below:

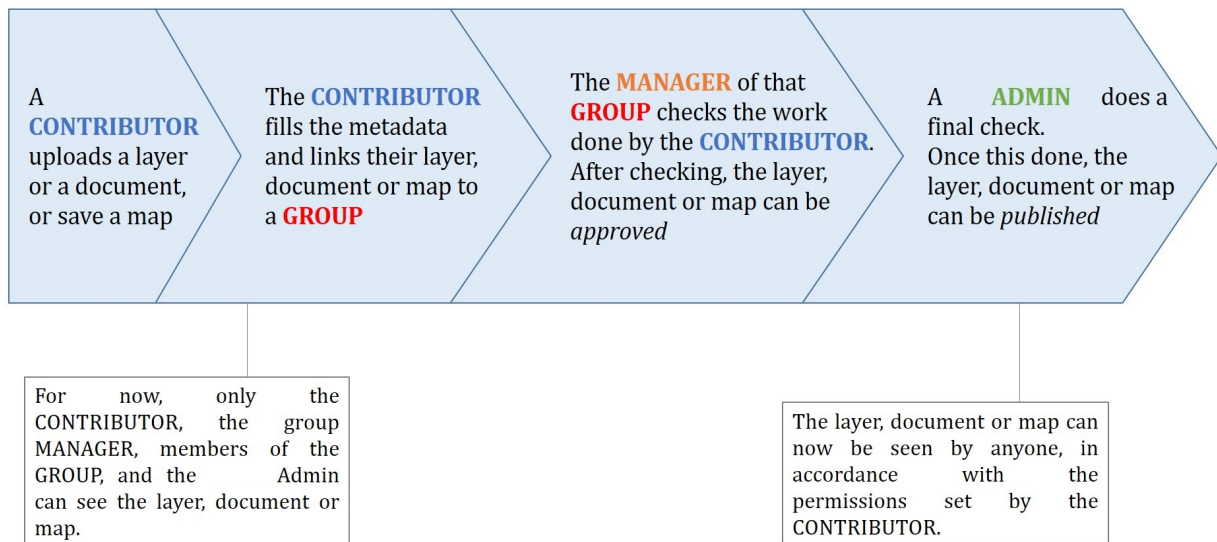


Fig. 343: From upload to publication: the review process on GeoNode

How to enable the advanced workflow

You have to tweak the GeoNode settings accordingly.

Please see the details of the following GeoNode Settings:

- [RESOURCE_PUBLISHING](#)
- [ADMIN_MODERATE_UPLOADS](#)
- [GROUP_PRIVATE_RESOURCES](#)

Summarizing, when all the options above of the Advanced Workflow are enabled, upon a new upload we will have:

- The «unpublished» resources will be **hidden to anonymous users only**. The **registered users** will be still able to access the resources (if they have the rights to do that, of course).
- The «unpublished» resources will remain hidden to users if the permission (see *Admin Guide section: “Manage Permissions”*) will be explicitly removed

- During the upload, whenever the advanced workflow is enabled, the **owner's Groups** are automatically allowed to access the resource, even if the «**anonymous**» flag has been disabled. Those permissions can be removed later on
- During the upload, «**managers**» of the owner's Groups associated to the resource, are always allowed to edit the resource, the same as they are admin for that resource
- «**managers**» of the owner's Groups associated to the resource are allowed to «**publish**» also the resources, not only to «**approve**» them

Change the owner rights in case of advanced workflow is on

After switching `ADMIN_MODERATE_UPLOADS` to `True` and resource is approved owner is no longer able to modify it. He will see new button on the resource detail page: Request change. After clicking this, view with short form is shown. On this view user can write short message why he want to modify the resource.

This message will be sent through messaging and email system to administrators:

After administrator unapprove the resource owner is again able to modify it.

The group Manager approval

Here, the role of the Manager of the group to which your layer, document or map is linked is to check that the uploaded item is correct. Particularly, in the case of a layer or a map, it consists of checking that the chosen cartographic representation and the style are fitting but also that the discretization is appropriate.

The Manager must also check that the metadata are properly completed and that the mandatory information (Title, Abstract, Edition, Keywords, Category, Group, Region) are filled.

If needed, the Manager can contact the contributor responsible of the layer, document or map in order to report potential comments or request clarifications.

Members of the group can also take part in the reviewing process and give some potential inputs to the responsible of the layer, document or map.

When the Manager considers that the layer, document or map is ready to be published, he should approve it. To do so, the Manager goes to the layer, document or map page, then opens the *Wizard* in order to edit the metadata. In the *Settings* tab, the manager checks the *Approved* box, and then updates the metadata and saves the changes:

Fig. 344: *The approbation process of an item by a Manager*

Following this approval, the GeoNode Administrators receive a notification informing them that an item is now waiting for publication

The publication by the GeoNode Administrator

Prior to the public release of an approved layer, a document or a map, the Administrator of the platform performs a final validation of the item and its metadata, notably to check that it is in line with license policies.

If needed, the GeoNode Administrator can contact the Manager who has approved the layer, document or map, as well as its responsible.

Once the layer, document or map is validated, the item is made public by the Administrator. It can now be viewed, accessed, and downloaded in accordance with the `Permissions` set by the responsible contributor.

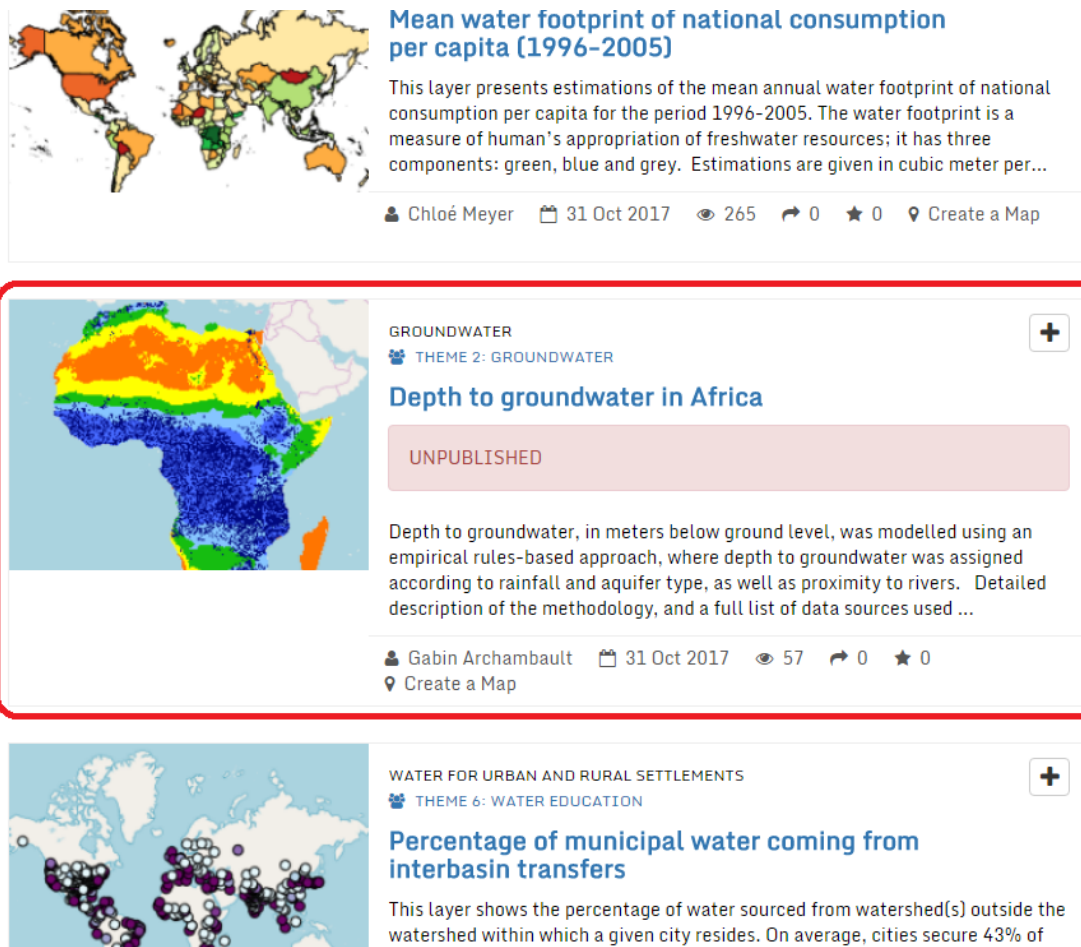


Fig. 345: An approved layer, waiting for publication by the GeoNode administrators

1.24.11 Manage profiles using the admin panel

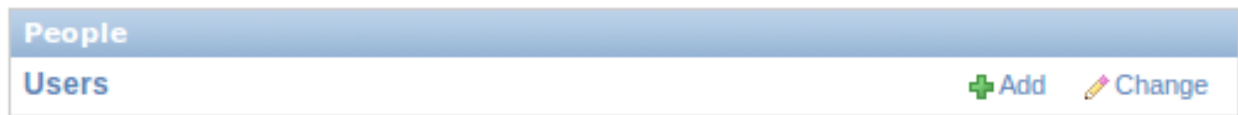
So far GeoNode implements two distinct roles, that can be assigned to resources such as layers, maps or documents:

- party who authored the resource
- party who can be contacted for acquiring knowledge about or acquisition of the resource

These two profiles can be set in the GeoNode interface by accessing the metadata page and setting the `Point of Contact` and `Metadata Author` fields respectively.

Is possible for an administrator to add new roles if needed, by clicking on the *Add Role* button in the *Base -> Contact Roles* section:

Clicking on the *People* section (see figure) will open a web for with some personal information plus a section called *Users*.



Is important that this last section is not modified here unless the administrator is very confident in that operation.

Contact roles		
Resource	Role	Delete?
Add another Contact Role		

1.24.12 Manage layers using the admin panel

Some of the Layers information can be edited directly through the admin interface although the best place is in the *Layer -> Metadata Edit* in GeoNode.

Clicking on the *Admin > Layers* link will show the list of available layers.

Aviso: It is not recommended to modify the Layers' `Attributes` or `Styles` directly from the Admin dashboard unless you are aware of your actions.

The `Metadata` information can be changed for multiple Layers at once through the *Metadata batch edit* action.

By clicking over one Layer link, it will show a detail page allowing you to modify some of the resource info like the metadata, the keywords, the title, etc.

Layers	
Attributes	+ Add ✎ Change
Layers	+ Add ✎ Change
Styles	+ Add ✎ Change
Upload sessions	+ Add ✎ Change

2011 2017

Action: Metadata batch edit Go 2 of 100 selected

☐	ID	
☒	544	_2017_03_15t4_44_53_last

☐ Delete selected layers
☒ Metadata batch edit

Nota: It is strongly recommended to always use the GeoNode *Metadata Wizard* or *Metadata Advanced* tools in order to edit the metadata info.

The Permissions can be changed also for multiple Layers at once through the *Set layers permissions* action.

Django administration

Home › Layers › Layers

Select layer to change

Q Search

◀ 2019 **November 14**

Action: Set Layers Permissions Go 2 of 2 selected

☒	ID	ALTERNATE	TITLE [EN]	DATE	CATE
☒	28	geonode:tasmania_water_bodies	tasmania_water_bodies	Nov. 14, 2019, 3:36 p.m.	---
☒	27	geonode:tasmania_state_boundaries	tasmania_state_boundaries	Nov. 14, 2019, 3:36 p.m.	---

By clicking over one Layer link, it will show a detail page allowing you to modify the permissions for the selected resources.



Layers Permissions

Group

----- ▼

User

----- ▼

Permission Type

☒ Read

☐ Write

☐ Download

Mode

☒ Set

☐ Unset

Cancel Submit

1.24.13 Manage the maps using the admin panel

Similarly to the Layers, it is possible to manage the available GeoNode Maps through the Admin panel also.

Move to *Admin > Maps* to access the Maps list.

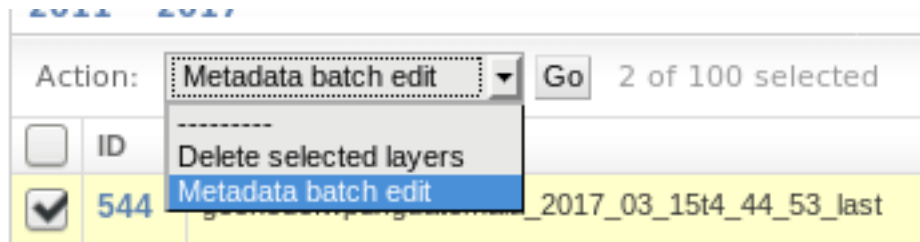
Maps	
Map layers	+ Add ✎ Change
Map snapshots	+ Add ✎ Change
Maps	+ Add ✎ Change

The Metadata information can be changed for multiple Maps at once through the *Metadata batch edit* action.

By clicking over one Map link, it will show a detail page allowing you to modify some of the resource info like the metadata, the keywords, the title, etc.

Nota: It is strongly recommended to always use the GeoNode *Metadata Wizard* or *Metadata Advanced* tools in order to edit the metadata info.

Notice that by enabling the `Featured` option here, will allow GeoNode to show the Map thumbnail and the Map detail link on the *Home Page*



Metadata upload

☐ Metadata uploaded preserve

Metadata xml:

```
<gmd:MD_Metadadata xmlns:gmd="http://www.isotc211.org/2005/gmd"/>
```

Popular count:

2

Share count:

0

☒ Featured

Should this resource be advertised in home page?



☒ Is Published

Should this resource be published and searchable?

☒ Approved

Is this resource validated from a publisher or editor?

Click to search for geospatial data published by other users, organizations and public sources. Download data in standard formats.

[Add layers »](#)



2 Documents

As for the layers and maps GeoNode allows to publish tabular and text data, manage their metadata and associated documents.

[Add documents »](#)

Data is available for browsing, aggregating and styling to generate maps which can be saved, downloaded, shared publicly or restricted to specify users only.

[Create maps »](#)

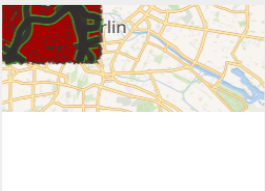


22 Users

Geonode allows registered users to easily upload geospatial data and various documents in several formats.

[See users »](#)

[Explore all datasets](#)



My Map

Data
Layers
Documents
Remote Feeds

Maps
Explore Maps
Create Map

About
People
Groups
Announcements

Contact Us
Mr. Pibody

1.24.14 Manage the documents using the admin panel

Similarly to the Layers and Maps, it is possible to manage the available GeoNode Documents through the Admin panel also.

Move to *Admin > Documents* to access the Documents list.

Documents

Documents [+ Add](#) [Change](#)

The Metadata information can be changed for multiple Documents at once through the *Metadata batch edit* action.

Action: **Metadata batch edit** [Go](#) 2 of 100 selected

☐	ID	
☒	544	_2017_03_15t4_44_53_last

Metadata batch edit

By clicking over one Document link, it will show a detail page allowing you to modify some of the resource info like the metadata, the keywords, the title, etc.

Nota: It is strongly recommended to always use the GeoNode *Metadata Wizard* or *Metadata Advanced* tools in order to edit the metadata info.

1.24.15 Manage the base metadata choices using the admin panel

Admin > Base contains almost all the objects you need to populate the resources metadata choices.

In other words the options available from the *select-boxes* of the *Metadata Wizard* and *Metadata Advanced* panels.

Nota: When editing the resource metadata through the *Metadata Wizard*, some fields are marked as `mandatory` and by filling those information the `Completeness` progress will advance accordingly.

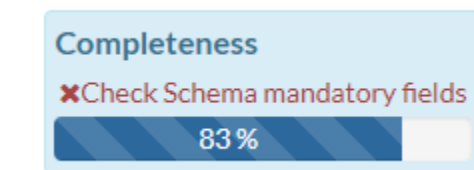


Fig. 349: *Metadata Completeness*

Even if not all the fields have been filled, the system won't prevent you to update the metadata; this is why the `Mandatory` fields are mandatory to be fully compliant with an ISO 19115 metadata schema, but are only recommended to be compliant with GeoNode.

Also the `Completeness` indicates how far the metadata is to be compliant with an ISO 19115 metadata schema.

Of course, it is **highly** recommended to always fill as much as possible at least all the metadata fields marked as `Mandatory`.

This will improve not only the quality of the data stored into the system, but will help the users to easily search for them on GeoNode.

All the `Search & Filter` panels and options of GeoNode are, in fact, based on the resources metadata fields. Too much generic descriptions and too empty metadata fields, will give highly un-precise and very wide search results to the users.

Hierarchical keywords

Through the *Admin > Base > Hierarchical keywords* panel it will be possible to manage all the keywords associated to the resources.

- The *Name* is the human readable text of the keyword, what users will see.
- The *Slug* is a unique label used by the system to identify the keyword; most of the times it is equal to the name.

Notice that through the *Position* and *Relative to* selectors, it is possible to establish a hierarchy between the available keywords. The hierarchy will be reflected in the form of a tree from the metadata panels.

By default each user with editing metadata rights on any resource, will be able to insert new keywords into the system by simply typing a free text on the keywords metadata field.

Django administration

Site administration

ACCOUNTS		
Email addresses	+ Add	Change

ACTSTREAM		
Actions	+ Add	Change
Follows	+ Add	Change

ANNOUNCEMENTS		
Announcements	+ Add	Change
Dismissals	+ Add	Change

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change

AVATAR		
Avatars	+ Add	Change

BASE		
Backups	+ Add	Change
Contact roles	+ Add	Change
Hierarchical keywords	+ Add	Change
Licenses	+ Add	Change
Links	+ Add	Change
Menu items	+ Add	Change
Menu placeholders	+ Add	Change
Menus	+ Add	Change
Metadata Regions	+ Add	Change
Metadata Restriction Code Types		Change
Metadata Spatial Representation Types		Change
Metadata Topic Categories	+ Add	Change

DIALOGOS		
Comments	+ Add	Change

Recent actions

My actions

- [Warning dear users](#)
Announcement
- [afabiani](#)
User
- [afabiani2](#)
User
- [John Smith Foundation Team](#)
Group profile
- [Warning dear users](#)
Announcement
- [Warning dear users](#)
Announcement
- [Warning dear users](#)
Announcement
- [Warning dear users](#)
Announcement
- [Transportation Planners](#)
Group profile
- [Dismissal object](#)
Dismissal

Fig. 346: Admin dashboard Base Panel

Metadata for places

Completeness
 ✖ Check Schema mandatory fields
 83%

Edit Preview Settings

Mandatory
Mandatory
Optional

1

Basic Metadata

2

Location and Licenses

3

Optional Metadata

4

Dataset Attributes

Language ?

English

License ?

Not Specified

 NextView
 Not Specified
 Open Data Commons Open Database License / OSM
Public Domain
 Public Domain / USG
 Varied / Derived
 Varied / Original

Regions

× Global

Data quality statement ?

General explanation of the data producer's knowledge about the lineage of a dataset

 Field declared Mandatory by the Metadata Schema

Restrictions ?

 * Field declared Mandatory by the Metadata Schema

Restrictions other ?

other restrictions and legal prerequisites for accessing and using the resource or metadata

Return to Layer
<< Back
Update
Next >>

Fig. 347: Metadata Wizard Panel

It is possible to force the user to select from a fixed list of keywords through the `FREE-TEXT_KEYWORDS_READONLY` setting.

When set to *True* keywords won't be writable from users anymore. Only admins can will be able to manage them through the *Admin > Base > Hierarchical keywords* panel.

Licenses

Through the *Admin > Base > Licenses* panel it will be possible to manage all the licenses associated to the resources.

The license description and the info URL will be shown on the resource detail page.

The license text will be shown on the catalogue metadata XML documents.

Aviso: It is **strongly** recommended to not publish resources without an appropriate license. Always make sure the data provider specifies the correct license and that all the restrictions have been honored.

Maintenance frequency

----- ▼

Free-text Keywords

features ×

ne_10m_populated_places_simple ×

buildings

builtup_area

catastro

catastro_sc

climate outlook 40 June July and August

Regions

× Global

Restrictions

----- ▼

Restrictions other

other restrictions and legal prerequisites for
accessing and using the resource or metadata

Fig. 348: Metadata Advanced Panel



Fig. 350: Hierarchical keywords list

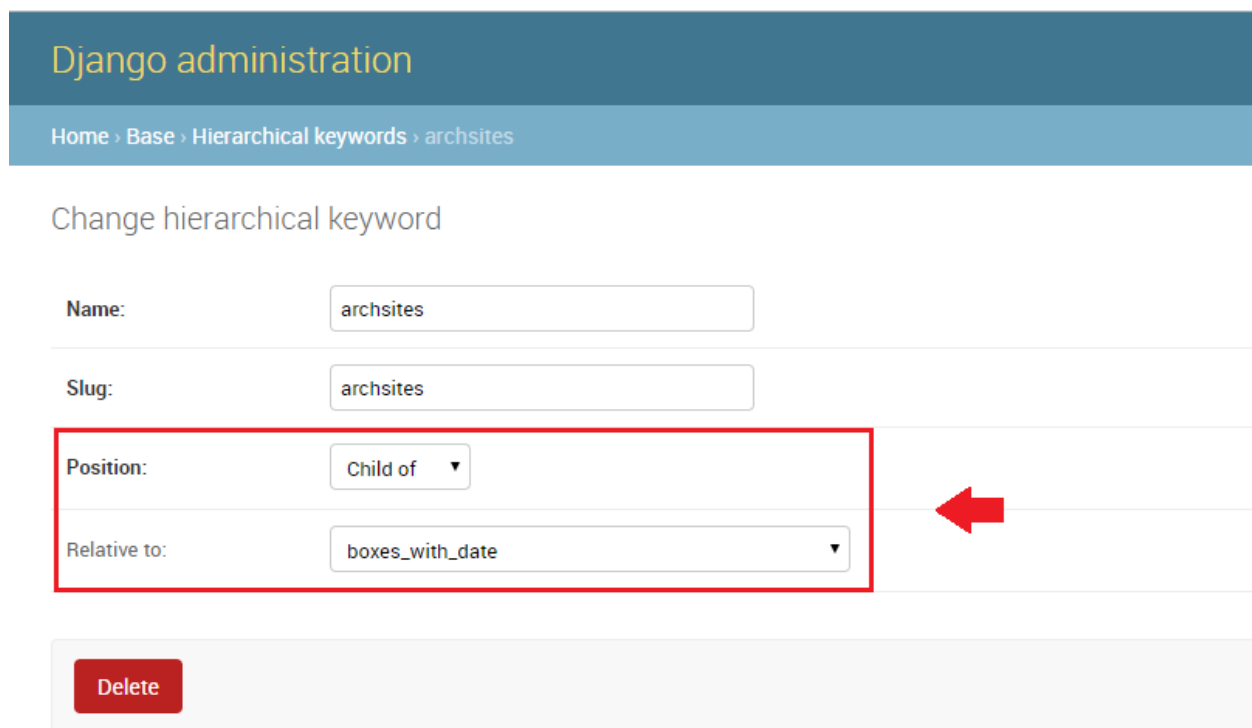


Fig. 351: Hierarchical keywords edit

[Edit](#) [Preview](#) [Settings](#)

Mandatory **Mandatory**

1 2

Basic Metadata **Location and Licenses**

Language ?

English

License ?

Open Data Commons Open Database License / OSM

NextView

Not Specified

Open Data Commons Open Database License / OSM

Public Domain

Public Domain / USG

Varied / Derived

Varied / Original

Regions

× Global

Data quality statement ?

General explanation of the d
knowledge about the lineag

Field declared Mandatory by the Metadat

Fig. 352: Metadata editor Licenses

Fig. 353: *Resource detail License*

```
<?xml-stylesheet href="http://www.isotc211.org/2005/resources/CodeList/gmxCodeLists.xml" type="text/xml" />
<gmd:ID_Keys>
  </gmd:ID_Keys>
  <gmd:descriptiveKeywords>
    <gmd:resourceConstraints>
      <gmd:ID_LegalConstraints>
        <gmd:useConstraints>
          <gmd:ID_RestrictionCode codeList="http://www.isotc211.org/2005/resources/CodeList/gmxCodeLists.xml#ID_RestrictionCode" codeListValue="license" codeSpace="ISOTC211/19115" license="<gmd:ID_RestrictionCode>
            <gmd:useConstraints>
              <gmd:otherConstraints>
                <gco:CharacterString>
                  Public Domain (PD): Works in the public domain may be used freely without the permission of the former copyright owner. (http://www.copyright.gov/help/faq/faq-definitions.html)
                </gco:CharacterString>
              </gmd:otherConstraints>
            </gmd:ID_RestrictionCode>
          </gmd:ID_LegalConstraints>
        </gmd:resourceConstraints>
      </gmd:ID_LegalConstraints>
      <gmd:ID_RestrictionCode codeList="http://www.isotc211.org/2005/resources/CodeList/gmxCodeLists.xml#ID_RestrictionCode" codeListValue="" codeSpace="ISOTC211/19115"/>
    </gmd:useConstraints>
  </gmd:ID_RestrictionCode>

```

Fig. 354: *Resource Metadata ISO License*

Metadata Regions

Through the *Admin > Base > Metadata Regions* panel it will be possible to manage all the admin areas associated to the resources.

Fig. 355: Resource Metadata Regions

Notice that those regions are used by GeoNode to filter search results also through the resource list view.

Nota: GeoNode tries to guess the Regions intersecting the data bounding boxes when uploading a new layer. Those should be refined by the user layer on anyway.

Metadata Restriction Code Types and Spatial Representation Types

Through the *Admin > Base > Metadata Restriction Code Types* and *Admin > Base > Metadata Spatial Representation Types* panels, it will be possible to **update only** the metadata descriptions for restrictions and spatial representation types.

Such lists are *read-only* by default since they have been associated to the specific codes of the ISO 19115 metadata schema. Changing them would require the system to provide a custom dictionary through the metadata catalog too. Such functionality is not supported actually by GeoNode.

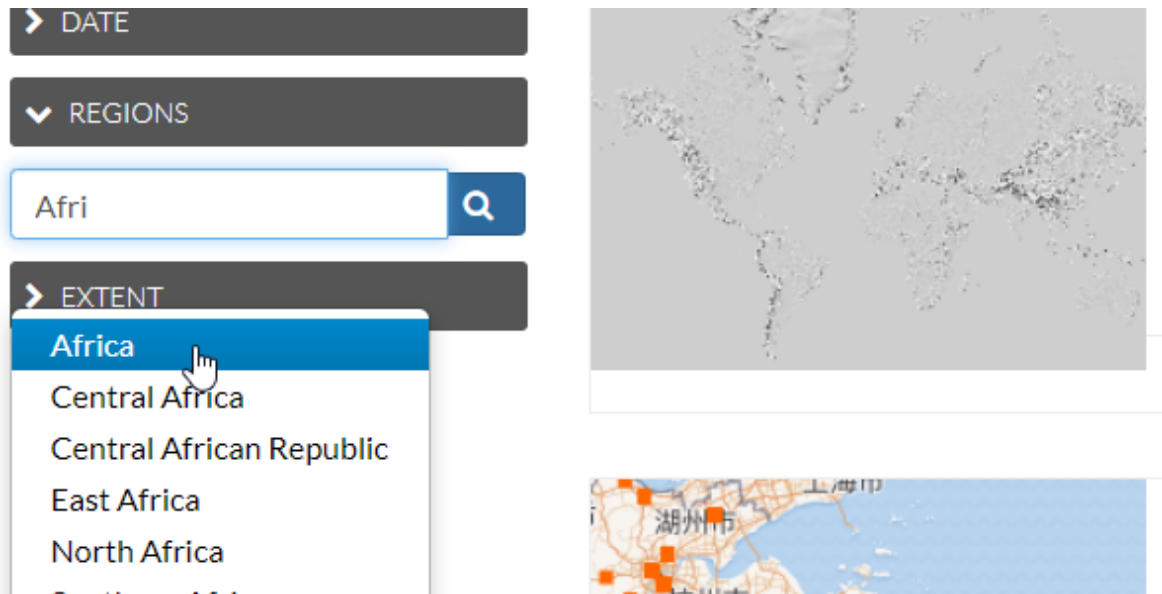


Fig. 356: GeoNode filtering by Metadata Regions

Metadata Topic Categories

Through the *Admin > Base > Metadata Topic Categories* panel it will be possible to manage all the resource metadata categories available into the system.

Notice that by default, GeoNode provides the standard topic categories available with the ISO 19115 metadata schema. Changing them means that the system won't be compliant with the standard ISO 19115 metadata schema anymore. ISO 19115 metadata schema extensions are not currently supported natively by GeoNode.

It is worth notice that GeoNode allows you to associate [Font Awesome Icons](#) to each topic category through their `fa-icon` code. Those icons will be used by GeoNode to represent the topic category on both the *Search & Filter* menus and *Metadata* panels.

Aviso: The list of the Metadata Topic Categories on the home page is currently fixed. To change it you will need to update or override the GeoNode `index.html` HTML template.

By default the Metadata Topic Categories are *writable*. Meaning that they can be removed or created by the *Admin* panel.

It is possible to make them fixed (it will be possible to update their descriptions and icons only) through the `MODIFY_TOPICCATEGORY` setting.

1.24.16 Announcements

As an Administrator you might need to broadcast announcements to the world about your portal or simply to the internal contributors.

GeoNode `Announcements` allow actually to do that; an admin has the possibility to create three types of messages, accordingly to their severity, decide their validity in terms of time period (start date and expiring date of the announcement), who can view them or not (everyone or just the registered members) and whenever a user can hide the message or not and how long.

A GeoNode announcement actually looks like this:

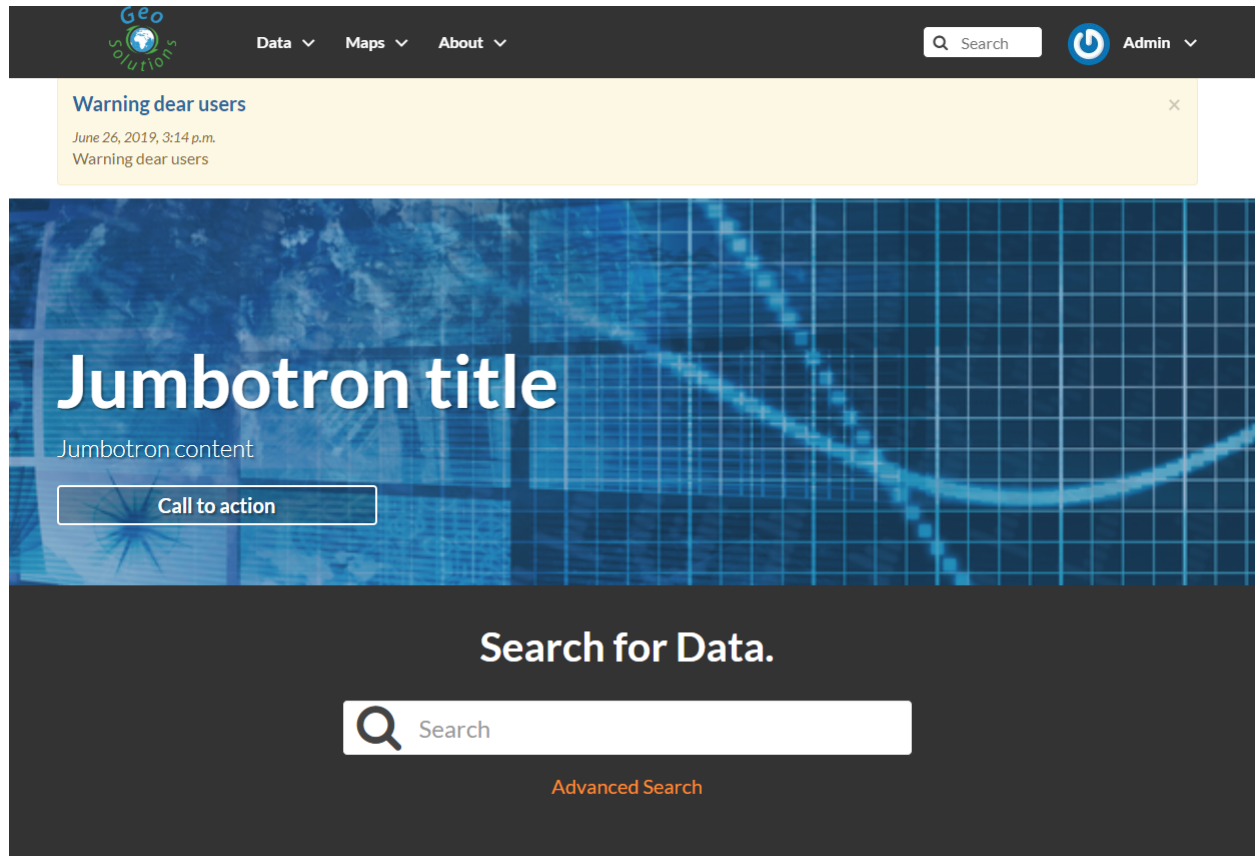


Fig. 357: A sample Warning Announcement

There are three types of announcements accordingly to their severity level: General, Warning and Critical. The difference is mainly the color of the announcement box.

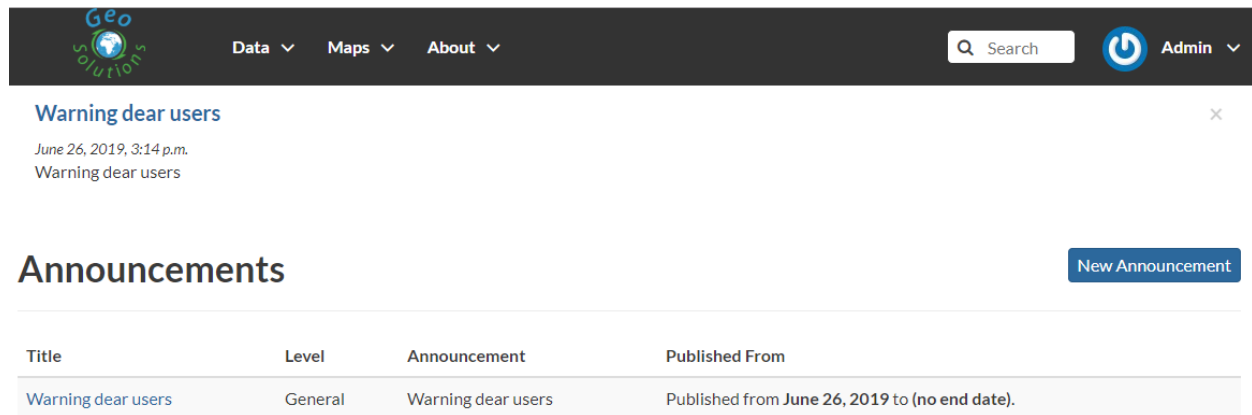
Only administrators and staff members can create and manage announcements.

Currently there are two ways to access and manage the announcements list:

1. Via the GeoNode interface, from the *Profile* panel

Nota: Those are accessible by both admins and staff members.

2. Via the GeoNode *Admin* panel



Warning dear users

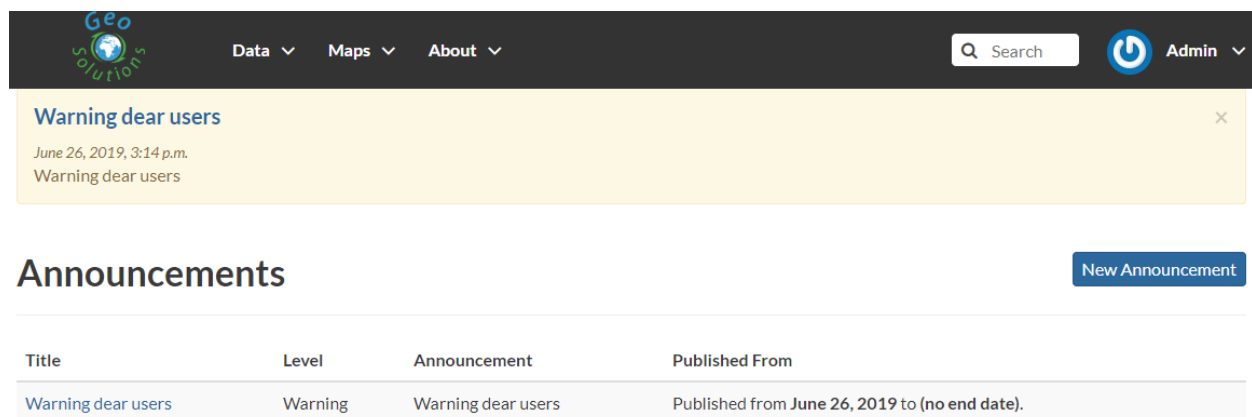
June 26, 2019, 3:14 p.m.
Warning dear users

Announcements

New Announcement

Title	Level	Announcement	Published From
Warning dear users	General	Warning dear users	Published from June 26, 2019 to (no end date).

Fig. 358: General Announcement



Warning dear users

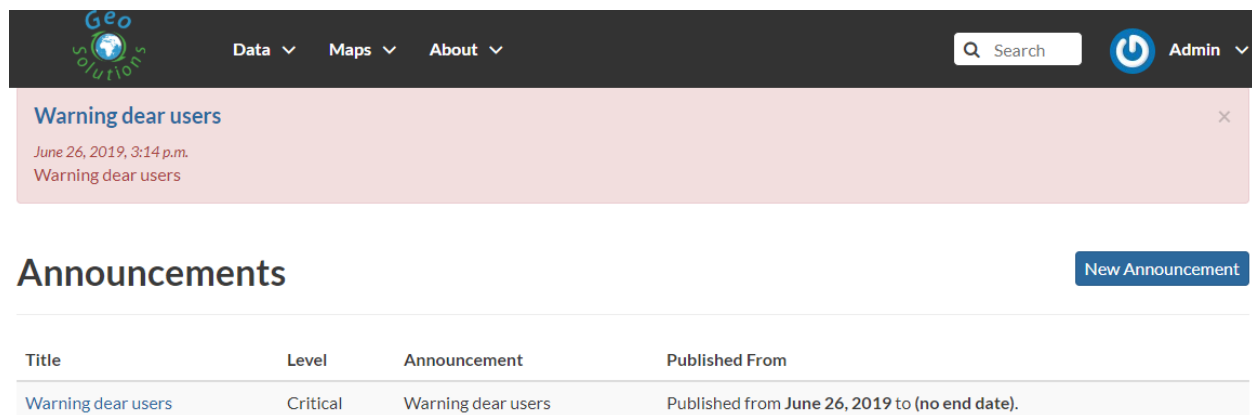
June 26, 2019, 3:14 p.m.
Warning dear users

Announcements

New Announcement

Title	Level	Announcement	Published From
Warning dear users	Warning	Warning dear users	Published from June 26, 2019 to (no end date).

Fig. 359: Warning Announcement



Warning dear users

June 26, 2019, 3:14 p.m.
Warning dear users

Announcements

New Announcement

Title	Level	Announcement	Published From
Warning dear users	Critical	Warning dear users	Published from June 26, 2019 to (no end date).

Fig. 360: Critical Announcement

Admin (admin)

Admin

Position	Not provided.
Organization	GeoSolutions
Location	56017 PI ITA
Voice	Not provided.
Fax	Not provided.
Description	Not provided.
Keywords	Not provided

User layers WMS GetCapabilities document

- Message User
- Edit profile
- Connected social accounts
- Associated e-mails
- Set/Change password
- Upload new layers
- Create a new layer
- Create a new map
- My Activities
- Favorites
- Notifications
- Announcements**
- Invite Users
- GeoServer
- Admin

Fig. 361: *Announcements from the Profile panel*

Nota: Those are accessible by admins only.

The functionalities are almost the same for both the interfaces, except that from the *Admin* panel it is possible to manage the dismissals too.

Dismissals are basically records of members that have read the announcement and closed the message box. An announcement can have one `dismissal` type among the three below:

1. *No Dismissal Allowed* it won't be possible to close the announcement's message box at all.
2. *Session Only Dismissal* (*) the default one, it will be possible to close the announcement's message box for the current browser session. It will show up again at next access.
3. *Permanent Dismissal Allowed* once the announcement's message box is closed, it won't appear again for the current member.

How to create and manage Announcements

From the *Profile* panel, click on `Announcements` link

Click either on *New Announcement* to create a new one or over a title of an existing one to manage its contents.

Create a new announcement is quite straight; you have to fill the fields provided by the form.

Aviso: In order to be visible, you will need to check the *Site wide* option **in any case**. You might want to hide the message to *anonymous* users by enabling the *Members only* option too.

Managing announcements from the *Admin* panel, is basically the same; the fields for the form will be exactly the same.

Accessing announcements options from the *Admin* panel, allows you to manage dismissals also. Through this interface you will be able to selectively decide members which can or cannot view a specific announcement, or force them to visualize the messages again by deleting the dismissals accordingly.

1.24.17 Menus, Items and Placeholders

GeoNode provides some integrated functionalities allowing you to quickly and easily customize the top-bar menu (see the example below).

With minor changes of the `basic.html` template, potentially, it could be possible to use the same approach for a more complex customization. Let's start with the simple one.

By default GeoNode provides a custom placeholder already defined into the `basic.html` template, called `TOPBAR_MENU`

```
...
<ul class="nav navbar-nav navbar-right">

    {% block my_extra_right_tab %}

        {% render_nav_menu 'TOPBAR_MENU' %}

    {% endblock my_extra_right_tab %}

</li>
```

(continues on next page)

Django administration

Site administration

The screenshot shows the Django administration interface. The 'ANNOUNCEMENTS' section is highlighted with a red box. It contains two sub-items: 'Announcements' and 'Dismissals', each with '+ Add' and 'Change' links. Other sections visible include 'ACCOUNTS' (Email addresses), 'ACTSTREAM' (Actions, Follows), 'AUTHENTICATION AND AUTHORIZATION' (Groups), and 'AVATAR' (Avatars).

Fig. 362: *Announcements from the Admin panel*

Announcements



New Announcement

Title	Level	Announcement	Published From
Warning dear users	Critical	Warning dear users	Published from June 26, 2019 to (no end date).



Fig. 363: *Announcements List from the Profile panel*

Create Announcement

Title

Level

General ▼

Content

☐ Site wide
☐ Members only

Dismissal type

Session Only Dismissal ▼

Publish_start

2019-06-27 13:24:04

Publish_end

Fig. 364: Create Announcement from the Profile panel

The screenshot shows the 'Change announcement' form in the Django admin panel. The form includes fields for Title (set to 'Warning dear users'), Level (set to 'Critical'), and Content (a text area with 'Warning dear users'). There are checkboxes for 'Site wide' and 'Members only'. The 'Publish_start' section has date and time pickers (Date: 2019-06-26, Time: 15:12:57). The 'Publish_end' section has similar pickers. The 'Dismissal type' is set to 'Session Only Dismissal'. At the bottom, there are buttons for 'Delete', 'Save and add another', 'Save and continue editing', and 'SAVE'.

Fig. 365: Create Announcement from the Admin panel

The screenshot shows the 'Add dismissal' form in the Django admin panel. It includes a 'User' dropdown (set to 'afabiani'), an 'Announcement' dropdown (set to 'Warning dear users'), and a 'Dismissed at' section with date and time pickers (Date: 2019-06-27, Time: 13:30:56). At the bottom, there are buttons for 'Save and add another', 'Save and continue editing', and 'SAVE'.

Fig. 366: Create Dismissal from the Admin panel

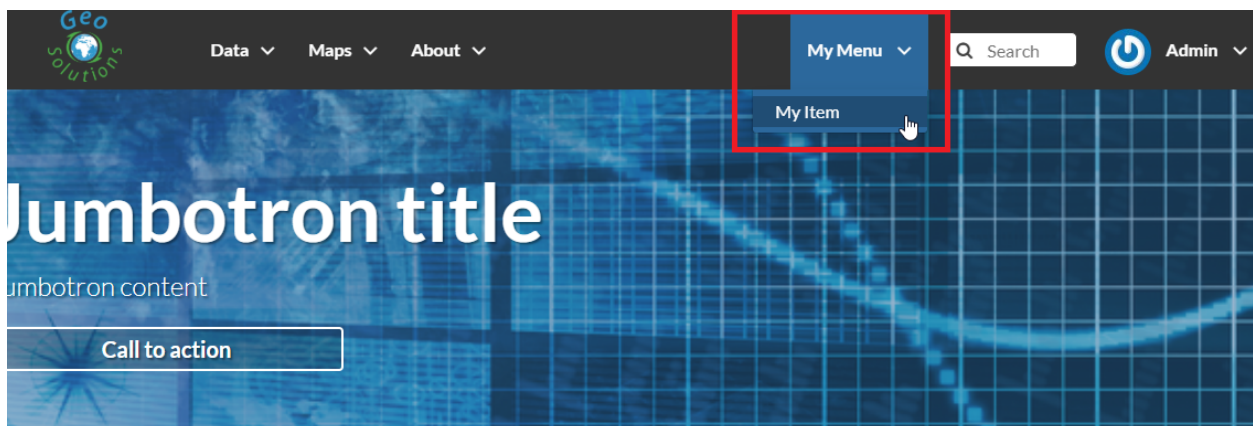


Fig. 367: GeoNode Top-Bar Menu customization

(continuação da página anterior)

```

<div class="search">
  <form id="search" action="{% url 'search' %}" >
    <span class="fa fa-search"></span>
    {% if HAYSTACK_SEARCH %}
    <input id="search_input" type="text" placeholder="{% trans 'Search' %}"
↪name="q">
      {% else %}
      <input id="search_input" type="text" placeholder="{% trans 'Search' %}"
↪name="title__icontains">
      {% endif %}
    </form>
  </div>
</li>
...

```

From the *Admin > Base* panel, it is possible to access to the Menu, Menu Items and Menu Placeholder options.

BASE		
Backups	+ Add	Change
Contact roles	+ Add	Change
Hierarchical keywords	+ Add	Change
Licenses	+ Add	Change
Links	+ Add	Change
Menu items	+ Add	Change
Menu placeholders	+ Add	Change
Menus	+ Add	Change
Metadata Regions	+ Add	Change
Metadata Restriction Code Types		Change
Metadata Spatial Representation Types		Change
Metadata Topic Categories	+ Add	Change
DIALOGOS		

Fig. 368: Menu, Menu Items and Menu Placeholder options on the Admin panel

The hierarchical structure of a custom Menu is the following one:

1. **Menu Placeholder**; first of all you need to define a *placeholder* both into the *Admin > Base* panel and the `basic.html`.

By default GeoNode provides an already defined one called `TOPBAR_MENU`

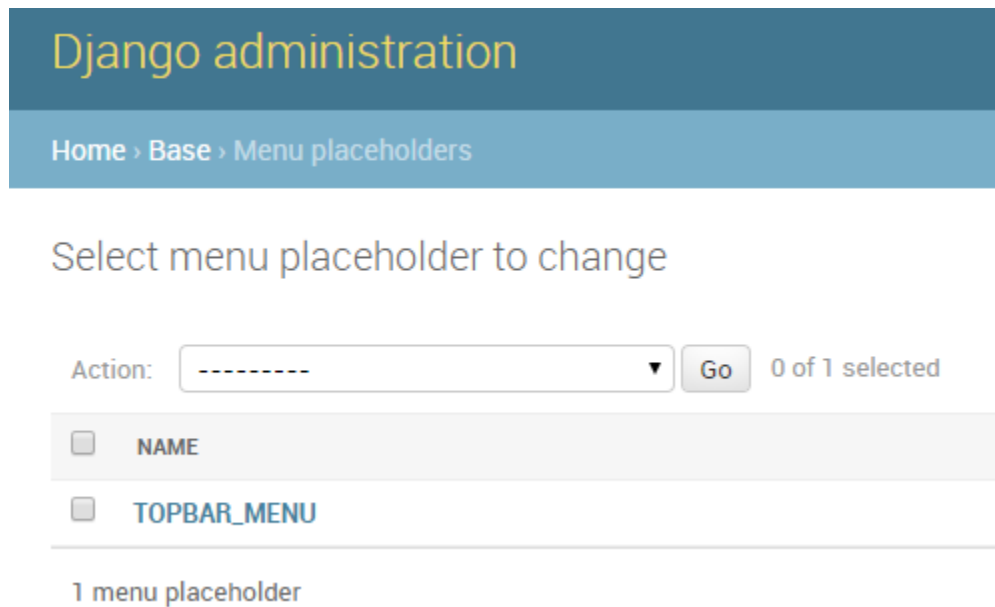


Fig. 369: The default `TOPBAR_MENU` Menu Placeholder on the Admin panel

2. **Menu;** second thing to do is to create a new *menu* associated to the corresponding *placeholder*. This is still possible from the *Admin > Base* panel

You will need to provide:

- A `Title`, representing the name of the `Menu` visible by the users

Aviso: By using this approach, internationalization won't be supported. For the time being GeoNode does not support this for menus created from the *Admin > Base* panel.

- A `Menu Placeholder` from the existing ones.
- A `Order` in the case you'll create more menus associated to the same placeholder.

3. `Menu Item`; finally you will need to create voices belonging to the *menu*. For the time being, GeoNode allows you to create only `href` links.

Aviso: The `Menu` won't be visible until you add at least one `Menu Item`

Django administration

Home › Base › Menus › My Menu

Change menu

Title:

My Menu

Placeholder:

TOPBAR_MENU ▼  

Order:

1

Delete

Fig. 370: Create a new Menu from the Admin panel

Django administration

Home › Base › Menu items › My Item

Change menu item

Title:	My Item
Menu:	My Menu ▾  
Order:	1
☒ Blank target	
Url:	https://dev.geonode.geo-solutions.it/

Delete

Fig. 371: Create a new Menu Item from the Admin panel

1.24.18 OAuth2 Access Tokens

This small section won't cover entirely the GeoNode OAuth2 security integration, this is explained in detail in other sections of the documentation (refer to *OAuth2 Fixtures Update and Base URL Migration* and *OAuth2 Tokens and Sessions*).

Here we will focus mainly on the *Admin > DJANGO/GEONODE OAUTH TOOLKIT* panel items with a specific attention to the `Access tokens` management.

The *Admin > DJANGO/GEONODE OAUTH TOOLKIT* panel (as shown in the figure below) allows an admin to manage everything related to GeoNode OAuth2 grants and permissions.

As better explained in other sections of the documentation, this is needed to correctly handle the communication between GeoNode and GeoServer.

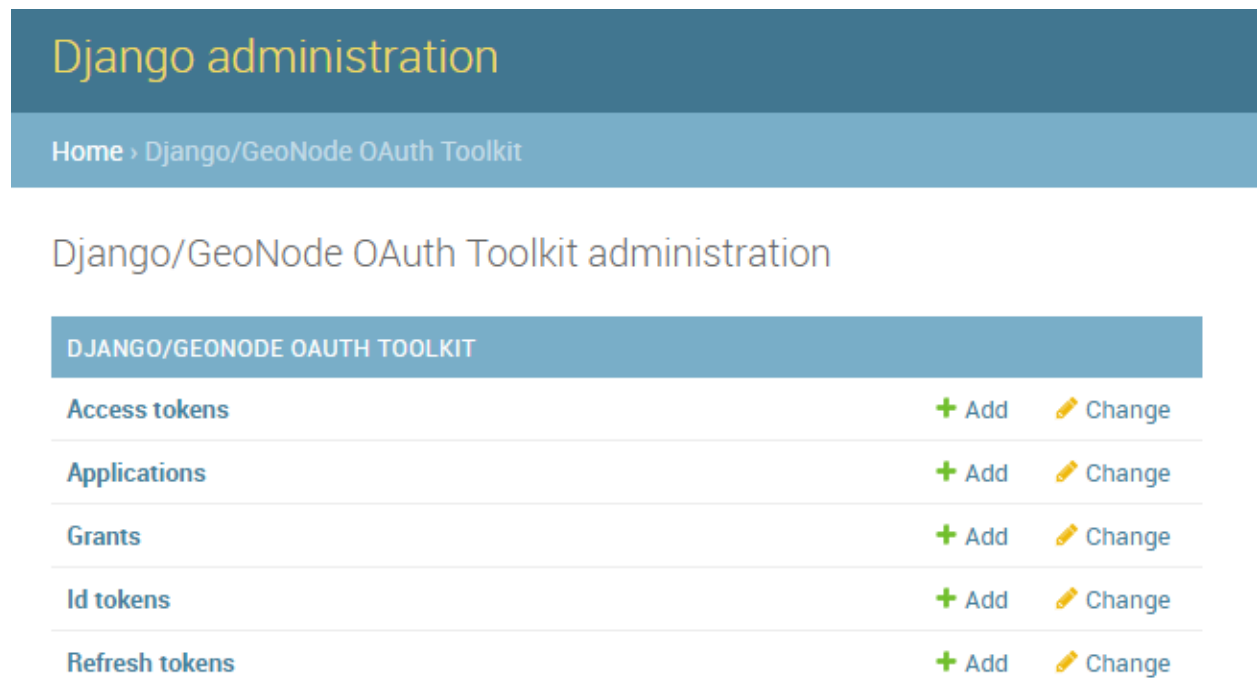


Fig. 372: *DJANGO/GEONODE OAUTH TOOLKIT Admin panel*

Specifically from this panel an admin can create, delete or extend OAuth2 `Access tokens`.

The section *OAuth2 Tokens and Sessions* better explains the concepts behind OAuth2 sessions; we want just to refresh the mind here about the basic concepts:

- If the `SESSION_EXPIRED_CONTROL_ENABLED` setting is set to *True* (by default it is set to *True*) a registered user cannot login to neither GeoNode nor GeoServer without a valid `Access token`.
- When logging-in into GeoNode through the sign-up form, GeoNode checks if a valid `Access token` exists and it creates a new one if not, or extends the existing one if expired.
- New `Access tokens` expire automatically after `ACCESS_TOKEN_EXPIRE_SECONDS` setting (by default 86400)
- When an `Access token` expires, the user will be kicked out from the session and forced to login again

Create a new token or extend an existing one

It is possible from the *Admin > DJANGO/GEONODE OAUTH TOOLKIT* panel to create a new `Access token` for a user.

In order to do that, just click on the *Add* button beside `Access tokens` topic

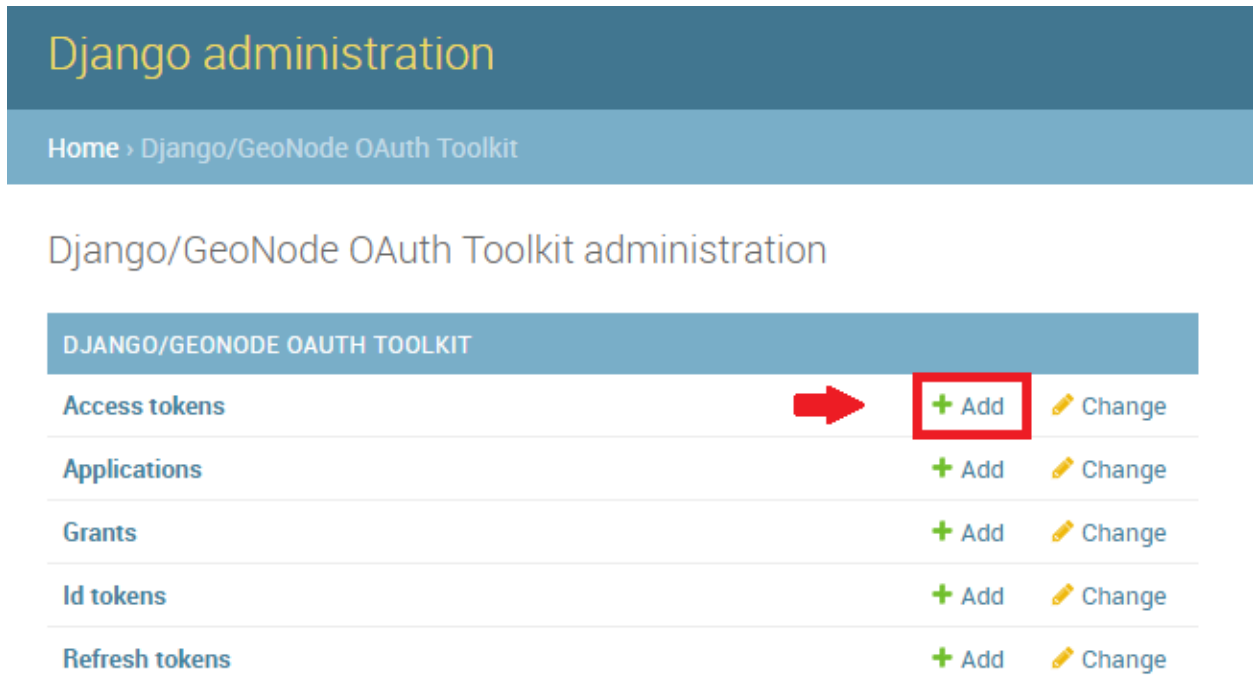


Fig. 373: Add a new ``Access token``

On the new form

select the followings:

1. `User`; use the search tool in order to select the correct user. The form want the user PK, which is a number, and **not** the username. The search tool will do everything for you.
2. `Source refresh token`; this is not mandatory, leave it blank.
3. `Token`; write here any alphanumeric string. This will be the `access_token` that the member can use to access the OWS services. We suggest to use a service like <https://passwordsgenerator.net/> in order to generate a strong token string.
4. `Application`; select **GeoServer**, this is mandatory
5. `Expires`; select an expiration date by using the *date-time* widgets.
6. `Scope`; select **write**, this is mandatory.

Django administration

Home › Django/GeoNode OAuth Toolkit › Access tokens › Add access token

Add access token

User:

Q

Source refresh token:

▼

Token:

Application:

▼

Expires:

Date:

Today | 

Time:

Now | 

Note: You are 2 hours ahead of server time.

Scope:

Fig. 374: Create an ``Access token``

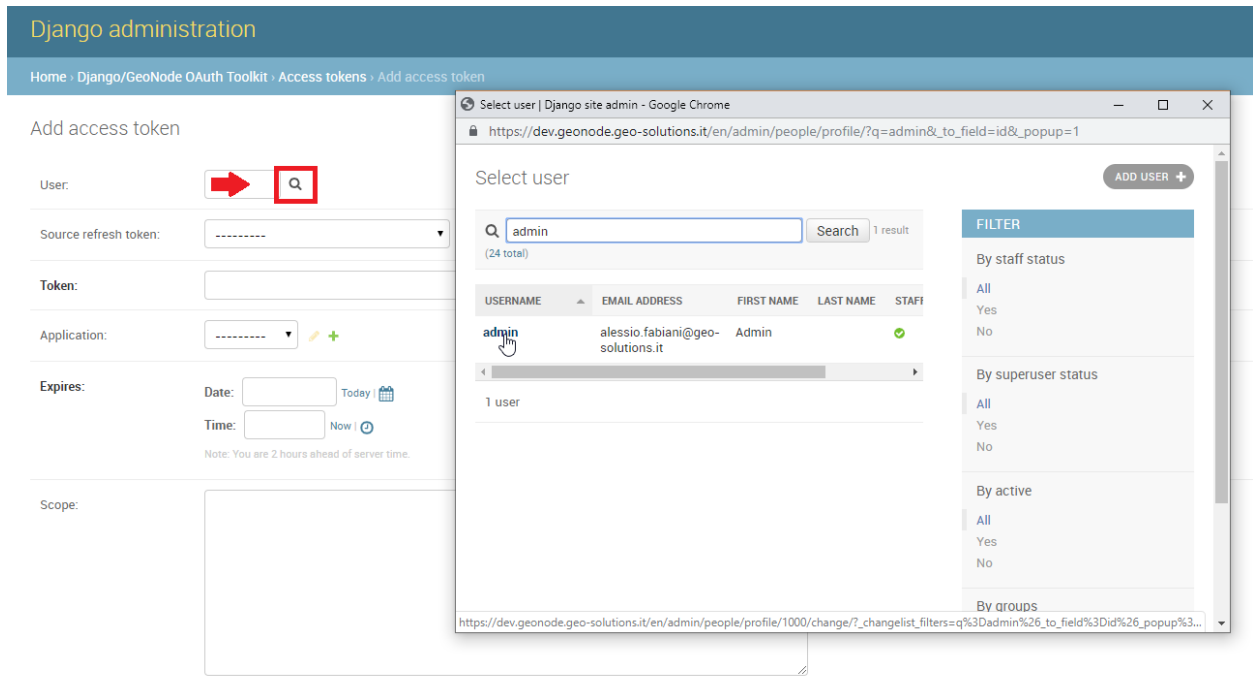


Fig. 375: Select a User

User:

Source refresh token:

Token:

Application:

Expires: Date: Today

Time: Now

Fig. 376: Select a Token

Add access token

User: 

Source refresh token:   

Token:

Application:   

Expires:   

Fig. 377: Select the GeoServer Application

Source refresh token:   

Token:

Application:  

Expires:

Date:  Today |

Time:  Now |

Note: You are 2 hours ahead of server time.



Fig. 378: Select the Token Expiration

Application: GeoServer  

Expires:

Date: 2029-06-30 Today 

Time: 11:03:39 Now 

Note: You are 2 hours ahead of server time.

Scope: write 



Fig. 379: Select the Application Scope

Do not forget to *Save*.

From now on, GeoNode will use this `Access Token` to control the user session (notice that the user need to login again if closing the browser session), and the user will be able to access the OWS Services by using the new `Access Token`, e.g.:

```
https://dev.geonode.geo-solutions.it/geoserver/ows?service=wms&version=1.3.0&
↪request=GetCapabilities&access_token=123456
```

Notice the `...quest=GetCapabilities&access_token=123456` (**access_token**) parameter at the end of the URL.

Force a User Session to expire

Everything said about the creation of a new `Access Token`, applies to the deletion of the latter.

From the same interface an admin can either select an expiration date or delete all the `Access Tokens` associated to a user, in order to force its session to expire.

Remember that the user could activate another session by logging-in again on GeoNode with its credentials.

In order to be sure the user won't force GeoNode to refresh the token, reset first its password or de-activate it.

1.25 GeoNode Management Commands

1.25.1 Migrate GeoNode Base URL

The `migrate_baseurl Management Command` allows you to fix all the GeoNode Links whenever, for some reason, you need to change the *Domain Name* of *IP Address* of GeoNode.

This **must** be used also in the cases you'll need to change the network schema from HTTP to HTTPS, as an instance.

First of all let's take a look at the `--help` option of the `migrate_baseurl` management command in order to inspect all the command options and features.

Run

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py migrate_baseurl --help
```

Nota: If you enabled `local_settings.py` the command will change as following:

```
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py migrate_baseurl --help
```

This will produce output that looks like the following

```
usage: manage.py migrate_baseurl [-h] [--version] [-v {0,1,2,3}]
                                [--settings SETTINGS]
                                [--pythonpath PYTHONPATH] [--traceback]
                                [--no-color] [-f]
                                [--source-address SOURCE_ADDRESS]
                                [--target-address TARGET_ADDRESS]

Migrate GeoNode VM Base URL

optional arguments:
-h, --help            show this help message and exit
--version            show program's version number and exit
-v {0,1,2,3}, --verbosity {0,1,2,3}
                    Verbosity level; 0=minimal output, 1=normal output,
                    2=verbose output, 3=very verbose output
--settings SETTINGS  The Python path to a settings module, e.g.
                    "myproject.settings.main". If this isn't provided, the
                    DJANGO_SETTINGS_MODULE environment variable will be
                    used.
--pythonpath PYTHONPATH
                    A directory to add to the Python path, e.g.
                    "/home/djangoprojects/myproject".
--traceback          Raise on CommandError exceptions
--no-color           Don't colorize the command output.
-f, --force          Forces the execution without asking for confirmation.
--source-address SOURCE_ADDRESS
                    Source Address (the one currently on DB e.g.
                    http://192.168.1.23)
--target-address TARGET_ADDRESS
                    Target Address (the one to be changed e.g. http://my-
                    public.geonode.org)
```

- **Example 1:** I want to move my GeoNode instance from `http://127.0.0.1` to `http://example.org`

Aviso: Make always sure you are using the **correct** settings

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py migrate_baseurl_
↪--source-address=127.0.0.1 --target-address=example.org
```

- **Example 2:** I want to move my GeoNode instance from `http://example.org` to `https://example.org`

Aviso: Make always sure you are using the **correct** settings

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py migrate_baseurl_
→--source-address=http:\\example.org --target-address=https:\\example.
→org
```

- **Example 3:** I want to move my GeoNode instance from `https:\\example.org` to `https:\\geonode.example.org`

Aviso: Make always sure you are using the **correct** settings

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py migrate_baseurl_
→--source-address=example.org --target-address=geonode.example.org
```

Nota: After migrating the base URL, make sure to sanitize the links and catalog metadata also ([Update Permissions, Metadata, Legends and Download Links](#)).

1.25.2 Update Permissions, Metadata, Legends and Download Links

The following three utility *Management Commands*, allow to fixup:

1. *Users/Groups Permissions on Layers*; those will be refreshed and synchronized with the *GIS Server* ones also
2. *Metadata, Legend and Download links on Layers and Maps*
3. Cleanup *Duplicated Links* and *Outdated Thumbnails*

Management Command `sync_geonode_layers`

This command allows to sync already existing permissions on Layers. In order to change/set Layers' permissions refer to the section [Batch Sync Permissions](#)

The options are:

- **filter**; Only update data the layer names that match the given filter.
- **username**; Only update data owned by the specified username.
- **updatepermissions**; Update the layer permissions; synchronize it back to the GeoSpatial Server. This option is also available from the *Layer Details* page.
- **updateattributes**; Update the layer attributes; synchronize it back to the GeoSpatial Server. This option is also available from the *Layer Details* page.
- **updatethumbnails**; Update the map styles and thumbnails. This option is also available from the *Layer Details* page.
- **remove-duplicates**; Removes duplicated Links.

First of all let's take a look at the `-help` option of the `sync_geonode_layers` management command in order to inspect all the command options and features.

Run

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py sync_geonode_layers --help
```

Nota: If you enabled `local_settings.py` the command will change as following:

```
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py sync_geonode_layers --
→help
```

This will produce output that looks like the following

```
usage: manage.py sync_geonode_layers [-h] [--version] [-v {0,1,2,3}]
                                     [--settings SETTINGS]
                                     [--pythonpath PYTHONPATH] [--traceback]
                                     [--no-color] [-i] [-d] [-f FILTER]
                                     [-u USERNAME] [--updatepermissions]
                                     [--updatethumbnails] [--updateattributes]

Update the GeoNode layers: permissions (including GeoFence database),
statistics, thumbnails

optional arguments:
-h, --help            show this help message and exit
--version             show program's version number and exit
-v {0,1,2,3}, --verbosity {0,1,2,3}
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
--settings SETTINGS  The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
--pythonpath PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
--traceback           Raise on CommandError exceptions
--no-color            Don't colorize the command output.
-i, --ignore-errors   Stop after any errors are encountered.
-d, --remove-duplicates
                        Remove duplicates first.
-f FILTER, --filter FILTER
                        Only update data the layers that match the given
                        filter.
-u USERNAME, --username USERNAME
                        Only update data owned by the specified username.
--updatepermissions  Update the layer permissions.
--updatethumbnails   Update the layer styles and thumbnails.
--updateattributes    Update the layer attributes.
```

- **Example 1:** I want to update/sync all layers permissions and attributes with the GeoSpatial Server

Aviso: Make always sure you are using the **correct** settings

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py sync_geonode_
→layers --updatepermissions --updateattributes
```

- **Example 2:** I want to regenerate the Thumbnails of all the Layers belonging to afabiani

Aviso: Make always sure you are using the **correct** settings

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py sync_geonode_
↳layers -u afabiani --updatethumbnails
```

Management Command `sync_geonode_maps`

This command is basically similar to the previous one, but affects the *Maps*; with some limitations.

The options are:

- **filter**; Only update data the maps titles that match the given filter.
- **username**; Only update data owned by the specified username.
- **updatethumbnails**; Update the map styles and thumbnails. This option is also available from the *Map Details* page.
- **remove-duplicates**; Removes duplicated Links.

First of all let's take a look at the `-help` option of the `sync_geonode_maps` management command in order to inspect all the command options and features.

Run

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py sync_geonode_maps --help
```

Nota: If you enabled `local_settings.py` the command will change as following:

```
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py sync_geonode_maps --
↳help
```

This will produce output that looks like the following

```
usage: manage.py sync_geonode_maps [-h] [--version] [-v {0,1,2,3}]
                                   [--settings SETTINGS]
                                   [--pythonpath PYTHONPATH] [--traceback]
                                   [--no-color] [-i] [-d] [-f FILTER]
                                   [-u USERNAME] [--updatethumbnails]

Update the GeoNode maps: permissions, thumbnails

optional arguments:
-h, --help            show this help message and exit
--version             show program's version number and exit
-v {0,1,2,3}, --verbosity {0,1,2,3}
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
--settings SETTINGS  The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
--pythonpath PYTHONPATH
                        A directory to add to the Python path, e.g.
```

(continues on next page)

(continuação da página anterior)

```

"/home/djangoprojects/myproject".
--traceback          Raise on CommandError exceptions
--no-color           Don't colorize the command output.
-i, --ignore-errors  Stop after any errors are encountered.
-d, --remove-duplicates
                    Remove duplicates first.
-f FILTER, --filter FILTER
                    Only update data the maps that match the given filter.
-u USERNAME, --username USERNAME
                    Only update data owned by the specified username.
--updatethumbnails  Update the map styles and thumbnails.

```

- **Example 1:** I want to regenerate the Thumbnail of the Map This is a test Map

Aviso: Make always sure you are using the **correct** settings

```

DJANGO_SETTINGS_MODULE=geonode.settings python manage.py sync_geonode_
↪maps --updatethumbnails -f 'This is a test Map'

```

Management Command `set_all_layers_metadata`

This command allows to reset **Metadata Attributes** and **Catalogue Schema** on Layers. The command will also update the *CSW Catalogue XML* and *Links* of GeoNode.

The options are:

- **filter**; Only update data the layers that match the given filter.
- **username**; Only update data owned by the specified username.
- **remove-duplicates**; Update the map styles and thumbnails.
- **delete-orphaned-thumbs**; Removes duplicated Links.
-
- **set-uuid**; will refresh the UUID based on the `UUID_HANDLER` if configured (Default False).
-
- **set_attrib**; If set will refresh the attributes of the resource taken from Geoserver. (Default True).
-
- **set_links**; If set will refresh the links of the resource. (Default True).

First of all let's take a look at the `-help` option of the `set_all_layers_metadata` management command in order to inspect all the command options and features.

Run

```

DJANGO_SETTINGS_MODULE=geonode.settings python manage.py set_all_layers_metadata --
↪help

```

Nota: If you enabled `local_settings.py` the command will change as following:

```
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py set_all_layers_
↪metadata --help
```

This will produce output that looks like the following

```
usage: manage.py set_all_layers_metadata [-h] [--version] [-v {0,1,2,3}]
                                         [--settings SETTINGS]
                                         [--pythonpath PYTHONPATH]
                                         [--traceback] [--no-color] [-i] [-d]
                                         [-t] [-f FILTER] [-u USERNAME]

Resets Metadata Attributes and Schema to All Layers

optional arguments:
-h, --help            show this help message and exit
--version             show program's version number and exit
-v {0,1,2,3}, --verbosity {0,1,2,3}
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
--settings SETTINGS  The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
--pythonpath PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
--traceback           Raise on CommandError exceptions
--no-color            Don't colorize the command output.
-i, --ignore-errors   Stop after any errors are encountered.
-d, --remove-duplicates
                        Remove duplicates first.
-t, --delete-orphaned-thumbnails
                        Delete Orphaned Thumbnails.
-f FILTER, --filter FILTER
                        Only update data the layers that match the given
                        filter
-u USERNAME, --username USERNAME
                        Only update data owned by the specified username
```

- **Example 1:** After having changed the Base URL, I want to regenerate all the Catalogue Schema and eventually remove all duplicates.

Aviso: Make always sure you are using the **correct** settings

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py set_all_layers_
↪metadata -d
```

1.25.3 Loading Data into GeoNode

There are situations where it is not possible or not convenient to use the *Upload Form* to add new Layers to GeoNode via the web interface. As an instance:

- The dataset is simply too big to be uploaded through a web interface.
- We would like to import some data from the mass storage programmatically.
- We would like to import some tables from a DataBase.
- We need to process the data first and, maybe, transform it to another format.

This section will walk you through the various options available to load data into your GeoNode from GeoServer, from the command-line or programmatically.

Aviso: Some parts of this section have been taken from the [GeoServer](#) project and training documentation.

Management Command `importlayers`

The `geonode.geoserver` Django app includes 2 management commands that you can use to load data in your GeoNode.

Both of them can be invoked by using the `manage.py` script.

First of all let's take a look at the `-help` option of the `importlayers` management command in order to inspect all the command options and features.

Run

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py importlayers --help
```

Nota: If you enabled `local_settings.py` the command will change as following:

```
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py importlayers --help
```

This will produce output that looks like the following

```
usage: manage.py importlayers [-h] [-hh HOST] [-u USERNAME] [-p PASSWORD]
                             [--version] [-v {0,1,2,3}] [--settings SETTINGS]
                             [--pythonpath PYTHONPATH] [--traceback] [--no-color]
                             [--force-color] [--skip-checks]
                             [path [path ...]]
```

Brings a directory full of data files into a GeoNode site.

Layers are added to the Django database, the GeoServer configuration, and the pycsw metadata index.

In order to perform the import, GeoNode must be up and running.

positional arguments:

path path [path...]

optional arguments:

`-h, --help` show this [help](#) message and [exit](#)
`--version` show program's [version number](#) and [exit](#)

(continues on next page)

(continuação da página anterior)

```

-v {0,1,2,3}, --verbosity {0,1,2,3}
    Verbosity level; 0=minimal output, 1=normal output,
    2=verbose output, 3=very verbose output
--settings SETTINGS    The Python path to a settings module, e.g.
    "myproject.settings.main". If this isn't provided, the
    DJANGO_SETTINGS_MODULE environment variable will be
    used.
--pythonpath PYTHONPATH
    A directory to add to the Python path, e.g.
    "/home/djangoprojects/myproject".
-hh HOST, --host HOST    Geonode host url
-u USERNAME, --username USERNAME
    Geonode username
-p PASSWORD, --password PASSWORD
    Geonode password

```

While the description of most of the options should be self explanatory, its worth reviewing some of the key options a bit more in details.

- The *-hh* Identifies the GeoNode server where we want to upload our layers. The default value is *http://localhost:8000*.
- The *-u* Identifies the username for the login. The default value is *admin*.
- The *-p* Identifies the password for the login. The default value is *admin*.

The import layers management command is invoked by specifying options as described above and specifying the path to a directory that contains multiple files. For purposes of this exercise, let's use the default set of testing layers that ship with geonode. You can replace this path with the directory to your own shapefiles.

This command will produce the following output to your terminal

```

san_andres_y_providencia_poi.shp: 201
san_andres_y_providencia_location.shp: 201
san_andres_y_providencia_administrative.shp: 201
san_andres_y_providencia_coastline.shp: 201
san_andres_y_providencia_highway.shp: 201
single_point.shp: 201
san_andres_y_providencia_water.shp: 201
san_andres_y_providencia_natural.shp: 201

1.7456605294117646 seconds per layer

Output data: {
  "success": [
    "san_andres_y_providencia_poi.shp",
    "san_andres_y_providencia_location.shp",
    "san_andres_y_providencia_administrative.shp",
    "san_andres_y_providencia_coastline.shp",
    "san_andres_y_providencia_highway.shp",
    "single_point.shp",
    "san_andres_y_providencia_water.shp",
    "san_andres_y_providencia_natural.shp"
  ],
  "errors": []
}

```

As output the command will print:

The status code, is the response coming from GeoNode. For example 201 means that the layer has been correctly uploaded

If you encounter errors while running this command, please check the GeoNode logs for more information.

Management Command `updatelayers`

While it is possible to import layers directly from your servers filesystem into your GeoNode, you may have an existing GeoServer that already has data in it, or you may want to configure data from a GeoServer which is not directly supported by uploading data.

GeoServer supports a wide range of data formats and connections to database, and while many of them are not supported as GeoNode upload formats, if they can be configured in GeoServer, you can add them to your GeoNode by following the procedure described below.

GeoServer supports 3 types of data: *Raster*, *Vector*, *Databases* and *Cascaded*.

For a list of the supported formats for each type of data, consult the following pages:

- <https://docs.geoserver.org/latest/en/user/data/vector/index.html>
- <https://docs.geoserver.org/latest/en/user/data/raster/index.html>
- <https://docs.geoserver.org/latest/en/user/data/database/index.html>
- <https://docs.geoserver.org/latest/en/user/data/cascaded/index.html>

Nota: Some of these raster or vector formats or database types require that you install specific plugins in your GeoServer in order to use the. Please consult the GeoServer documentation for more information.

Data from a PostGIS database

Lets walk through an example of configuring a new PostGIS database in GeoServer and then configuring those layers in your GeoNode.

First visit the GeoServer administration interface on your server. This is usually on port 8080 and is available at <http://localhost:8080/geoserver/web/>

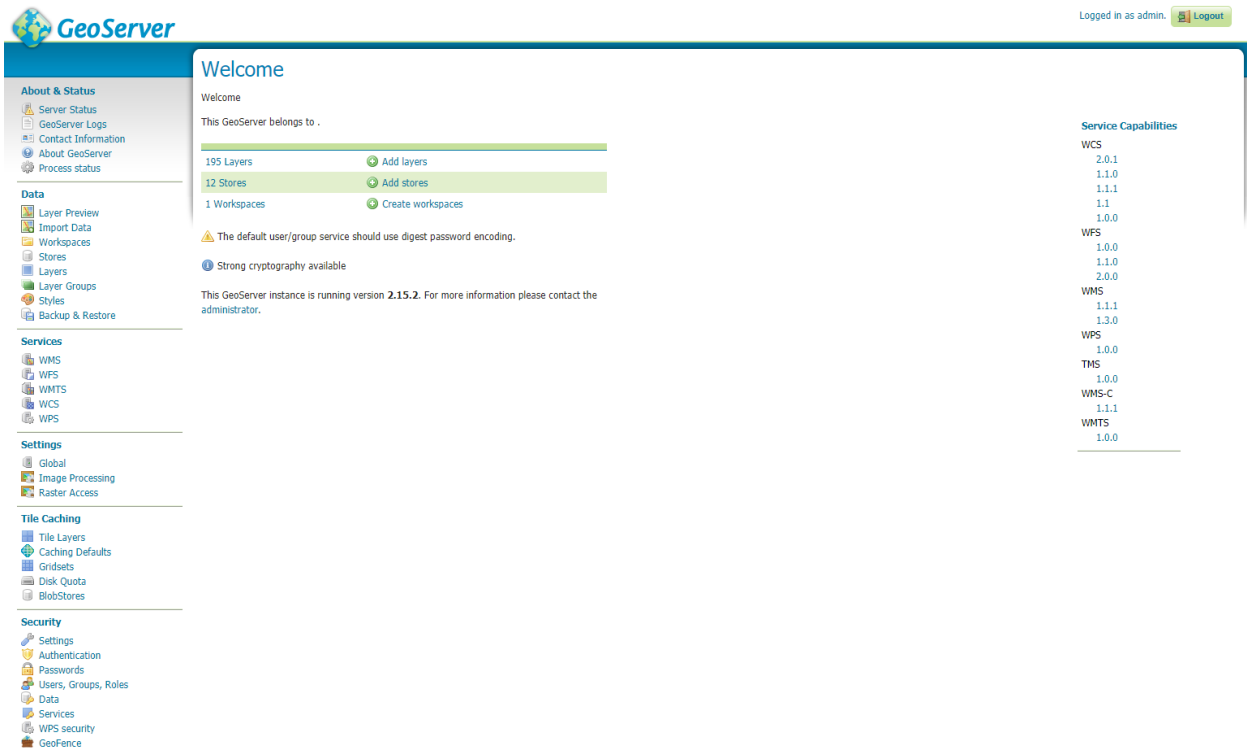
1. You should login with the superuser credentials you setup when you first configured your GeoNode instance.

Once you are logged in to the GeoServer Admin interface, you should see the following.

Nota: The number of stores, layers and workspaces may be different depending on what you already have configured in your GeoServer.

2. Next you want to select the «Stores» option in the left hand menu, and then the «Add new Store» option. The following screen will be displayed.
3. In this case, we want to select the PostGIS store type to create a connection to our existing database. On the next screen you will need to enter the parameters to connect to your PostGIS database (alter as necessary for your own database).

Nota: If you are unsure about any of the settings, leave them as the default.



GeoServer

Logged in as admin. [Logout](#)

Welcome

Welcome

This GeoServer belongs to .

195 Layers	Add layers
12 Stores	Add stores
1 Workspaces	Create workspaces

⚠ The default user/group service should use digest password encoding.

🔒 Strong cryptography available

This GeoServer instance is running version **2.15.2**. For more information please contact the administrator.

Service Capabilities

WCS	2.0.1
	1.1.0
	1.1.1
	1.1
	1.0.0
WFS	1.0.0
	1.1.0
	2.0.0
WMS	1.1.1
	1.3.0
WPS	1.0.0
	1.0.0
TMS	1.0.0
WMS-C	1.1.1
WMTS	1.0.0

About & Status

- Server Status
- GeoServer Logs
- Contact Information
- About GeoServer
- Process status

Data

- Layer Preview
- Import Data
- Workspaces
- Stores
- Layers
- Layer Groups
- Styles
- Backup & Restore

Services

- WMS
- WFS
- WMTS
- WCS
- WPS

Settings

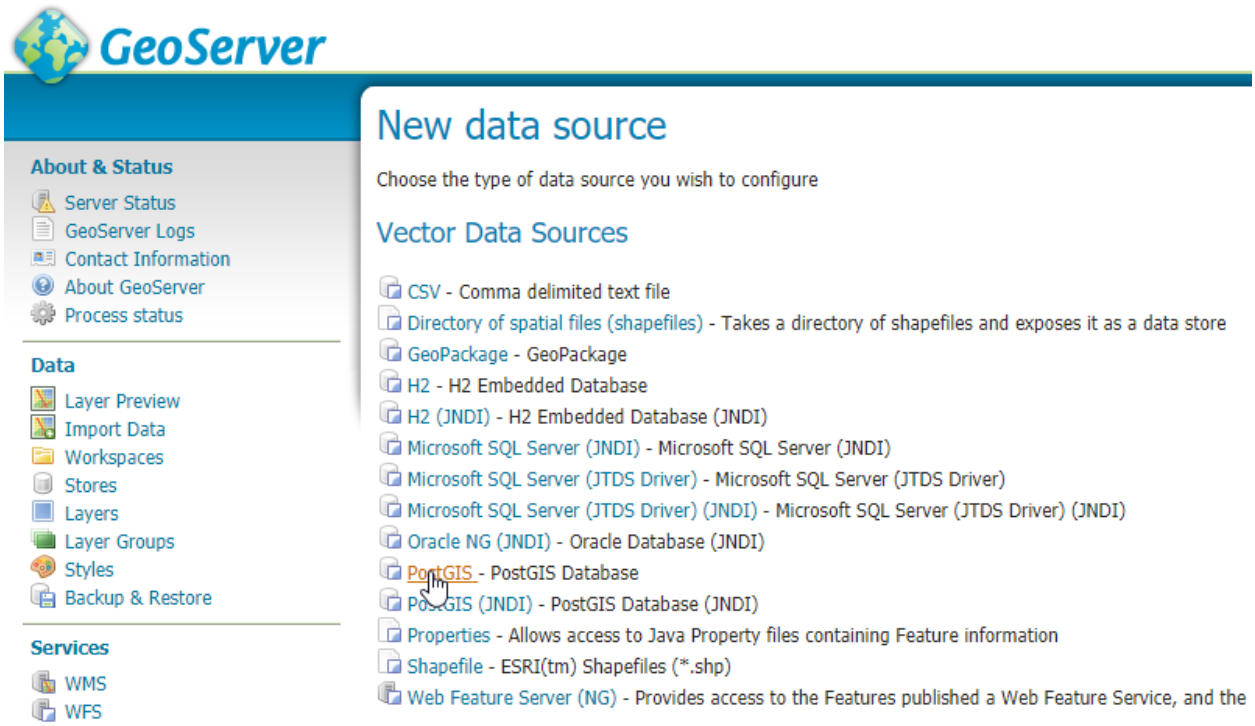
- Global
- Image Processing
- Raster Access

Tile Caching

- Tile Layers
- Caching Defaults
- Gridsets
- Disk Quota
- BlobStores

Security

- Settings
- Authentication
- Passwords
- Users, Groups, Roles
- Data
- Services
- WFS security
- GeoFence



GeoServer

New data source

Choose the type of data source you wish to configure

Vector Data Sources

- CSV - Comma delimited text file
- Directory of spatial files (shapefiles) - Takes a directory of shapefiles and exposes it as a data store
- GeoPackage - GeoPackage
- H2 - H2 Embedded Database
- H2 (JNDI) - H2 Embedded Database (JNDI)
- Microsoft SQL Server (JNDI) - Microsoft SQL Server (JNDI)
- Microsoft SQL Server (JTDS Driver) - Microsoft SQL Server (JTDS Driver)
- Microsoft SQL Server (JTDS Driver) (JNDI) - Microsoft SQL Server (JTDS Driver) (JNDI)
- Oracle NG (JNDI) - Oracle Database (JNDI)
- PostGIS - PostGIS Database
- PostGIS (JNDI) - PostGIS Database (JNDI)
- Properties - Allows access to Java Property files containing Feature information
- Shapefile - ESRI(tm) Shapefiles (*.shp)
- Web Feature Server (NG) - Provides access to the Features published a Web Feature Service, and the

**About & Status**

- Server Status
- GeoServer Logs
- Contact Information
- About GeoServer
- Process status

Data

- Layer Preview
- Import Data
- Workspaces
- Stores
- Layers
- Layer Groups
- Styles
- Backup & Restore

Services

- WMS
- WFS
- WMTS
- WCS
- WPS

Settings

- Global
- Image Processing
- Raster Access

Tile Caching

- Tile Layers
- Caching Defaults

New Vector Data Source

Add a new vector data source

PostGIS
PostGIS Database

Basic Store Info

Workspace *

geonode ▼

Data Source Name *

my_db_data

Description

☒ Enabled

Connection Parameters

host *

localhost

port *

5432

database

geonode_data

schema

public

user *

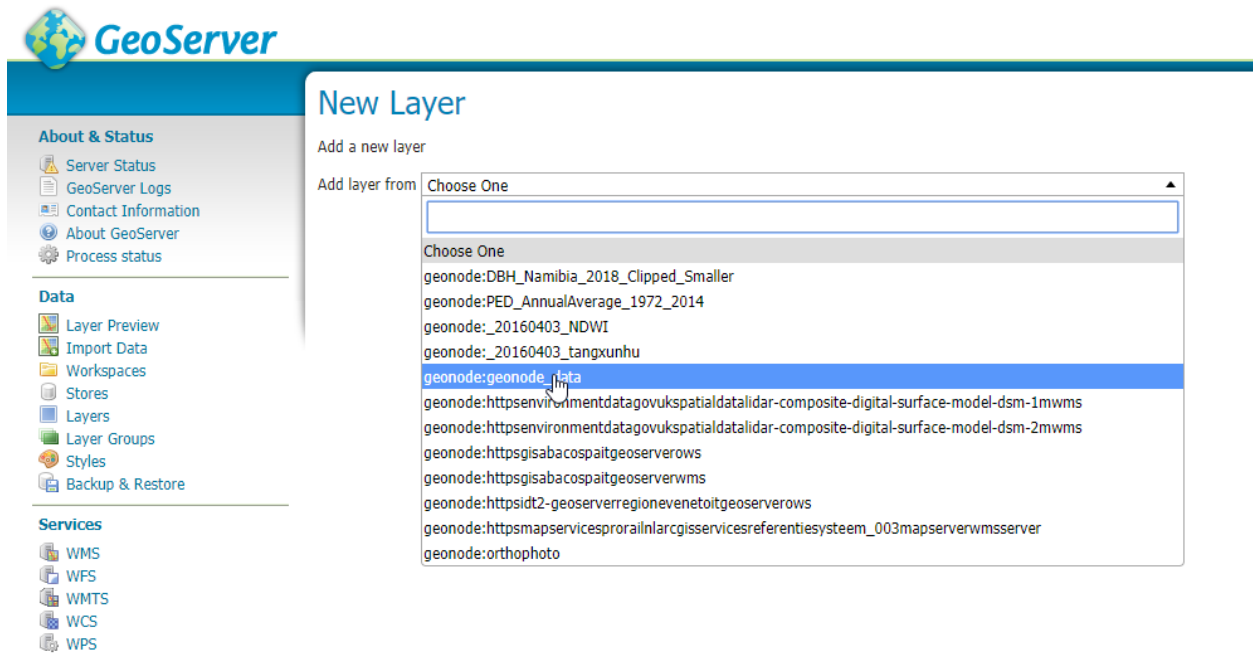
geonode

passwd

Namespace *

http://www.geonode.org/

- The next screen lets you configure the layers in your database. This will of course be different depending on the layers in your database.



- Select the «Publish» button for one of the layers and the next screen will be displayed where you can enter metadata for this layer. Since we will be managing this metadata in GeoNode, we can leave these alone for now.
- The things that *must* be specified are the Declared SRS and you must select the «Compute from Data» and «Compute from native bounds» links after the SRS is specified.
- Click save and this layer will now be configured for use in your GeoServer.
- The next step is to configure these layers in GeoNode. The `updatelayers` management command can be used for this purpose. As with `importlayers`, it's useful to look at the command line options for this command by passing the `-help` option

Run

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py updatelayers --
↪help
```

Nota: If you enabled `local_settings.py` the command will change as following:

```
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py
↪updatelayers --help
```

This will produce output that looks like the following

New Layer

Add a new layer

Add layer from

You can create a new feature type by manually configuring the attribute names and types. [Create new feature type...](#)

On databases you can also create a new feature type by configuring a native SQL statement. [Configure new SQL view...](#)

Here is a list of resources contained in the store 'geonode_data'. Click on the layer you wish to configure

Published	Layer name	Action
☒	testspnrc	Publish again
☒	trial_7thsept_29Records	Publish again
☒	trial_7thsept_30Records	Publish again
☒	ua_test	Publish again
☒	verger	Publish again
☒	verger0	Publish again
☒	voirie_gen	Publish again
☒	waterways	Publish again
☒	waterways0	Publish again
☐	_1_SARMIENTO_ENERO_2018	Publish
☐	cambodia_oilfields	Publish
☐	kagaa	Publish
☐	map4utm2	Publish
☐	study001_ins00_Sept6_Buffer	Publish
☐	study_as_geojson_7thSep	Publish
☐	study_as_logitude_latitude	Publish
☐	study_as_logitude_latitude0	Publish
☐	study_as_logitude_latitude1	Publish
☐	study_as_logitude_latitude2	Publish

 [Cloud](#)
 [Image Processing](#)
 [Raster Access](#)

Tile Caching

 [Tile Layers](#)
 [Caching Defaults](#)
 [Gridsets](#)
 [Disk Quota](#)
 [BlobStores](#)

Security

 [Settings](#)
 [Authentication](#)
 [Passwords](#)
 [Users, Groups, Roles](#)
 [Data](#)
 [Services](#)
 [WPS security](#)
 [GeoFence](#)
 [GeoFence Data Rules](#)
 [GeoFence Admin Rules](#)

Monitor

 [Activity](#)
 [Reports](#)

Demos

Tools

Keywords

Current Keywords

[Remove selected](#)

New Keyword

Vocabulary

[Add Keyword](#)

Metadata links

No metadata links so far

[Add link](#) *Note only FGDC and TC211 metadata links show up in WMS 1.1.1 capabilities*

Data links

No data links so far

[Add link](#)

Coordinate Reference Systems

Native SRS

[EPSG:WGS 84...](#)

Declared SRS

[Find...](#) [EPSG:WGS 84...](#)

SRS handling

Bounding Boxes

Native Bounding Box

Min X	Min Y	Max X	Max Y
-69.078485273	-45.60003889	-69.057928671999	-45.579602872

[Compute from data](#)

[Compute from bounds](#)

Lat/Lon Bounding Box

Min X	Min Y	Max X	Max Y

[Compute from native bounds](#)

Curved geometries control


```
usage: manage.py updatelayers [-h] [--version] [-v {0,1,2,3}]
                                [--settings SETTINGS] [--pythonpath_
→PYTHONPATH]
                                [--traceback] [--no-color] [-i]
                                [--skip-unadvertised]
                                [--skip-geonode-registered] [--remove-
→deleted]
                                [-u USER] [-f FILTER] [-s STORE] [-w_
→WORKSPACE]
                                [-p PERMISSIONS]

Update the GeoNode application with data from GeoServer

optional arguments:
-h, --help            show this help message and exit
--version            show program's version number and exit
-v {0,1,2,3}, --verbosity {0,1,2,3}
                        Verbosity level; 0=minimal output, 1=normal_
→output,
                        2=verbose output, 3=very verbose output
--settings SETTINGS  The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't_
→provided, the
                        DJANGO_SETTINGS_MODULE environment variable will_
→be
                        used.
--pythonpath PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
--traceback          Raise on CommandError exceptions
--no-color           Don't colorize the command output.
-i, --ignore-errors  Stop after any errors are encountered.
--skip-unadvertised  Skip processing unadvertised layers from GeoSever.
--skip-geonode-registered
                        Just processing GeoServer layers still not_
→registered
                        in GeoNode.
--remove-deleted     Remove GeoNode layers that have been deleted from
                        GeoSever.
-u USER, --user USER
→imported            Name of the user account which should own the_
                        layers
-f FILTER, --filter FILTER
                        Only update data the layers that match the given
                        filter
-s STORE, --store STORE
→geoserver           Only update data the layers for the given_
                        store name
-w WORKSPACE, --workspace WORKSPACE
                        Only update data on specified workspace
-p PERMISSIONS, --permissions PERMISSIONS
                        Permissions to apply to each layer
```

Aviso: One of the `--workspace` or `--store` must be always specified if you want to ingest Layers belonging to a specific Workspace. As an instance, in order to ingest the layers present into the geonode workspace, you will

need to specify the option `-w geonode`.

9. Let's ingest the layer `geonode:_1_SARMIENTO_ENERO_2018` from the geonode workspace.

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py updatelayers -w
↳geonode -f _1_SARMIENTO_ENERO_2018
```

```
Inspecting the available layers in GeoServer ...
Found 1 layers, starting processing
/usr/local/lib/python2.7/site-packages/owslib/iso.py:117: FutureWarning:
↳the .identification and .serviceidentification properties will merge
↳into .identification being a list of properties. This is currently
↳implemented in .identificationinfo. Please see https://github.com/
↳geopython/OWSLib/issues/38 for more information
FutureWarning)
/usr/local/lib/python2.7/site-packages/owslib/iso.py:495: FutureWarning:
↳The .keywords and .keywords2 properties will merge into the .keywords
↳property in the future, with .keywords becoming a list of MD_Keywords
↳instances. This is currently implemented in .keywords2. Please see
↳https://github.com/geopython/OWSLib/issues/301 for more information
FutureWarning)
Content-Type: text/html; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: [master.demo.geonode.org] A new layer has been uploaded
From: webmaster@localhost
To: mapadeldelito@chubut.gov.ar
Reply-To: webmaster@localhost
Date: Tue, 08 Oct 2019 12:26:17 -0000
Message-ID: <20191008122617.28801.94967@d3cf85425231>

<body>
You have received the following notice from master.demo.geonode.org:
<p>

The user <i><a href="http://master.demo.geonode.org/people/profile/admin
↳">admin</a></i> uploaded the following layer:<br/>
<strong>_1_SARMIENTO_ENERO_2018</strong><br/>
You can visit the layer's detail page here: http://master.demo.geonode.
↳org/layers/geonode:_1_SARMIENTO_ENERO_2018

</p>
<p>
To change how you receive notifications, please go to http://master.demo.
↳geonode.org
</p>
</body>

-----
↳-----
Content-Type: text/html; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: [master.demo.geonode.org] A new layer has been uploaded
From: webmaster@localhost
```

(continues on next page)

(continuação da página anterior)

```

To: giacomob8vinci@gmail.com
Reply-To: webmaster@localhost
Date: Tue, 08 Oct 2019 12:26:17 -0000
Message-ID: <20191008122617.28801.53784@d3cf85425231>

<body>
You have received the following notice from master.demo.geonode.org:
<p>

The user <i><a href="http://master.demo.geonode.org/people/profile/admin
→">admin</a></i> uploaded the following layer:<br/>
<strong>_1_SARMIENTO_ENERO_2018</strong><br/>
You can visit the layer's detail page here: http://master.demo.geonode.
→org/layers/geonode:_1_SARMIENTO_ENERO_2018

</p>
<p>
To change how you receive notifications, please go to http://master.demo.
→geonode.org
</p>
</body>

-----
→-----
Content-Type: text/html; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: [master.demo.geonode.org] A new layer has been uploaded
From: webmaster@localhost
To: fmgagliano@gmail.com
Reply-To: webmaster@localhost
Date: Tue, 08 Oct 2019 12:26:17 -0000
Message-ID: <20191008122617.28801.26265@d3cf85425231>

<body>
You have received the following notice from master.demo.geonode.org:
<p>

The user <i><a href="http://master.demo.geonode.org/people/profile/admin
→">admin</a></i> uploaded the following layer:<br/>
<strong>_1_SARMIENTO_ENERO_2018</strong><br/>
You can visit the layer's detail page here: http://master.demo.geonode.
→org/layers/geonode:_1_SARMIENTO_ENERO_2018

</p>
<p>
To change how you receive notifications, please go to http://master.demo.
→geonode.org
</p>
</body>

-----
→-----
Found geoserver resource for this layer: _1_SARMIENTO_ENERO_2018
... Creating Default Resource Links for Layer [geonode:_1_SARMIENTO_
→ENERO_2018]

```

(continues on next page)

(continuação da página anterior)

```
-- Resource Links[Prune old links]...
-- Resource Links[Prune old links]...done!
-- Resource Links[Compute parameters for the new links]...
-- Resource Links[Create Raw Data download link]...
-- Resource Links[Create Raw Data download link]...done!
-- Resource Links[Set download links for WMS, WCS or WFS and KML]...
-- Resource Links[Set download links for WMS, WCS or WFS and KML]...done!
-- Resource Links[Legend link]...
-- Resource Links[Legend link]...done!
-- Resource Links[Thumbnail link]...
-- Resource Links[Thumbnail link]...done!
-- Resource Links[OWS Links]...
-- Resource Links[OWS Links]...done!
Content-Type: text/html; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: [master.demo.geonode.org] A layer has been updated
From: webmaster@localhost
To: mapadeldelito@chubut.gov.ar
Reply-To: webmaster@localhost
Date: Tue, 08 Oct 2019 12:26:20 -0000
Message-ID: <20191008122620.28801.81598@d3cf85425231>
```

```
<body>
You have received the following notice from master.demo.geonode.org:
<p>

The following layer was updated:<br/>
<strong>_1_SARMIENTO_ENERO_2018</strong>, owned by <i><a href="http://
↪master.demo.geonode.org/people/profile/admin">admin</a></i><br/>
You can visit the layer's detail page here: http://master.demo.geonode.
↪org/layers/geonode:_1_SARMIENTO_ENERO_2018

</p>
<p>
To change how you receive notifications, please go to http://master.demo.
↪geonode.org
</p>
</body>
```

```
-----
↪-----
Content-Type: text/html; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: [master.demo.geonode.org] A layer has been updated
From: webmaster@localhost
To: giacomovinci@gmail.com
Reply-To: webmaster@localhost
Date: Tue, 08 Oct 2019 12:26:20 -0000
Message-ID: <20191008122620.28801.93778@d3cf85425231>

<body>
You have received the following notice from master.demo.geonode.org:
<p>
```

(continues on next page)

(continuação da página anterior)

```

The following layer was updated:<br/>
<strong>_1_SARMIENTO_ENERO_2018</strong>, owned by <i><a href="http://
↳master.demo.geonode.org/people/profile/admin">admin</a></i><br/>
You can visit the layer's detail page here: http://master.demo.geonode.
↳org/layers/geonode:_1_SARMIENTO_ENERO_2018

</p>
<p>
To change how you receive notifications, please go to http://master.demo.
↳geonode.org
</p>
</body>

-----
↳-----
Content-Type: text/html; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: [master.demo.geonode.org] A layer has been updated
From: webmaster@localhost
To: fmgagliano@gmail.com
Reply-To: webmaster@localhost
Date: Tue, 08 Oct 2019 12:26:20 -0000
Message-ID: <20191008122620.28801.58585@d3cf85425231>

<body>
You have received the following notice from master.demo.geonode.org:
<p>

The following layer was updated:<br/>
<strong>_1_SARMIENTO_ENERO_2018</strong>, owned by <i><a href="http://
↳master.demo.geonode.org/people/profile/admin">admin</a></i><br/>
You can visit the layer's detail page here: http://master.demo.geonode.
↳org/layers/geonode:_1_SARMIENTO_ENERO_2018

</p>
<p>
To change how you receive notifications, please go to http://master.demo.
↳geonode.org
</p>
</body>

-----
↳-----
Found geoserver resource for this layer: _1_SARMIENTO_ENERO_2018
/usr/local/lib/python2.7/site-packages/geoserver/style.py:80:
↳FutureWarning: The behavior of this method will change in future
↳versions. Use specific 'len(elem)' or 'elem is not None' test instead.
if not user_style:
/usr/local/lib/python2.7/site-packages/geoserver/style.py:84:
↳FutureWarning: The behavior of this method will change in future
↳versions. Use specific 'len(elem)' or 'elem is not None' test instead.
if user_style:
... Creating Default Resource Links for Layer [geonode:_1_SARMIENTO_
↳ENERO_2018]

```

(continues on next page)

(continuação da página anterior)

```
-- Resource Links[Prune old links]...
-- Resource Links[Prune old links]...done!
-- Resource Links[Compute parameters for the new links]...
-- Resource Links[Create Raw Data download link]...
-- Resource Links[Create Raw Data download link]...done!
-- Resource Links[Set download links for WMS, WCS or WFS and KML]...
-- Resource Links[Set download links for WMS, WCS or WFS and KML]...done!
-- Resource Links[Legend link]...
-- Resource Links[Legend link]...done!
-- Resource Links[Thumbnail link]...
-- Resource Links[Thumbnail link]...done!
-- Resource Links[OWS Links]...
-- Resource Links[OWS Links]...done!
[created] Layer _1_SARMIENTO_ENERO_2018 (1/1)

Finished processing 1 layers in 5.0 seconds.

1 Created layers
0 Updated layers
0 Failed layers
5.000000 seconds per layer
```

Nota: In case you don't specify the `-f` option, the layers that already exist in your GeoNode will be just updated and the configuration synchronized between GeoServer and GeoNode.

Aviso: When updating from GeoServer, the configuration on GeoNode will be changed!

Using GDAL and OGR to convert your Data for use in GeoNode

GeoNode supports uploading data in *ESRI shapefiles*, *GeoTIFF*, *CSV*, *GeoJSON*, *ASCII-GRID* and *KML / KMZ* formats (for the last three formats only if you are using the `geonode.importer` backend).

- If your data is in other formats, you will need to convert it into one of these formats for use in GeoNode.
- If your *Raster* data is not correctly processed, it might be almost unusable with GeoServer and GeoNode. You will need to process it using *GDAL*.

You need to make sure that you have the GDAL library installed on your system. On Ubuntu you can install this package with the following command:

```
sudo apt-get install gdal-bin
```

OGR (Vector Data)

OGR is used to manipulate vector data. In this example, we will use MapInfo .tab files and convert them to shapefiles with the ogr2ogr command. We will use sample MapInfo files from the website linked below.

<http://services.land.vic.gov.au/landchannel/content/help?name=sampleddata>

You can download the Admin;(Postcode) layer by issuing the following command:

```
$ wget http://services.land.vic.gov.au/sampleddata/shape/admin_postcode_vm.zip
```

You will need to unzip this dataset by issuing the following command:

```
$ unzip admin_postcode_vm.zip
```

This will leave you with the following files in the directory where you executed the above commands:

```
|-- ANZVI0803003025.htm
|-- DSE_Data_Access_Licence.pdf
|-- VMADMIN.POSTCODE_POLYGON.xml
|-- admin_postcode_vm.zip
--- vicgrid94
    --- mif
        --- lga_polygon
            --- macedon\ ranges
                |-- EXTRACT_POLYGON.mid
                |-- EXTRACT_POLYGON.mif
            --- VMADMIN
                |-- POSTCODE_POLYGON.mid
                --- POSTCODE_POLYGON.mif
```

First, lets inspect this file set using the following command:

```
$ ogrinfo -so vicgrid94/mif/lga_polygon/macedon\ ranges/VMADMIN/POSTCODE_POLYGON.mid_
↳ POSTCODE_POLYGON
```

The output will look like the following:

```
Had to open data source read-only.
INFO: Open of `vicgrid94/mif/lga_polygon/macedon ranges/VMADMIN/POSTCODE_POLYGON.mid'
      using driver `MapInfo File' successful.

Layer name: POSTCODE_POLYGON
Geometry: 3D Unknown (any)
Feature Count: 26
Extent: (2413931.249367, 2400162.366186) - (2508952.174431, 2512183.046927)
Layer SRS WKT:
PROJCS["unnamed",
  GEOGCS["unnamed",
    DATUM["GDA94",
      SPHEROID["GRS 80",6378137,298.257222101],
      TOWGS84[0,0,0,-0,-0,-0,0]],
    PRIMEM["Greenwich",0],
    UNIT["degree",0.0174532925199433]],
  PROJECTION["Lambert_Conformal_Conic_2SP"],
  PARAMETER["standard_parallel_1",-36],
  PARAMETER["standard_parallel_2",-38],
  PARAMETER["latitude_of_origin",-37],
```

(continues on next page)

(continuação da página anterior)

```

    PARAMETER["central_meridian",145],
    PARAMETER["false_easting",2500000],
    PARAMETER["false_northing",2500000],
    UNIT["Meter",1]]
PFI: String (10.0)
POSTCODE: String (4.0)
FEATURE_TYPE: String (6.0)
FEATURE_QUALITY_ID: String (20.0)
PFI_CREATED: Date (10.0)
UFI: Real (12.0)
UFI_CREATED: Date (10.0)
UFI_OLD: Real (12.0)

```

This gives you information about the number of features, the extent, the projection and the attributes of this layer.

Next, lets go ahead and convert this layer into a shapefile by issuing the following command:

```

$ ogr2ogr -t_srs EPSG:4326 postcode_polygon.shp vicgrid94/mif/lga_polygon/macedon\
↪ranges/VMADMIN/POSTCODE_POLYGON.mid POSTCODE_POLYGON

```

Note that we have also reprojected the layer to the WGS84 spatial reference system with the `-t_srs ogr2ogr` option.

The output of this command will look like the following:

```

Warning 6: Normalized/launched field name: 'FEATURE_TYPE' to 'FEATURE_TY'
Warning 6: Normalized/launched field name: 'FEATURE_QUALITY_ID' to 'FEATURE_QU'
Warning 6: Normalized/launched field name: 'PFI_CREATED' to 'PFI_CREATE'
Warning 6: Normalized/launched field name: 'UFI_CREATED' to 'UFI_CREATE'

```

This output indicates that some of the field names were truncated to fit into the constraint that attributes in shapefiles are only 10 characters long.

You will now have a set of files that make up the `postcode_polygon.shp` shapefile set. We can inspect them by issuing the following command:

```

$ ogrinfo -so postcode_polygon.shp postcode_polygon

```

The output will look similar to the output we saw above when we inspected the MapInfo file we converted from:

```

INFO: Open of `postcode_polygon.shp'
      using driver `ESRI Shapefile' successful.

Layer name: postcode_polygon
Geometry: Polygon
Feature Count: 26
Extent: (144.030296, -37.898156) - (145.101137, -36.888878)
Layer SRS WKT:
GEOGCS["GCS_WGS_1984",
  DATUM["WGS_1984",
    SPHEROID["WGS_84", 6378137, 298.257223563]],
  PRIMEM["Greenwich", 0],
  UNIT["Degree", 0.017453292519943295]]
PFI: String (10.0)
POSTCODE: String (4.0)
FEATURE_TY: String (6.0)
FEATURE_QU: String (20.0)
PFI_CREATE: Date (10.0)

```

(continues on next page)

(continuação da página anterior)

```

UFI: Real (12.0)
UFI_CREATE: Date (10.0)
UFI_OLD: Real (12.0)

```

These files can now be loaded into your GeoNode instance via the normal uploader.

Visit the upload page in your GeoNode, drag and drop the files that composes the shapefile that you have generated using the GDAL ogr2ogr command (postcode_polygon.dbf, postcode_polygon.prj, postcode_polygon.shp, postcode_polygon.shx). Give the permissions as needed and then click the «Upload files» button.

The screenshot displays the GeoNode 'UPLOAD LAYERS' page. At the top, there's a navigation bar with 'HOME', 'LAYERS', 'MAPS', 'DOCUMENTS', 'PEOPLE', and 'SEARCH'. The main heading is 'UPLOAD LAYERS'. Below it is a large dashed box with the text 'Drop files here'. Underneath this box, it says 'or select them one by one:' followed by a 'Choose Files' button and 'No file chosen'. A dark bar labeled 'FILES TO BE UPLOADED' contains a list of four files: 'postcode_polygon.dbf', 'postcode_polygon.prj', 'postcode_polygon.shp', and 'postcode_polygon.shx', each with a 'Remove' link. Below the file list is a 'Select the charset or leave default' dropdown menu currently set to 'UTF-8/Unicode', with 'Clear' and 'Upload files' buttons. To the right, the 'PERMISSIONS' section has three sections: 'Who can view and download this data?' with radio buttons for 'Anyone', 'Any registered user', and 'Only users who can edit'; 'Who can edit this data?' with radio buttons for 'Any registered user' and 'Only the following users or groups:' followed by a 'Choose one or more users...' input; and 'Who can manage and edit this data?' with a 'Choose one or more users...' input. The footer shows 'Powered by GeoNode version 2.0.dev.20130911150113' and a language dropdown set to 'English'.

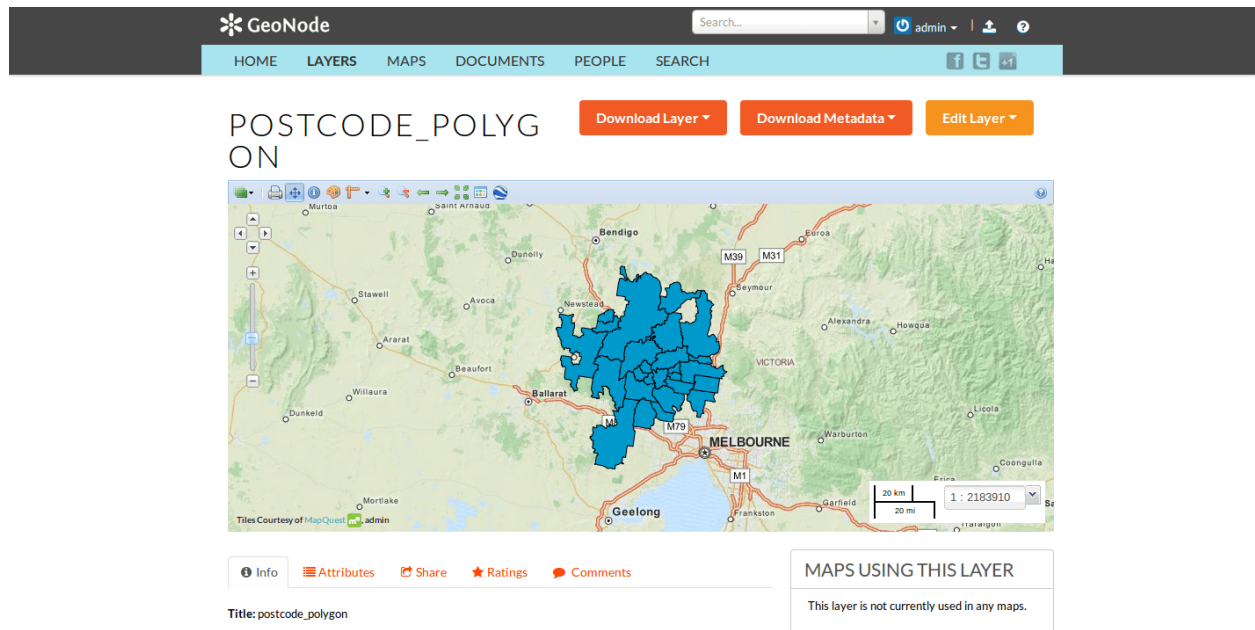
As soon as the import process completes, you will have the possibility to go straight to the layer info page («Layer Info» button), or to edit the metadata for that layer («Edit Metadata» button), or to manage the styles for that layer («Manage Styles»).

GDAL (Raster Data)

Let's see several examples on how to either convert raster data into different formats and/or process it to get the best performances.

References:

- https://geoserver.geo-solutions.it/edu/en/raster_data/processing.html
- https://geoserver.geo-solutions.it/edu/en/raster_data/advanced_gdal/



Raster Data Conversion: Arc/Info Binary and ASCII Grid data into GeoTIFF format.

Let's assume we have a sample ASCII Grid file compressed as an archive.

```
# Un-tar the files
tar -xvf sample_asc.tar
```

You will be left with the following files on your filesystem:

```
-- batemans_ele
|   |-- dblbnd.adf
|   |-- hdr.adf
|   |-- metadata.xml
|   |-- prj.adf
|   |-- sta.adf
|   |-- w001001.adf
|   |-- w001001x.adf
|-- batemans_elevation.asc
```

The file `batemans_elevation.asc` is an Arc/Info ASCII Grid file and the files in the `batemans_ele` directory are an Arc/Info Binary Grid file.

You can use the `gdalinfo` command to inspect both of these files by executing the following command:

```
gdalinfo batemans_elevation.asc
```

The output should look like the following:

```
Driver: AAIGrid/Arc/Info ASCII Grid
Files: batemans_elevation.asc
Size is 155, 142
Coordinate System is ``
Origin = (239681.000000000000000,6050551.000000000000000)
Pixel Size = (100.000000000000000,-100.000000000000000)
```

(continues on next page)

(continuação da página anterior)

```

Corner Coordinates:
Upper Left  ( 239681.000, 6050551.000)
Lower Left  ( 239681.000, 6036351.000)
Upper Right ( 255181.000, 6050551.000)
Lower Right ( 255181.000, 6036351.000)
Center      ( 247431.000, 6043451.000)
Band 1 Block=155x1 Type=Float32, ColorInterp=Undefined
NoData Value=-9999

```

You can then inspect the batemans_ele files by executing the following command:

```
gdalinfo batemans_ele
```

And this should be the corresponding output:

```

Driver: AIG/Arc/Info Binary Grid
Files: batemans_ele
  batemans_ele/dblwnd.adf
  batemans_ele/hdr.adf
  batemans_ele/metadata.xml
  batemans_ele/prj.adf
  batemans_ele/sta.adf
  batemans_ele/w001001.adf
  batemans_ele/w001001x.adf
Size is 155, 142
Coordinate System is:
PROJCS["unnamed",
  GEOGCS["GDA94",
    DATUM["Geocentric_Datum_of_Australia_1994",
      SPHEROID["GRS 1980",6378137,298.257222101,
        AUTHORITY["EPSG","7019"]],
      TOWGS84[0,0,0,0,0,0,0],
      AUTHORITY["EPSG","6283"]],
    PRIMEM["Greenwich",0,
      AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
      AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4283"]],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["latitude_of_origin",0],
  PARAMETER["central_meridian",153],
  PARAMETER["scale_factor",0.9996],
  PARAMETER["false_easting",500000],
  PARAMETER["false_northing",1000000],
  UNIT["METERS",1]]
Origin = (239681.0000000000000000,6050551.0000000000000000)
Pixel Size = (100.0000000000000000,-100.0000000000000000)
Corner Coordinates:
Upper Left  ( 239681.000, 6050551.000) (150d 7'28.35"E, 35d39'16.56"S)
Lower Left  ( 239681.000, 6036351.000) (150d 7'11.78"E, 35d46'56.89"S)
Upper Right ( 255181.000, 6050551.000) (150d17'44.07"E, 35d39'30.83"S)
Lower Right ( 255181.000, 6036351.000) (150d17'28.49"E, 35d47'11.23"S)
Center      ( 247431.000, 6043451.000) (150d12'28.17"E, 35d43'13.99"S)
Band 1 Block=256x4 Type=Float32, ColorInterp=Undefined
  Min=-62.102 Max=142.917
NoData Value=-3.4028234663852886e+38

```


You will notice that the `batemans_elevation.asc` file does *not* contain projection information while the `batemans_ele` file does. Because of this, let's use the `batemans_ele` files for this exercise and convert them to a GeoTiff for use in GeoNode. We will also reproject this file into WGS84 in the process. This can be accomplished with the following command.

```
gdalwarp -t_srs EPSG:4326 batemans_ele batemans_ele.tif
```

The output will show you the progress of the conversion and when it is complete, you will be left with a `batemans_ele.tif` file that you can upload to your GeoNode.

You can inspect this file with the `gdalinfo` command:

```
gdalinfo batemans_ele.tif
```

Which will produce the following output:

```
Driver: GTiff/GeoTIFF
Files: batemans_ele.tif
Size is 174, 130
Coordinate System is:
GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563,
            AUTHORITY["EPSG","7030"]],
        AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0],
    UNIT["degree",0.0174532925199433],
    AUTHORITY["EPSG","4326"]]
Origin = (150.119938943722502,-35.654598806259330)
Pixel Size = (0.0010111114155919,-0.0010111114155919)
Metadata:
  AREA_OR_POINT=Area
Image Structure Metadata:
  INTERLEAVE=BAND
Corner Coordinates:
Upper Left  ( 150.1199389, -35.6545988) (150d 7'11.78"E, 35d39'16.56"S)
Lower Left  ( 150.1199389, -35.7860436) (150d 7'11.78"E, 35d47' 9.76"S)
Upper Right ( 150.2958728, -35.6545988) (150d17'45.14"E, 35d39'16.56"S)
Lower Right ( 150.2958728, -35.7860436) (150d17'45.14"E, 35d47' 9.76"S)
Center      ( 150.2079059, -35.7203212) (150d12'28.46"E, 35d43'13.16"S)
Band 1 Block=174x11 Type=Float32, ColorInterp=Gray
```

Raster Data Optimization: Optimizing and serving big raster data

(ref: https://geoserver.geo-solutions.it/edu/en/raster_data/advanced_gdal/example5.html)

When dealing with big raster datasets it could be very useful to use tiles.

Tiling allows large raster datasets to be broken-up into manageable pieces and are fundamental in defining and implementing a higher level raster I/O interface.

In this example we will use the original dataset of the `chiangMai_ortho_optimized` public raster layer which is currently available on the Thai [CHIANG MAI Urban Flooding GeoNode platform](#).

This dataset contains an orthorectified image stored as RGBa GeoTiff with 4 bands, three bands for the RGB and one for transparency (the alpha channel).

Calling the `gdalinfo` command to see detailed information:

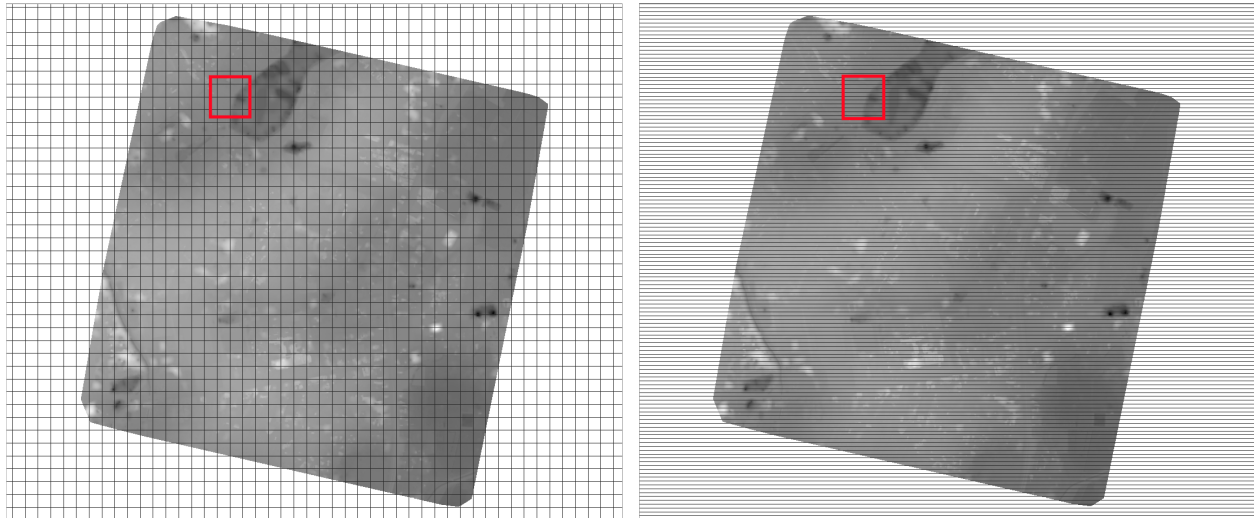
```
gdalinfo chiangMai_ortho.tif
```

It will produce the following results:

```
Driver: GTiff/GeoTIFF
Files: chiangMai_ortho.tif
Size is 63203, 66211
Coordinate System is:
PROJCS["WGS 84 / UTM zone 47N",
  GEOGCS["WGS 84",
    DATUM["WGS_1984",
      SPHEROID["WGS 84",6378137,298.257223563,
        AUTHORITY["EPSG","7030"]],
      AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
      AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
      AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4326"]],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["latitude_of_origin",0],
  PARAMETER["central_meridian",99],
  PARAMETER["scale_factor",0.9996],
  PARAMETER["false_easting",500000],
  PARAMETER["false_northing",0],
  UNIT["metre",1,
    AUTHORITY["EPSG","9001"]],
  AXIS["Easting",EAST],
  AXIS["Northing",NORTH],
  AUTHORITY["EPSG","32647"]]]
Origin = (487068.7747500000040513,2057413.8898100000080615)
Pixel Size = (0.028850000000000,-0.028850000000000)
Metadata:
AREA_OR_POINT=Area
TIFFTAG_SOFTWARE=pix4dmapper
Image Structure Metadata:
COMPRESSION=LZW
INTERLEAVE=PIXEL
Corner Coordinates:
Upper Left  ( 487068.775, 2057413.890) ( 98d52'38.72"E, 18d36'27.34"N)
Lower Left  ( 487068.775, 2055503.702) ( 98d52'38.77"E, 18d35'25.19"N)
Upper Right ( 488892.181, 2057413.890) ( 98d53'40.94"E, 18d36'27.38"N)
Lower Right ( 488892.181, 2055503.702) ( 98d53'40.98"E, 18d35'25.22"N)
Center      ( 487980.478, 2056458.796) ( 98d53' 9.85"E, 18d35'56.28"N)
Band 1 Block=63203x1 Type=Byte, ColorInterp=Red
NoData Value=-10000
Mask Flags: PER_DATASET ALPHA
Band 2 Block=63203x1 Type=Byte, ColorInterp=Green
NoData Value=-10000
Mask Flags: PER_DATASET ALPHA
Band 3 Block=63203x1 Type=Byte, ColorInterp=Blue
NoData Value=-10000
Mask Flags: PER_DATASET ALPHA
Band 4 Block=63203x1 Type=Byte, ColorInterp=Alpha
NoData Value=-10000
```

As you can see, this GeoTiff has not been tiled. For accessing subsets though, tiling can make a difference. With tiling, data are stored and compressed in blocks (tiled) rather than line by line (stripped).

In the command output above it is visible that each band has blocks with the same width of the image (63203) and a unit length. The grids in the picture below show an image with equally sized tiles (left) and the same number of strips (right). To read data from the red subset, the intersected area will have to be decompressed.



In the tiled image we will have to decompress only 16 tiles, whereas in the stripped image on the right we'll have to decompress many more strips.

Drone images data usually have a stripped structure so, in most cases, they need to be optimized to increase performances.

Let's take a look at the `gdal_translate` command used to optimize our GeoTiff:

```
gdal_translate -co TILED=YES -co COMPRESS=JPEG -co PHOTOMETRIC=YCBCR
--config GDAL_TIFF_INTERNAL_MASK YES -b 1 -b 2 -b 3 -mask 4
chiangMai_ortho.tif
chiangMai_ortho_optimized.tif
```

Nota: For the details about the command parameters see https://geoserver.geo-solutions.it/edu/en/raster_data/advanced_gdal/example5.html

Once the process ended, call the `gdalinfo` command on the resulting tif file:

```
gdalinfo chiangMai_ortho_optimized.tif
```

The following should be the results:

```
Driver: GTiff/GeoTIFF
Files: chiangMai_ortho_optimized.tif
Size is 63203, 66211
Coordinate System is:
PROJCS["WGS 84 / UTM zone 47N",
  GEOGCS["WGS 84",
    DATUM["WGS_1984",
      SPHEROID["WGS 84",6378137,298.257223563,
        AUTHORITY["EPSG","7030"]],
      AUTHORITY["EPSG","6326"]],
```

(continues on next page)

```

    PRIMEM["Greenwich",0,
        AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
        AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4326"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",99],
    PARAMETER["scale_factor",0.9996],
    PARAMETER["false_easting",500000],
    PARAMETER["false_northing",0],
    UNIT["metre",1,
        AUTHORITY["EPSG","9001"]],
    AXIS["Easting",EAST],
    AXIS["Northing",NORTH],
    AUTHORITY["EPSG","32647"]]]
Origin = (487068.7747500000040513,2057413.8898100000080615)
Pixel Size = (0.028850000000000,-0.028850000000000)
Metadata:
AREA_OR_POINT=Area
TIFFTAG_SOFTWARE=pix4dmapper
Image Structure Metadata:
COMPRESSION=YCbCr JPEG
INTERLEAVE=PIXEL
SOURCE_COLOR_SPACE=YCbCr
Corner Coordinates:
Upper Left  ( 487068.775, 2057413.890) ( 98d52'38.72"E, 18d36'27.34"N)
Lower Left  ( 487068.775, 2055503.702) ( 98d52'38.77"E, 18d35'25.19"N)
Upper Right ( 488892.181, 2057413.890) ( 98d53'40.94"E, 18d36'27.38"N)
Lower Right ( 488892.181, 2055503.702) ( 98d53'40.98"E, 18d35'25.22"N)
Center      ( 487980.478, 2056458.796) ( 98d53' 9.85"E, 18d35'56.28"N)
Band 1 Block=256x256 Type=Byte, ColorInterp=Red
NoData Value=-10000
Mask Flags: PER_DATASET
Band 2 Block=256x256 Type=Byte, ColorInterp=Green
NoData Value=-10000
Mask Flags: PER_DATASET
Band 3 Block=256x256 Type=Byte, ColorInterp=Blue
NoData Value=-10000
Mask Flags: PER_DATASET

```

Our GeoTiff is now tiled with 256x256 tiles, has 3 bands and a 1-bit mask for nodata.

We can also add internal overviews to the file using the `gdaladdo` command:

```
gdaladdo -r average chiangMai_ortho_optimized.tif 2 4 8 16 32 64 128 256 512
```

Overviews are duplicate versions of your original data, but resampled to a lower resolution, they can also be compressed with various algorithms, much in the same way as the original dataset.

By default, overviews take the same compression type and transparency masks of the input dataset (applied through the `gdal_translate` command), so the parameters to be specified are:

- `-r average`: computes the average of all non-NODATA contributing pixels
- `2 4 8 16 32 64 128 256 512`: the list of integral overview levels to build (from gdal version 2.3 levels are no longer required to build overviews)

Calling the `gdalinfo` command again:

```
gdalinfo chiangMai_ortho_optimized.tif
```

It results in:

```
Driver: GTiff/GeoTIFF
Files: chiangMai_ortho_optimized.tif
Size is 63203, 66211
Coordinate System is:
PROJCS["WGS 84 / UTM zone 47N",
  GEOGCS["WGS 84",
    DATUM["WGS_1984",
      SPHEROID["WGS 84",6378137,298.257223563,
        AUTHORITY["EPSG","7030"]],
      AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
      AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
      AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4326"]],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["latitude_of_origin",0],
  PARAMETER["central_meridian",99],
  PARAMETER["scale_factor",0.9996],
  PARAMETER["false_easting",500000],
  PARAMETER["false_northing",0],
  UNIT["metre",1,
    AUTHORITY["EPSG","9001"]],
  AXIS["Easting",EAST],
  AXIS["Northing",NORTH],
  AUTHORITY["EPSG","32647"]]]
Origin = (487068.774750000040513,2057413.889810000080615)
Pixel Size = (0.028850000000000,-0.028850000000000)
Metadata:
AREA_OR_POINT=Area
TIFFTAG_SOFTWARE=pix4dmapper
Image Structure Metadata:
COMPRESSION=YCbCr JPEG
INTERLEAVE=PIXEL
SOURCE_COLOR_SPACE=YCbCr
Corner Coordinates:
Upper Left  ( 487068.775, 2057413.890) ( 98d52'38.72"E, 18d36'27.34"N)
Lower Left  ( 487068.775, 2055503.702) ( 98d52'38.77"E, 18d35'25.19"N)
Upper Right ( 488892.181, 2057413.890) ( 98d53'40.94"E, 18d36'27.38"N)
Lower Right ( 488892.181, 2055503.702) ( 98d53'40.98"E, 18d35'25.22"N)
Center      ( 487980.478, 2056458.796) ( 98d53' 9.85"E, 18d35'56.28"N)
Band 1 Block=256x256 Type=Byte, ColorInterp=Red
NoData Value=-10000
Overviews: 31602x33106, 15801x16553, 7901x8277, 3951x4139, 1976x2070, 988x1035, ↵
↳494x518, 247x259, 124x130
Mask Flags: PER_DATASET
Overviews of mask band: 31602x33106, 15801x16553, 7901x8277, 3951x4139, 1976x2070, ↵
↳988x1035, 494x518, 247x259, 124x130
Band 2 Block=256x256 Type=Byte, ColorInterp=Green
NoData Value=-10000
Overviews: 31602x33106, 15801x16553, 7901x8277, 3951x4139, 1976x2070, 988x1035, ↵
↳494x518, 247x259, 124x130
Mask Flags: PER_DATASET
```

(continues on next page)

(continuação da página anterior)

```

Overviews of mask band: 31602x3Results in:3106, 15801x16553, 7901x8277, 3951x4139,
↳1976x2070, 988x1035, 494x518, 247x259, 124x130
Band 3 Block=256x256 Type=Byte, ColorInterp=Blue
NoData Value=-10000
Overviews: 31602x33106, 15801x16553, 7901x8277, 3951x4139, 1976x2070, 988x1035,
↳494x518, 247x259, 124x130
Mask Flags: PER_DATASET
Overviews of mask band: 31602x33106, 15801x16553, 7901x8277, 3951x4139, 1976x2070,
↳988x1035, 494x518, 247x259, 124x130

```

Notice that the transparency masks of internal overviews have been applied (their compression does not show up in the file metadata).

UAVs usually provide also two other types of data: DTM (Digital Terrain Model) and DSM (Digital Surface Model).

Those data require different processes to be optimized. Let's look at some examples to better understand how to use gdal to accomplish that task.

From the [CHIANG MAI Urban Flooding GeoNode platform](#) it is currently available the `chiangMai_dtm_optimized` layer, let's download its original dataset.

This dataset should contain the DTM file `chiangMai_dtm.tif`.

Calling the `gdalinfo` command on it:

```
gdalinfo chiangMai_dtm.tif
```

The following information will be displayed:

```

Driver: GTiff/GeoTIFF
Files: chiangMai_dtm.tif
Size is 12638, 13240
Coordinate System is:
PROJCS["WGS 84 / UTM zone 47N",
  GEOGCS["WGS 84",
    DATUM["WGS_1984",
      SPHEROID["WGS 84",6378137,298.257223563,
        AUTHORITY["EPSG","7030"]],
      AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
      AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
      AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4326"]],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["latitude_of_origin",0],
  PARAMETER["central_meridian",99],
  PARAMETER["scale_factor",0.9996],
  PARAMETER["false_easting",500000],
  PARAMETER["false_northing",0],
  UNIT["metre",1,
    AUTHORITY["EPSG","9001"]],
  AXIS["Easting",EAST],
  AXIS["Northing",NORTH],
  AUTHORITY["EPSG","32647"]]
Origin = (487068.774750000040513,2057413.889810000080615)
Pixel Size = (0.144270000000000,-0.144270000000000)

```

(continues on next page)

(continuação da página anterior)

```

Metadata:
AREA_OR_POINT=Area
TIFFTAG_SOFTWARE=pix4dmapper
Image Structure Metadata:
COMPRESSION=LZW
INTERLEAVE=BAND
Corner Coordinates:
Upper Left  ( 487068.775, 2057413.890) ( 98d52'38.72"E, 18d36'27.34"N)
Lower Left  ( 487068.775, 2055503.755) ( 98d52'38.77"E, 18d35'25.19"N)
Upper Right ( 488892.059, 2057413.890) ( 98d53'40.94"E, 18d36'27.37"N)
Lower Right ( 488892.059, 2055503.755) ( 98d53'40.98"E, 18d35'25.22"N)
Center      ( 487980.417, 2056458.822) ( 98d53' 9.85"E, 18d35'56.28"N)
Band 1 Block=12638x1 Type=Float32, ColorInterp=Gray
NoData Value=-10000

```

Reading this image could be very slow because it has not been tiled yet. So, as discussed above, its data need to be stored and compressed in tiles to increase performances.

The following `gdal_translate` command should be appropriate for that purpose:

```

gdal_translate -co TILED=YES -co COMPRESS=DEFLATE chiangMai_dtm.tif chiangMai_dtm_
→optimized.tif

```

When the data to compress consists of imagery (es. aerial photographs, true-color satellite images, or colored maps) you can use lossy algorithms such as JPEG. We are now compressing data where the precision is important, the band data type is Float32 and elevation values should not be altered, so a lossy algorithm such as JPEG is not suitable. JPEG should generally only be used with Byte data (8 bit per channel) so we have choosen the lossless DEFLATE compression through the `COMPRESS=DEFLATE` creation option.

Calling the `gdalinfo` command again:

```

gdalinfo chiangMai_dtm_optimized.tif

```

We can observe the following results:

```

Driver: GTiff/GeoTIFF
Files: chiangMai_dtm_optimized.tif
Size is 12638, 13240
Coordinate System is:
PROJCS["WGS 84 / UTM zone 47N",
    GEOGCS["WGS 84",
        DATUM["WGS_1984",
            SPHEROID["WGS 84",6378137,298.257223563,
                AUTHORITY["EPSG","7030"]],
            AUTHORITY["EPSG","6326"]],
        PRIMEM["Greenwich",0,
            AUTHORITY["EPSG","8901"]],
        UNIT["degree",0.0174532925199433,
            AUTHORITY["EPSG","9122"]],
        AUTHORITY["EPSG","4326"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",99],
    PARAMETER["scale_factor",0.9996],
    PARAMETER["false_easting",500000],
    PARAMETER["false_northing",0],
    UNIT["metre",1,

```

(continues on next page)

(continuação da página anterior)

```

    AUTHORITY["EPSG","9001"]],
    AXIS["Easting",EAST],
    AXIS["Northing",NORTH],
    AUTHORITY["EPSG","32647"]]]
Origin = (487068.7747500000040513,2057413.8898100000080615)
Pixel Size = (0.1442700000000000,-0.1442700000000000)
Metadata:
AREA_OR_POINT=Area
TIFFTAG_SOFTWARE=pix4dmapper
Image Structure Metadata:
COMPRESSION=DEFLATE
INTERLEAVE=BAND
Corner Coordinates:
Upper Left  ( 487068.775, 2057413.890) ( 98d52'38.72"E, 18d36'27.34"N)
Lower Left  ( 487068.775, 2055503.755) ( 98d52'38.77"E, 18d35'25.19"N)
Upper Right ( 488892.059, 2057413.890) ( 98d53'40.94"E, 18d36'27.37"N)
Lower Right ( 488892.059, 2055503.755) ( 98d53'40.98"E, 18d35'25.22"N)
Center      ( 487980.417, 2056458.822) ( 98d53' 9.85"E, 18d35'56.28"N)
Band 1 Block=256x256 Type=Float32, ColorInterp=Gray
NoData Value=-10000

```

We need also to create overviews through the `gdaladdo` command:

```
gdaladdo -r nearest chiangMai_dtm_optimized.tif 2 4 8 16 32 64
```

Unlike the previous example, overviews will be created with the **nearest resampling algorithm**. That is due to the nature of the data we are representing: we should not consider the average between two elevation values but simply the closer one, it is more reliable regarding the conservation of the original data.

Calling the `gdalinfo` command again:

```
gdalinfo chiangMai_dtm_optimized.tif
```

We can see the following information:

```

Driver: GTiff/GeoTIFF
Files: chiangMai_dtm_optimized.tif
Size is 12638, 13240
Coordinate System is:
PROJCS["WGS 84 / UTM zone 47N",
    GEOGCS["WGS 84",
        DATUM["WGS_1984",
            SPHEROID["WGS 84",6378137,298.257223563,
                AUTHORITY["EPSG","7030"]],
            AUTHORITY["EPSG","6326"]],
        PRIMEM["Greenwich",0,
            AUTHORITY["EPSG","8901"]],
        UNIT["degree",0.0174532925199433,
            AUTHORITY["EPSG","9122"]],
        AUTHORITY["EPSG","4326"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",99],
    PARAMETER["scale_factor",0.9996],
    PARAMETER["false_easting",500000],
    PARAMETER["false_northing",0],
    UNIT["metre",1,

```

(continues on next page)

(continuação da página anterior)

```

    AUTHORITY["EPSG","9001"]],
    AXIS["Easting",EAST],
    AXIS["Northing",NORTH],
    AUTHORITY["EPSG","32647"]]]
Origin = (487068.7747500000040513,2057413.8898100000080615)
Pixel Size = (0.1442700000000000,-0.1442700000000000)
Metadata:
AREA_OR_POINT=Area
TIFFTAG_SOFTWARE=pix4dmapper
Image Structure Metadata:
COMPRESSION=DEFLATE
INTERLEAVE=BAND
Corner Coordinates:
Upper Left  ( 487068.775, 2057413.890) ( 98d52'38.72"E, 18d36'27.34"N)
Lower Left  ( 487068.775, 2055503.755) ( 98d52'38.77"E, 18d35'25.19"N)
Upper Right ( 488892.059, 2057413.890) ( 98d53'40.94"E, 18d36'27.37"N)
Lower Right ( 488892.059, 2055503.755) ( 98d53'40.98"E, 18d35'25.22"N)
Center      ( 487980.417, 2056458.822) ( 98d53' 9.85"E, 18d35'56.28"N)
Band 1 Block=256x256 Type=Float32, ColorInterp=Gray
NoData Value=-10000
Overviews: 6319x6620, 3160x3310, 1580x1655, 790x828, 395x414, 198x207

```

Overviews have been created. By default, they inherit the same compression type of the original dataset (there is no evidence of it in the `gdalinfo` output).

Other Raster Data Use Cases

- Serving a large number of GrayScale GeoTiff with Palette
- Serving a large number of DTM ASCII Grid Files
- Serving a large number of Cartographic Black/White GeoTiff with Palette
- Serving a large number of satellite/aerial RGB GeoTiff with compression
- Optimizing and serving UAV data
- Optimizing and serving 16-bits satellite/aerial RGB GeoTiff

Process Raster Datasets Programmatically

In this section we will provide a set of *shell* scripts which might be very useful to batch process a lot of raster datasets programmatically.

1. process_gray.sh

```

for filename in *.tif*; do echo gdal_translate -co TILED=YES -co
→COMPRESS=DEFLATE $filename ${filename}/*.tif/.optimized.tif; done >
→gdal_translate.sh
chmod +x gdal_translate.sh
./gdal_translate.sh

```

```

for filename in *.optimized.tif*; do echo gdaladdo -r nearest $filename
→2 4 8 16 32 64 128 256 512; done > gdaladdo.sh
for filename in *.optimized.tif*; do echo mv \"$filename\" \"${filename}
→.optimized.tif/.tif/\"; done > rename.sh

```

(continues on next page)

(continuação da página anterior)

```
chmod +x *.sh
./gdaladdo.sh
./rename.sh
```

2. process_rgb.sh

```
for filename in *.tif*; do echo gdal_translate -co TILED=YES -co_
→COMPRESS=JPEG -co PHOTOMETRIC=YCBCR -b 1 -b 2 -b 3 $filename $
→{filename}/*.tif/optimized.tif; done > gdal_translate.sh
chmod +x gdal_translate.sh
./gdal_translate.sh
```

```
for filename in *.optimized.tif*; do echo gdaladdo -r average $filename_
→2 4 8 16 32 64 128 256 512; done > gdaladdo.sh
for filename in *.optimized.tif*; do echo mv \"$filename\" \"${filename}/
→optimized.tif/optimized.tif\"; done > rename.sh
chmod +x *.sh
./gdaladdo.sh
./rename.sh
```

3. process_rgb_alpha.sh

```
for filename in *.tif*; do echo gdal_translate -co TILED=YES -co_
→COMPRESS=JPEG -co PHOTOMETRIC=YCBCR --config GDAL_TIFF_INTERNAL_MASK_
→YES -b 1 -b 2 -b 3 -mask 4 $filename ${filename}/*.tif/optimized.tif;_
→done > gdal_translate.sh
chmod +x gdal_translate.sh
./gdal_translate.sh
```

```
for filename in *.optimized.tif*; do echo gdaladdo -r average $filename_
→2 4 8 16 32 64 128 256 512; done > gdaladdo.sh
for filename in *.optimized.tif*; do echo mv \"$filename\" \"${filename}/
→optimized.tif/optimized.tif\"; done > rename.sh
chmod +x *.sh
./gdaladdo.sh
./rename.sh
```

4. process_rgb_palette.sh

```
for filename in *.tif*; do echo gdal_translate -co TILED=YES -co_
→COMPRESS=DEFLATE $filename ${filename}/*.tif/optimized.tif; done >_
→gdal_translate.sh
chmod +x gdal_translate.sh
./gdal_translate.sh
```

```
for filename in *.optimized.tif*; do echo gdaladdo -r average $filename_
→2 4 8 16 32 64 128 256 512; done > gdaladdo.sh
for filename in *.optimized.tif*; do echo mv \"$filename\" \"${filename}/
→optimized.tif/optimized.tif\"; done > rename.sh
chmod +x *.sh
./gdaladdo.sh
./rename.sh
```

1.25.4 Create Users and Super Users

Your first step will be to create a user. There are three options to do so, depending on which kind of user you want to create you may choose a different option. We will start with creating a *superuser*, because this user is the most important. A superuser has all the permissions without explicitly assigning them.

The easiest way to create a superuser (in linux) is to open your terminal and type:

```
$ DJANGO_SETTINGS_MODULE=geonode.settings python manage.py createsuperuser
```

Nota: If you enabled `local_settings.py` the command will change as following:

```
$ DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py_
↪createsuperuser
```

You will be asked a username (in this tutorial we will call the superuser you now create *your_superuser*), an email address and a password.

Now you've created a superuser you should become familiar with the *Django Admin Interface*. As a superuser you are having access to this interface, where you can manage users, layers, permission and more. To learn more detailed about this interface check this [LINK](#). For now it will be enough to just follow the steps. To attend the *Django Admin Interface*, go to your geonode website and *sign in* with *your_superuser*. Once you've logged in, the name of your user will appear on the top right. Click on it and the following menu will show up:

Clicking on *Admin* causes the interface to show up.

Go to *Auth -> Users* and you will see all the users that exist at the moment. In your case it will only be *your_superuser*. Click on it, and you will see a section on *Personal Info*, one on *Permissions* and one on *Important dates*. For the moment, the section on *Permissions* is the most important.

As you can see, there are three boxes that can be checked and unchecked. Because you've created a superuser, all three boxes are checked as default. If only the box *active* would have been checked, the user would not be a superuser and would not be able to access the *Django Admin Interface* (which is only available for users with the *staff* status). Therefore keep the following two things in mind:

- a superuser is able to access the *Django Admin Interface* and he has all permissions on the data uploaded to GeoNode.
- an ordinary user (created from the GeoNode interface) only has *active* permissions by default. The user will not have the ability to access the *Django Admin Interface* and certain permissions have to be added for him.

Until now we've only created superusers. So how do you create an ordinary user? You have two options:

1. Django Admin Interface

First we will create a user via the *Django Admin Interface* because we've still got it open. Therefore go back to *Auth -> Users* and you should find a button on the right that says *Add user*.

Click on it and a form to fill out will appear. Name the new user *test_user*, choose a password and click *save* at the right bottom of the site.

Now you should be directed to the site where you could change the permissions on the user *test_user*. As default only *active* is checked. If you want this user also to be able to attend this admin interface you could also check *staff status*. But for now we leave the settings as they are!

Menu



-  Upload Layers
-  Profile
-  Recent Activity
-  Inbox
-  Announcements
-  Remote Services
-  Invite User
-  GeoServer
-  Admin
-  Help

Log out

Django administration

Site administration

Account		
Account deletions	+ Add	✎ Change
Accounts	+ Add	✎ Change
Signup codes	+ Add	✎ Change
Actstream		
Actions	+ Add	✎ Change
Follows	+ Add	✎ Change
Announcements		
Announcements	+ Add	✎ Change
Dismissals	+ Add	✎ Change
Auth		
Groups	+ Add	✎ Change
Users	+ Add	✎ Change
Avatar		
Avatars	+ Add	✎ Change
Base		
Contact roles	+ Add	✎ Change
Links	+ Add	✎ Change
Metadata Regions	+ Add	✎ Change

Permissions	
☒ Active	Designates whether this user should be treated as active. Unselect this instead of deleting accounts.
☒ Staff status	Designates whether the user can log into this admin site.
☒ Superuser status	Designates that this user has all permissions without explicitly assigning them.

Django administration

Welcome, **barbara**. [Change password](#) / [Log out](#)

Home > Auth > Users

Select user to change Add user +

Django administration

Welcome, **barbara**. [Change password](#) / [Log out](#)

Home > Auth > Users > Add user

Add user

First, enter a username and password. Then, you'll be able to edit more user options.

Username:
Required. 30 characters or fewer. Letters, digits and @/./+/-/_ only.

Password:

Password confirmation:
Enter the same password as above, for verification.

[Save and add another](#) [Save and continue editing](#) [Save](#)

To test whether the new user was successfully created, go back to the GeoNode web page and try to sign in.

2. GeoNode website

To create an ordinary user you could also just use the GeoNode website. If you installed GeoNode using a release, you should see a *Register* button on the top, beside the *Sign in* button (you might have to log out before).



Hit the button and again a form will appear for you to fill out. This user will be named *geonode_user*

By hitting *Sign up* the user will be signed up, as default only with the status *active*.

1.25.5 Batch Sync Permissions

GeoNode provides a very useful management command `set_layers_permissions` allowing an administrator to easily add / remove permissions to groups and users on one or more layers.

The `set_layers_permissions` command arguments are:

- **permissions** to set/unset → read (r), write (w), download (d), owner (o)

```

READ_PERMISSIONS = [
    'view_resourcebase'
]
WRITE_PERMISSIONS = [
    'change_layer_data',
    'change_layer_style',
    'change_resourcebase_metadata'
]
DOWNLOAD_PERMISSIONS = [
    'download_resourcebase'
]
OWNER_PERMISSIONS = [
    'change_resourcebase',
    'delete_resourcebase',
    'change_resourcebase_permissions',
    'publish_resourcebase'
]

```

SIGN UP

Username

Password

Password (again)

Email

[Sign up](#)

- **resources** (layers) which permissions will be assigned on → type the layer title (use quotation mark for titles with white space), multiple choices can be typed with white space separator, if no titles are provided all the layers will be considered
- **users** who permissions will be assigned to, multiple choices can be typed with a white space separator
- **groups** who permissions will be assigned to, multiple choices can be typed with a white space separator
- **delete** flag (optional) which means the permissions will be unset

Usage examples:

1. Assign **write** permissions on the layers **layer_X** and **layer Y** to the users **user_A** and **user_B** and to the group **group_C**.

```
python manage.py set_layers-permissions -p write -u user_A user_B -g_
↪group_C -r layer_X 'layer Y'
```

2. Assign **owner** permissions on all the layers to the group **group_C**.

```
python manage.py set_layers-permissions -p owner -g group_C
```

3. Unset **download** permissions on the layer **layer_X** for the user **user_A**.

```
python manage.py set_layers-permissions -p download -u user_A -r layer_X_
↪-d
```

The same functionalities, with some limitations, are available also from the *Admin Dashboard >> Layers*.

An action named *Set layers permissions* is available from the list, redirecting the administrator to a form to set / unset read, write, download and ownership permissions on the selected layers.

Django administration

Home › Layers › Layers

Select layer to change

Q Search

◀ 2019 November 14

Action: Set Layers Permissions ▼ Go 2 of 2 selected

☒	ID	ALTERNATE	TITLE [EN]	DATE	CATE
☒	28	geonode:tasmania_water_bodies	tasmania_water_bodies	Nov. 14, 2019, 3:36 p.m.	---
☒	27	geonode:tasmania_state_boundaries	tasmania_state_boundaries	Nov. 14, 2019, 3:36 p.m.	---



Layers Permissions

Group
 ▼

User
 ▼

Permission Type

☒ Read
☐ Write
☐ Download

Mode

☒ Set
☐ Unset

Cancel Submit

1.25.6 Delete Certain GeoNode Resources

The `delete_resources` *Management Command* allows to remove resources meeting a certain condition, specified in a form of a serialized django Q() expression.

First of all let's take a look at the `--help` option of the `delete_resources` management command in order to inspect all the command options and features.

Run

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py delete_resources --help
```

Nota: If you enabled `local_settings.py` the command will change as following:

```
DJANGO_SETTINGS_MODULE=geonode.local_settings python manage.py delete_resources --help
```

This will produce output the following output:

```
usage: manage.py delete_resources [-h] [-c CONFIG_PATH]
                                  [-l LAYER_FILTERS [LAYER_FILTERS ...]]
                                  [-m MAP_FILTERS [MAP_FILTERS ...]]
                                  [-d DOCUMENT_FILTERS [DOCUMENT_FILTERS ...]]
                                  [--version] [-v {0,1,2,3}]
                                  [--settings SETTINGS]
                                  [--pythonpath PYTHONPATH] [--traceback]
                                  [--no-color] [--force-color]

Delete resources meeting a certain condition

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG_PATH, --config CONFIG_PATH
                        Configuration file path. Default is:
                        delete_resources.json
  -l LAYER_FILTERS [LAYER_FILTERS ...], --layer_filters LAYER_FILTERS [LAYER_FILTERS .
  ↪ ..]
  -m MAP_FILTERS [MAP_FILTERS ...], --map_filters MAP_FILTERS [MAP_FILTERS ...]
  -d DOCUMENT_FILTERS [DOCUMENT_FILTERS ...], --document_filters DOCUMENT_FILTERS_
  ↪ [DOCUMENT_FILTERS ...]
  --version            show program's version number and exit
  -v {0,1,2,3}, --verbosity {0,1,2,3}
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
  --settings SETTINGS  The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
  --pythonpath PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
  --traceback          Raise on CommandError exceptions
  --no-color           Don't colorize the command output.
  --force-color        Force colorization of the command output.
```

There are two ways to declare Q() expressions filtering which resources should be deleted:

1. With a JSON configuration file: passing `-c` argument specifying the path to the JSON configuration file.

- **Example 1:** Relative path to the config file (to `manage.py`)

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py delete_
↪resources -c geonode/base/management/commands/delete_resources.json
```

- **Example 2:** Absolute path to the config file

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py delete_
↪resources -c /home/User/Geonode/configs/delete_resources.json
```

2. With CLI: passing `-l -d -m` list arguments for each of resources (layers, documents, maps)

- **Example 3:** Delete resources without configuration file

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py delete_
↪resources -l 'Q(pk__in: [1, 2]) | Q(title__icontains:"italy")'
↪'Q(owner__name=admin)' -d '*' -m "Q(pk__in=[1, 2])"
```

Configuration File

The JSON configuration file should contain a single *filters* object, which consists of *layer*, *map* and *document* lists. Each list specifies the filter conditions applied to a corresponding queryset, defining which items will be deleted. The filters are evaluated and directly inserted into Django `.filter()` method, which means the filters occurring as separated list items are treated as AND condition. To create OR query `|` operator should be used. For more info please check Django [documentation](<https://docs.djangoproject.com/en/3.2/topics/db/queries/#complex-lookups-with-q-objects>). The only exception is passing a list with `'*'` which will cause deleting all the queryset of the resource.

- **Example 4:** Example content of the configuration file, which will delete layers with ID's 1, 2, and 3, those owned by *admin* user, along with all defined maps.

```
{
  "filters": {
    "layer": [
      "Q(pk__in=[1, 2, 3]) | Q(title__icontains='italy')",
      "Q(user__name=admin)"
    ],
    "map": ["*"],
    "document": []
  }
}
```

CLI

The CLI configuration can be specified with `-l -d -m` list arguments, which in fact are a translation of the configuration JSON file. `-l -d -m` arguments are evaluated in the same manner as `filters.layer`, `filters.map` and `filter.document` accordingly from the Example 4. The following example's result will be equivalent to Example 4:

- **Example 5:** Example CLI configuration, which will delete layers with ID's 1, 2, and 3, along with all maps.

```
DJANGO_SETTINGS_MODULE=geonode.settings python manage.py delete_
↪resources -l 'Q(pk__in: [1, 2, 3]) | Q(title__icontains:"italy")'
↪'Q(owner__name=admin)' -m '*'
```

1.26 Changing the default Languages

1.26.1 Changing the Default Language

GeoNode's default language is English, but GeoNode users can change the interface language with the pulldown menu at the top-right of most GeoNode pages. Once a user selects a language GeoNode remembers that language for subsequent pages.

1.26.2 GeoNode Configuration

As root edit the geonode config file `/home/geonode/geonode/geonode/settings.py` (or `/etc/geonode/settings.py` if GeoNode has been installed using **apt-get**) and change `LANGUAGE_CODE` to the desired default language.

Nota: A list of language codes can be found in the global django config file `/usr/local/lib/python2.7/dist-packages/django/conf/global_settings.py` (or `/var/lib/geonode/lib/python2.7/site-packages/django/conf/global_settings.py` if GeoNode has been installed using **apt-get**).

For example, to make French the default language use:

```
LANGUAGE_CODE = 'fr'
```

Unfortunately Django overrides this setting, giving the language setting of a user's browser priority. For example, if `LANGUAGE_CODE` is set to French, but the user has configured their operating system for Spanish they may see the Spanish version when they first visit GeoNode.

1.26.3 Additional Steps

If this is not the desired behaviour, and all users should initially see the default `LANGUAGE_CODE`, regardless of their browser's settings, do the following steps to ensure Django ignores the browser language settings. (Users can always use the pulldown language menu to change the language at any time.)

As **root** create a new directory within GeoNode's site packages

```
mkdir /usr/lib/python2.7/dist-packages/setmydefaultlanguage
```

or

```
mkdir /var/lib/geonode/lib/python2.7/site-packages/setmydefaultlanguage
```

if GeoNode has been installed using **apt-get**.

As root create and edit a new file `/usr/lib/python2.7/dist-packages/setmydefaultlanguage/__init__.py` and add the following lines

```
class ForceDefaultLanguageMiddleware(object):
    """
    Ignore Accept-Language HTTP headers

    This will force the I18N machinery to always choose settings.LANGUAGE_CODE
    as the default initial language, unless another one is set via sessions or cookies
```

(continues on next page)

(continuação da página anterior)

```
Should be installed *before* any middleware that checks request.META['HTTP_ACCEPT_
↪LANGUAGE'],
namely django.middleware.locale.LocaleMiddleware
"""
def process_request(self, request):
    if request.META.has_key('HTTP_ACCEPT_LANGUAGE'):
        del request.META['HTTP_ACCEPT_LANGUAGE']
```

At the end of the GeoNode configuration file `/home/geonode/geonode/geonode/settings.py` (or `/etc/geonode/settings.py` if GeoNode has been installed using **apt-get**) add the following lines to ensure the above class is executed

```
MIDDLEWARE_CLASSES += (
    'setmydefaultlanguage.ForceDefaultLanguageMiddleware',
)
```

1.26.4 Restart

You will need to restart GeoNode accordingly to the installation method you have choosen.

As an instance in case you are using *NGINX* with *UWSGI*, as root you will need to run the following commands

```
service uwsgi restart
service nginx restart
```

Please refer to Translating GeoNode for information on editing GeoNode pages in different languages and create new GeoNode Translations.

1.27 GeoNode Upgrade from older versions

1.27.1 Upgrade from 3.1.x

1. Upgrade the dependencies
2. Perform the `migrations` management command; in case some attribute is conflicting, remove it manually from the DB
3. Proform the `collectstatic` management command
4. Perform the `set_all_layers_metadata -d` management command
5. Drop the `rabbitmq` image and volume and let GeoNode recreate the `queues` automatically

1.27.2 Upgrade from 2.10.x / 3.0

Upgrade the instance dependencies

Check the *1. Install the dependencies* and *2. GeoNode Installation* sections in order to upgrade your Python environment.

Also, make sure the code is Python 3.8 compatible and that you switched and aligned the **source code** and the **requirements.txt** to the 3.x branch.

This must be done manually and with particular attention.

```
workon geonode3
cd /<full_path_to_geonode>

pip install pip --upgrade
pip install -r requirements.txt --upgrade --no-cache --no-cache-dir
pip install -e . --upgrade
pip install pygdal=="`gdal-config --version`.*"

./manage.sh collectstatic --noinput
```

Prepare the Database and Migrate to the new Schema

Fix the tables in order to migrate to the new schema

```
./manage.sh dbshell
```

```
ALTER TABLE base_resourcebase ADD COLUMN doi_bkp varchar;
UPDATE base_resourcebase SET doi_bkp = doi;
ALTER TABLE base_resourcebase DROP COLUMN doi;

CREATE TABLE base_backup(name varchar);

CREATE TABLE base_usergeolimit_bkp ( like base_usergeolimit including all);
CREATE TABLE base_groupgeolimit_bkp ( like base_usergeolimit including all);
CREATE TABLE base_resourcebase_users_geolimits_bkp ( like base_usergeolimit including
↳all);
CREATE TABLE base_resourcebase_groups_geolimits_bkp ( like base_usergeolimit
↳including all);

DROP TABLE IF EXISTS base_configuration CASCADE;
DROP TABLE IF EXISTS base_usergeolimit CASCADE;
DROP TABLE IF EXISTS base_groupgeolimit CASCADE;
DROP TABLE IF EXISTS base_resourcebase_users_geolimits CASCADE;
DROP TABLE IF EXISTS base_resourcebase_groups_geolimits CASCADE;

\q
```

Migrate to the new schema

```
./manage.sh makemigrations
./manage.sh migrate
```

Restore the old contents

```
./manage.sh dbshell
```

```
UPDATE base_resourcebase SET doi = doi_bkp;
ALTER TABLE base_resourcebase DROP COLUMN doi_bkp;

INSERT INTO base_usergeolimit (SELECT * FROM base_usergeolimit_bkp);
INSERT INTO base_groupgeolimit (SELECT * FROM base_groupgeolimit_bkp);
INSERT INTO base_resourcebase_users_geolimits (SELECT * FROM base_resourcebase_users_
↳geolimits_bkp);
INSERT INTO base_resourcebase_groups_geolimits (SELECT * FROM base_resourcebase_
↳groups_geolimits_bkp);

DROP TABLE IF EXISTS base_usergeolimit_bkp CASCADE;
DROP TABLE IF EXISTS base_groupgeolimit_bkp CASCADE;
DROP TABLE IF EXISTS base_resourcebase_users_geolimits_bkp CASCADE;
DROP TABLE IF EXISTS base_resourcebase_groups_geolimits_bkp CASCADE;

\q
```

1.27.3 Upgrade from 2.4.x

These are the notes of a migration from 2.4.x to 2.10.1. These notes could possibly work also when migrating from 2.6.x, 2.7.x, 2.8.x but are not tested in that scenarios. You should run this procedure on your local machine and once you successfully migrated the database move the backup to your GeoNode 2.10.1 production instance.

PostgreSQL

Create a role and a database for Django GeoNode 2.4:

```
create role user with superuser login with password '***';
create database gn_24 with owner user;
\c gn_24
create extension postgis;
```

Restore backup from your production backup:

```
psql gn_24 < gn_24.sql
```

Run GeoNode migrations

Activate your GeoNode virtualenv and set the env vars:

```
. env/bin/Activate
export vars_210
```

Here are the variables to export - update them to your environment settings:

```
export DATABASE_URL=postgis://user:***@localhost:5432/dbname
export DEFAULT_BACKEND_DATASTORE=data
export GEODATABASE_URL=postgis://user:***@localhost:5432/geonode_data
export ALLOWED_HOSTS="['localhost', '192.168.100.10']"
export STATIC_ROOT=~/.www/geonode/static/
export GEOSERVER_LOCATION=http://localhost:8080/geoserver/
export GEOSERVER_PUBLIC_LOCATION=http://localhost:8080/geoserver/
export GEOSERVER_ADMIN_PASSWORD=geoserver
export SESSION_EXPIRED_CONTROL_ENABLED=False
```

Downgrade psycopg2:

```
pip install psycopg2==2.7.7
```

Apply migrations and apply basic fixtures:

```
cd wfp-geonode
./manage.py migrate --fake-initial
paver sync
```

Regenerate from scratch the upload application tables in the database:

```
delete from django_migrations where app = 'upload';
drop table upload_upload cascade;
drop table upload_uploadfile;
```

Regenerate upload tables with migrate:

```
./manage.py migrate upload
```

Upgrade psycopg2:

```
pip install -r geonode/requirements.txt
```

Create superuser

To create a superuser you should drop the following constraints (they can be re-enabled if needed):

```
alter table people_profile alter column last_login drop not null;
```

```
./manage.py createsuperuser
```

Fixes on database

For some reason some resources were unpublished:

```
UPDATE base_resourcebase SET is_published = true;
```

Remove a foreign key from account_account which is not used anymore (GeoNode dev team: maybe even better let's remove all of the account tables, I think they are stale now):

```
ALTER TABLE account_account DROP CONSTRAINT user_id_refs_id_726cb6b4;  
ALTER TABLE account_signupcode DROP CONSTRAINT "inviter_id_refs_id_49a7c0d9";
```

Fix the remote service layers by running this script:

```
python migration/fixes_remote_layers.py
```

1.28 GeoNode Async Signals

1.28.1 Supervisord and Systemd

1.28.2 Celery

1.28.3 Rabbitmq and Redis

1.28.4 How to: Async Upload via API

In geonode is possible to upload resources via API in async/sync way.

Here is available a full example of upload via API <https://github.com/GeoNode/geonode/blob/582d6efda74adb8042d1d897004bbf764e6e0285/geonode/upload/api/tests.py#L416>

Step 1

Create a common client session, this is fundamental due the fact that geonode will check the request session. For example with requests we will do something like:

```
import requests  
client = requests.session()
```

Note: in Django this part is already managed

Step 2

Call the `api/v2/uploads/upload` endpoint in PUT (is a form-data endpoint) by specifying in files a dictionary with the names and the files that we want to uploads and a data payload with the required informations. For example:

```
params = {  
    "permissions": '{ "users": {"AnonymousUser": ["view_resourcebase"]} , "groups":{}}  
↪', # layer permissions  
    "time": "false",  
    "layer_title": "layer_title",  
}
```

(continues on next page)

(continuação da página anterior)

```

        "time": "false",
        "charset": "UTF-8",
    }

files = {
    "filename": <_io.BufferedReader name="filename">
}

client.put(
    "http://localhost:8000/api/v2/uploads/upload/",
    auth=HTTPBasicAuth(username, password),
    data=params,
    files=files,
)

Returns:
- dict with import id of the resource

```

Step 3

Call in the final upload page in order to trigger the actual import. If correctly set, Geoserver will manage the upload asynchronously.

```
client.get("http://localhost:8000/upload/final?id={import_id}")
```

The `import\_id` is returned from the previous step

Step 4

The upload as been completed on GeoNode, we should check until Geoserver has complete his part. To do so, is enough to call the detailed information about the upload that we are performing

```
client.get(f"http://localhost:8000/api/v2/uploads/{upload_id}")
```

When the status is *PROCESSED* and the completion is *100%* we are able to see the resource in geonode and geoserver

1.29 GeoNode add a thesaurus

1.29.1 Loading a thesaurus

There are 2 possible ways to upload a Thesaurus in geonode:

- Admin panel
- Django command-line
- settings.py (deprecated)

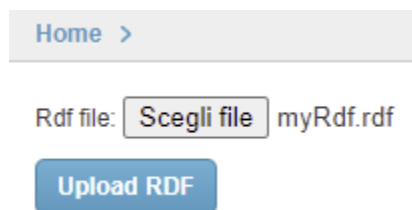
1.29.2 Admin panel

You can add a thesaurus into you GeoNode using the `upload thesaurus` available in the Admin panel

Navigate to the thesaurus page in the admin panel `http://<your_geonode_host>/admin/base/thesaurus`. On the top-right of the page a button named *Upload thesaurus* will be available:

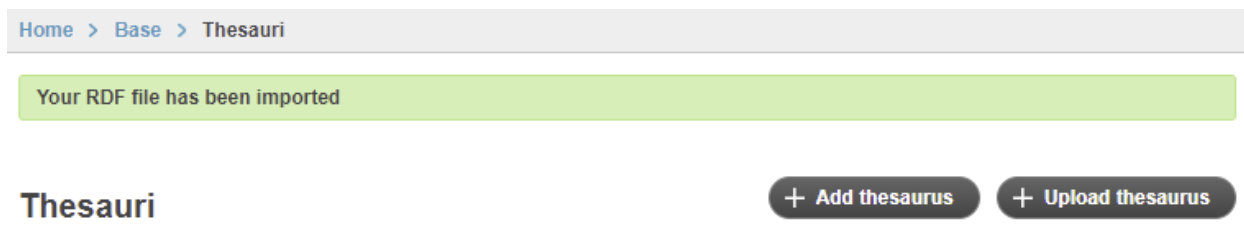


After clicking on it, a simple form for the upload will be shown. In order to let the upload works, is required to choose an *RDF* file



By clicking on *Upload CSV*, the system will load the thesaurus by assigning to it a *slugify* name based on the file name. The name can be easily change later in the edit page.

If everything goes fine, a successfull message will be shown:



Otherwise the UI will show the error message:

1.29.3 Command line

A thesaurus can be loaded into GeoNode by using the `load_thesaurus` command:

```
python manage.py load_thesaurus --help

-d, --dry-run          Only parse and print the thesaurus file, without perform
↳ insertion in the DB.
--name=NAME            Identifier name for the thesaurus in this GeoNode instance.
--file=FILE            Full path to a thesaurus in RDF format.
```

In order add the inspire-themes thesaurus into a geonode instance, download it as file `inspire-theme.rdf` with the command:

[Home](#) > [Base](#) > [Thesauri](#)

duplicate key value violates unique constraint "base_thesaurus_identifier_key" DETAIL: Key (identifier)=(myrdf-rdf) already exists.

Thesauri

+ Add thesaurus

+ Upload thesaurus

```
wget -O inspire-theme.rdf https://raw.githubusercontent.com/geonetwork/core-geonetwork/master/web/src/test/resources/thesaurus/external/thesauri/theme/httpinspireeuropaetheme-theme.rdf
```

and then issue the command:

```
python manage.py load_thesaurus --file inspire-theme.rdf --name inspire_themes
```

The name is the identifier you'll use to refer to this thesaurus in your GeoNode instance.

If you only want to make sure that a thesaurus file will be properly parsed, give the `--dry-run` parameter, so that nothing will be added to the DB.

Note: if the name starts with the string `fake`, the file will not be accessed at all, and some test keywords will be added to a fake new thesaurus. In this case the `dry-run` param will not be used.

1.29.4 Configure a thesaurus in GeoNode

Configuration from *Admin*

After you loaded a thesaurus into GeoNode, it should be configured in the *Admin* panel.

The panel can be reached from *Admin* link of the *User Menu* in the navigation bar or through this URL: `http://<your_geonode_host>/admin/base/thesaurus`.

Once you are on the Thesaurus lists, select one thesaurus to open the Edit page

- **identifier:** (mandatory string) the identifier you used in the `load_thesaurus` commands.
- **title:** (mandatory string) The title of the thesaurus, is ingested by the `load_thesaurus` command.
- **date:** (mandatory date) The Date of the thesaurus, is ingested by the `load_thesaurus` command.
- **description:** (mandatory string) The description of the thesaurus, is ingested by the `load_thesaurus` command.
- **slug:** (mandatory string) The slug of the thesaurus, is ingested by the `load_thesaurus` command.
- **about:** (optional string) The about of the thesaurus, is ingested by the `load_thesaurus` command.
- **card min:** (optional integer) Decide the minimum cardinality, default = 0
- **card max:** (optional integer) Decide the maximum cardinality, default = -1
- **facet:** (boolean) Decide if the thesaurus will be shown in the facet list. default: True
- **order:** (integer) Decide the listing order of the thesaurus in the facet list and in the metadata editor. default: 0, asc order from 0 to N

Cardinality:

- `card_max=0` → Disabled, The Thesaurus will not appear in the GUI

Change thesaurus

Identifier	thesaurus_unique_identifier
Title	Thesaurus Title
Date	2018-05-23T10:25:56
Description	Thesaurus description
Slug	slug
About	http://about-thesaurus.com
Card min	1
Card max	0
☒ Facet	

Fig. 380: The GeoNode Thesaurus edit Interface

- $card_max=1$ & $card_min = 0$ → Single choice, optional.
- $card_max=1$ & $card_min = 1$ → Single choice, required
- $card_max=-1$ & $card_min = 0$ → [0..N] Multiple choices, optional
- $card_max=-1$ & $card_min = 1$ → [1..N] Multiple choices, required

After the setup, in *Editing Tools* → *Metadata* → *Wizard* the thesaurus block will be shown like the following image:

Thesaurus with multiple choices

× keyword1
× keyword2

Thesaurus with single choice

keyword

Fig. 381: The metadata interface with the Thesaurus enabled

Configuration via *settings.py*

Aviso: *Deprecated* The Thesaurus configuration via settings is deprecated, will be removed in the future.

After you loaded a thesaurus into GeoNode, it should be configured in the `settings.py` file (or in the `local_settings`) in this way:

```
THESAURUS = {'name': 'THESAURUS NAME', 'required': True|False, 'filter': True|False, }
```

- **name:** (mandatory string) the identifier you used in the `load_thesaurus` commands.
- **required:** (optional boolean) if `True`, a keyword of this thesaurus is mandatory to complete the metadata. *Currently not implemented.*
- **filter:** (optional boolean) if `True`, a faceted list of keywords of this thesaurus will be presented on the search page.

So, in order to set up the INSPIRE themes thesaurus you may set the `THESAURUS` value as:

```
THESAURUS = {'name': 'inspire_themes', 'required': True, 'filter': True}
```

1.29.5 Apply a thesaurus to a resource

After you've finished the setup you should find a new input widget in each resource metadata wizard allowing you to choose a thesaurus for your resource.

After applying a thesaurus to resources those should be listed in the filter section in GeoNodes resource list views.

▼ THESAURUS: INSPIRE_THE	
Atmospheric conditions	1
Coordinate reference syst...	1
Sea regions	1

1.30 Participate in the Discussion

1.30.1 Join the community, ask for help or report bugs

In case of general questions the GeoNode Community is present at following *channels*

- User Mailing List: <https://lists.osgeo.org/cgi-bin/mailman/listinfo/geonode-users>
- Developer Mailing List: <https://lists.osgeo.org/cgi-bin/mailman/listinfo/geonode-devel>
- Gitter Chat: <https://gitter.im/GeoNode/general>

For reporting bugs please open a ticket at Github issues:

- <https://github.com/GeoNode/geonode/issues>

1.31 Write Documentation

1.31.1 How to contribute to GeoNode's Documentation

If you feel like adding or changing something in the GeoNode documentation you are very welcome to do so. The documentation always needs improvement as the development of the software is going quite fast.

To contribute to the GeoNode documentation you should:

- Read the GeoServer Style Guidelines
- Create an account on GitHub
- Fork the GeoNode repository
- Edit the files
- Submit pull requests

All these things can generally be done within your browser, you won't need to download anything. However, if you need to add images or planning bigger changes working locally is recommended.

Style Guidelines

While we do not have strict rules for writing docs, we encourage you to read GeoServer Style Guidelines before you start writing: <https://docs.geoserver.org/latest/en/docguide/style.html>

Create an account on GitHub

The first step is to create an account on GitHub. Just go to [Github](#), find a username that suits you, enter your email and a password and hit *Sign up for GitHub*. After you've signed in, visit the `geonode_documentation` repository <https://github.com/geonode/documentation>.

Fork the documentation repository

In order to make changes, you first have to fork the repository. On the top right of the website, you will find a button named «fork» to do so.

If you want to read more about forking please visit the official GitHub docs: <https://help.github.com/articles/fork-a-repo>.

Edit files on Github

For smaller changes you can use the GitHub website. Navigate your Browser to your forked repository. To make changes to files, navigate to the file in question and hit the *edit* button on the right top.

Nota: The documentation is written in *reStructuredText*, a lightweight markup language. To learn how to use it see: <https://docutils.sourceforge.net/docs/user/rst/quickref.html>.

By hitting the *preview* button you will be able to see how your changes will look like. To save your changes, click on *Commit Changes* at the bottom of the site.

To ask the documentation maintainers to integrate your changes the creation of a *Pull Request* is needed. Therefore use the *new pull request* button to start the process. Find more about Pull requests at the official GitHub documentation: <https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-pull-requests> .

Edit files locally

If you're planning bigger changes on the structure of the documentation, it is advisable to make your changes locally. Further, while you can work on your master branch, it is recommended to create a dedicated branch for your changes.

Start by navigating to a folder where you like to keep your repository locally and install the needed dependencies

```
$ cd /opt
$ git clone https://github.com/your_documentation_repository
$ git remote add upstream https://github.com/geonode/documentation
# add the GeoNode documentation repository as "upstream" source

$ cd your_documentation_repository
$ git fetch upstream;
# get last commits from upstream

$ git merge upstream/master master
# merge the upstream with your fork
# if you like, you can also use 'git pull', which is nothing else than fetching and
↪merging in one step

$ git push
# update your repository at GitHub (origin)
```

Your repository should now be up to date! For more information on those commands go to <https://git-scm.com/docs>. Let's install the dependencies

```
$ pip install virtualenv
$ virtualenv docs_env
$ source docs_env/bin/activate
$ pip install sphinx sphinx_rtd_theme sphinx-autobuild
```

You can now start the sphinx development server which will serve and live-reload your docs at <https://localhost:8000>

```
$ sphinx-autobuild . _build
```

When finished create a build with following command

```
$ make html
# for a last check you can open the index.html in _build subdirectory
```

Create a pull request

As with directly editing files in your browser, you will need to create a Pull request to ask for integrating your changes into the main repository.

```
$ git status
# will list all changed files

$ git add ...
# add the files of interest
```

(continues on next page)

(continuação da página anterior)

```
$ git commit -m 'Fixes #1234 Updated docs for ...'
# choose a meaningful commit message

$ git push <branch>
```

After running these commands, navigate your browser to your GitHub repository and create a pull request as explained above.

1.32 Provide Translations

1.32.1 Contribute to Translations

Behind the scenes, GeoNode is using a software called GNU gettext further text-based translation files (django.po and djangojs.po) for translating content. If you'd like to know more about how all of this works you'll find a full description at the [Django Docs](#). Following will concentrate on what is needed for edit existing or contribute a new translation.

Download the translation File

All language files live in a specific subfolder called after their [iso code](#) within the [locale folder](#). For example, for French, the main translation file called django.po can be downloaded from [here](#).

Next, to download the language file, we need to install an OpenSource Editor called «poedit» for editing from: <https://poedit.net/download>

Translation process

Make a copy of the file before starting the translation so that you can revert in case of errors.

After installing “poedit”, you should be able to double click on the “.po” file to open it. Poedit’s interface should look similar to the one shown in the picture below:

Identifying translation issues

From the “poedit” menu “View”, make sure that “Entries with Errors first” is checked:

Next click on “Validate Translations” from the “Catalogue” menu:

“Poedit” will place translations which may require additional consideration on top of the list. A warning mark means that the interpretation might be not entirely consistent with the original phrase. This is not necessarily an error, just a warning asking the user to double check.

Following to marked phrases, “Poedit” will show untranslated sentences. When clicking on one, it can be translated through the bottom panel.

During translation pay special attention to the button saying “needs work”. In case this button is checked, the phrase will be marked as “fuzzy” and ignored in GeoNode.

The screenshot shows the GeoNode translation interface. At the top, there's a header bar with a title 'django.po — GeoNode' and a green 'Upgrade to Pro' button. Below the header, there's a table with two columns: 'Source text — English' and 'Translation — French'. The table contains the following rows:

Source text — English	Translation — French
Request to download a resource	Demander pour télécharger la ressource
A request for downloading a resource was sent	Une requête d'autorisation de téléchargement de la ressource a été...
series	séries
computer program or routine	logiciel ou routine
feature type	type d'entité
copy or imitation of an existing or hypothetical object	copie ou imitation d'un objet existant ou hypothétique
collection hardware	Dispositif de collecte
collection session	session de collecte
non-geographic data	données non géographiques
property type	type de propriété
field session	campagne d'acquisition
dataset	jeu de données
service interfaces	interfaces de service
attribute class	classe d'attribut
characteristic of a feature	caractéristique d'une entité
tile or spatial subset of geographic data	tuile ou sous-ensemble spatial des données géographiques
feature	entité

Below the table, there's a section for 'Source text:' with the text 'Request to download a resource'. To the right of this section is a toggle switch labeled 'Needs Work' which is currently turned off. Below the source text, there's a section for 'Translation:' with the text 'Demander pour télécharger la ressource'. On the far right, there's a sidebar with 'Translation suggestions:' showing a suggestion from Microsoft with a 90% match. At the bottom of the sidebar, there's a button 'Add Comment'. At the bottom of the main interface, there's a status bar that says 'Translated: 946 of 946 (100 %)'.

Saving translations

As soon as the translation is complete, it must be saved and compiled. Saving is straightforward. All you have to do is clicking the “Save” button from the top menu.

As a last step we compile the file. Compiling the translation means to create a binary «.mo» file out of the edited «.po» file. To do so, click on «Compile to MO»

Poedit will ask where to write the «.mo» file to, by default, this is the same folder as the edited “.po” resides in. The “.mo” file can be overwritten if necessary.

Push translations to the repository

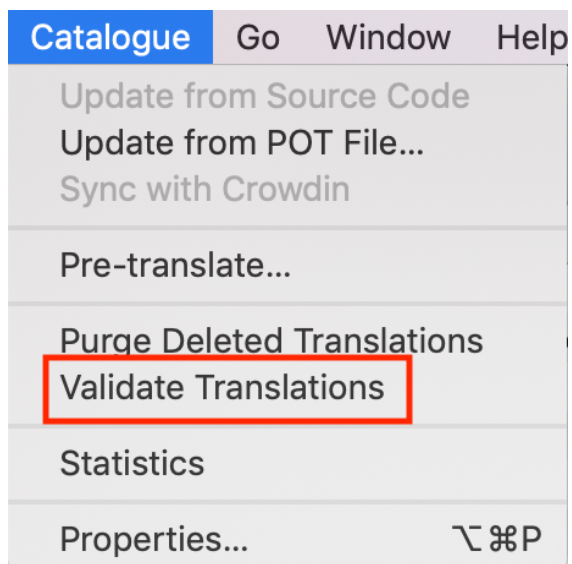
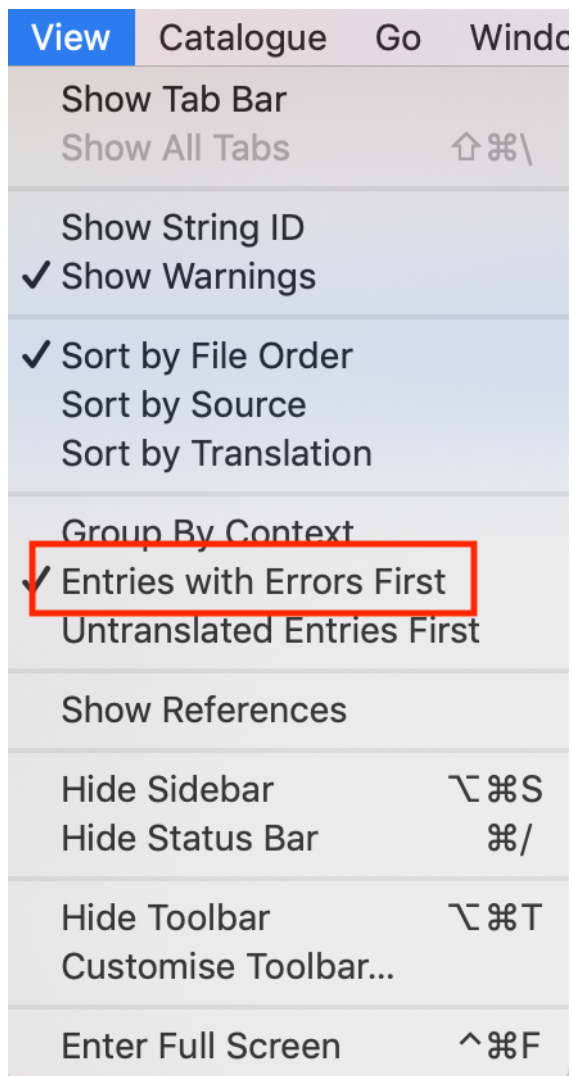
For sharing our updates, we must upload the files to GeoNode’s GitHub repository. Go to the correct file position which, in case for French is: https://github.com/GeoNode/geonode/tree/master/geonode/locale/fr/LC_MESSAGES

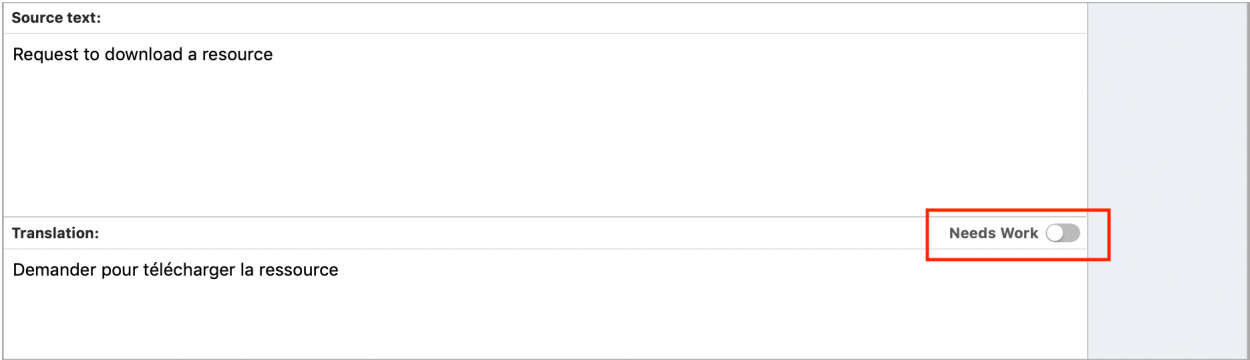
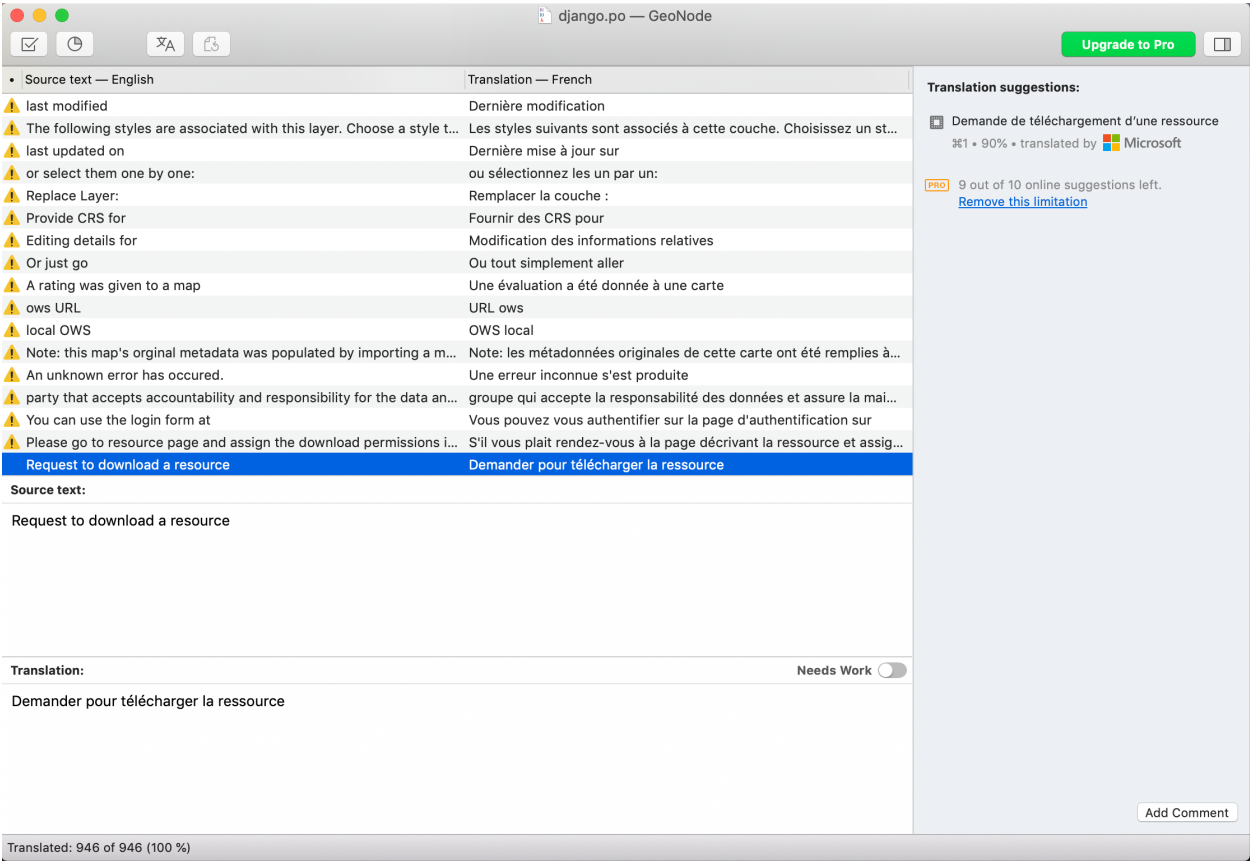
Click on «Upload Files»

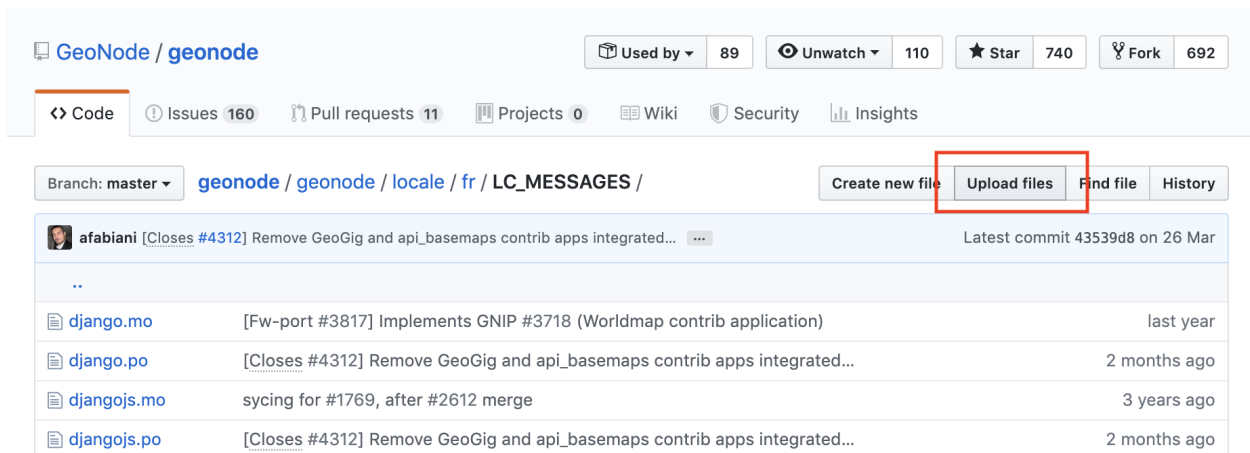
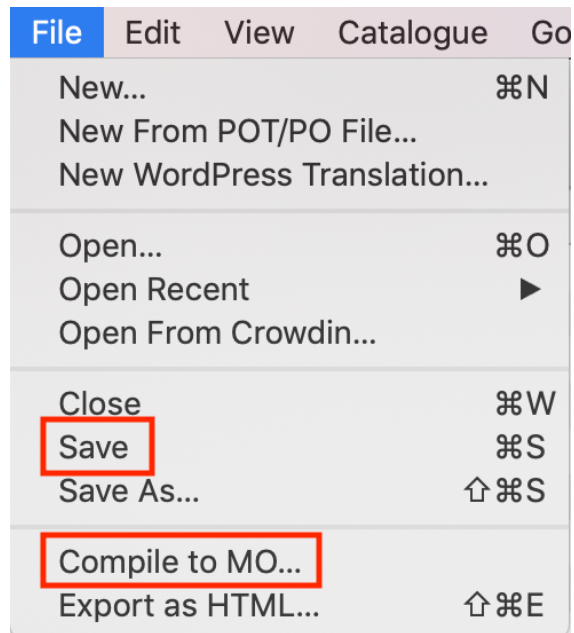
Drag the updated files into the Upload form, and write a title/description of the changes

Click on «Create a new branch for this commit. . . » and then click on the green button.

The last step will create a *PULL REQUEST* which can be reviewed and then approved by a developer.







GeoNode / geonode

Used by 89

Unwatch 110

Star 740

Fork 692

<> Code

Issues 160

Pull requests 11

Projects 0

Wiki

Security

Insights

geonode / geonode / locale / fr / LC_MESSAGES



Drag files here to add them to your repository

Or [choose your files](#)

Commit changes

Add files via upload

Add an optional extended description...

☐ You can't commit to `master` because it is a [protected branch](#).

☒ Create a **new branch** for this commit and start a pull request. [Learn more about pull requests](#).

Commit changes

Cancel

Activate updated translation at your server

Once the files have been pushed to GitHub, it will be necessary to update your server to respect changed files.

At this time, this can be done only by an administrator. From the server “shell” following commands need to be executed:

```
workon geonode
cd /opt/geonode
DJANGO_SETTINGS_MODULE=geonode.settings python -W ignore manage.py collectstatic --
↳ noinput
sudo service uwsgi restart
```

Texts not listed in .po files

In case you find a template output without corresponding translation you can add it as follows:

Identify the corresponding template file which is responsible for outputting the text. Add a `{% trans "TEXT" %}` tag. Save the template file and run the following:

```
django-admin makemessages --no-location -l en -d django -e "html,txt,py" -i docs
django-admin makemessages --no-location -l en -d djangojs -e "js" -i docs -i node_
↳ modules -i lib
```

This will update the english .po file. also to update the language which should be edited by settings the `-l fr` parameter. Continue with updating the .po file as described above.

1.33 Write Code

1.34 Frontend Development

1.34.1 Frontend development

Knowledge of handling node/npm is required.

The GeoNode frontend dependencies can be found in `./geonode/static`. To manage dependencies, we recommend the use of yarn package manager (<https://yarnpkg.com/lang/en>).

First steps:

```
yarn install
```

Installs the required libraries to `./node_modules`

```
yarn install <package>@version [--dev]
```

Installs a package with a defined version. Using `-dev` installs a dependency that is only available for the build process (see: `package.json devDependencies`).

```
yarn remove <package>
```

Removes a package.

```
yarn outdated
```

Shows version information.

```
yarn why <package>
```

Get information on why this package was installed.

For further information on how to install or use please visit the official yarn documentation.

File/Folder overview:

```
./static_dependencies.json
```

includes all dependencies associated with each file. For example all files which should be minified to `assets.min.js` are named as values. All files that should be copied to `lib` folder (for `DEBUG_STATIC`) are values of key `other_dependencies` and so on. Before you can use a dependency it has to be added to `package.json` by use of yarn.

```
./Gruntfile.js
```

reads the dependencies from `static_dependencies.json` and contains all workflows.

`geonode/static/geonode`

The `./geonode` folder contains GeoNode's stylesheets and javascript files. The CSS files are generated via less. CSS files should therefore never be changed directly but it's corresponding less file. Further this folder should never be deleted!

`geonode/static/lib`

The `./lib` folder contains all the third-party files. This folder can be deleted as it will be fully generated by use of `grunt development|production`

Example 1 – Change styling:

1. In your settings set `DEBU_STATIC=True`. This will load unminified assets in your template.
2. Start the development server with `paver start`.
3. Use `grunt watch` to watch all less files for change.
4. Change styling in `./geonode/static/geonode/less`
5. If our changes are as expected create a new build with `grunt development` (files are not minimized) or `grunt production` (files are minimized)

Example 2 – add/update a new library:

1. In your settings set `DEBU_STATIC=True`. This will load unminified assets in your template.
2. `yarn add angular@1.7`
3. `vim static_dependencies.json` Edit the file and add your dependency to its fitting destination. For example, `assets.min.js`
4. Check if some Django template (for example, `base.html`) includes the file and add it or adjust the version
5. use `grunt production` to build the package

For further tasks have a look at [gruntfile.js](#) or ask for help in the [development mailing list](#)

Nota: Please make maintainers work easier and add a message to your commit why a library has been added. (For example, `commit -m "select2 added for permissions form on layer detail page"`)

1.35 API Schema

1.35.1 GeoNode API Schema

API v2 - REST

OpenAPI 3.0 Schema

GET `/api/v2/`

Returns a list of available endpoints

Status Codes

- **200 OK** – A json with a list of available endpoints

GET `/api/v2/documents/`

API endpoint that return all the documents available with detailed information (paginated).

Query Parameters

- **ordering** (*string*) – Which field to use when ordering the results.
- **page** (*integer*) – A page number within the paginated result set.
- **page_size** (*integer*) – Number of results to return per page.
- **search** (*string*) – A search term.

Status Codes

- **200 OK** –

POST `/api/v2/documents/`

Either create a single or many model instances in bulk using the Serializer's `many=True` ability from Django REST >= 2.2.5.

The data can be represented by the serializer name (single or plural forms), dict or list.

Examples:

POST `/dogs/` {

 «name»: «Fido», «age»: 2

}

POST `/dogs/` {

 «dog»: { «name»: «Lucky», «age»: 3

 }

}

POST `/dogs/` {


```

    «dogs»: [ { «name»: «Fido», «age»: 2 }, { «name»: «Lucky», «age»: 3 }
  ]
}
POST /dogs/ [
  { «name»: «Fido», «age»: 2 }, { «name»: «Lucky», «age»: 3 }
]

```

Status Codes

- 201 Created –

DELETE /api/v2/documents/

Either delete a single or many model instances in bulk, by sending a json with the ID of the document to delete

```

DELETE /dogs/ {
  «dogs»: [ { «id»: 1 }, { «id»: 2 }
]
}
DELETE /dogs/ [
  { «id»: 1 }, { «id»: 2 }
]

```

Status Codes

- 204 No Content – No response body

GET /api/v2/documents/{id}/

API endpoint that return detailed information of a specific document.

Parameters

- **id** (*integer*) – A unique integer value identifying this document.

Status Codes

- 200 OK –

PUT /api/v2/documents/{id}/

Update one or more model instances.

If ENABLE_BULK_UPDATE is set, multiple previously-fetched records may be updated in a single call, provided their IDs.

If ENABLE_PATCH_ALL is set, multiple records may be updated in a single PATCH call, even without knowing their IDs.

WARNING: ENABLE_PATCH_ALL should be considered an advanced feature and used with caution. This feature must be enabled at the viewset level and must also be requested explicitly by the client via the «patch-all» query parameter.

This parameter can have one of the following values:

true (or 1): records will be fetched and then updated in a transaction loop

- The *Model.save* method will be called and model signals will run
- This can be slow if there are too many signals or many records in the query

- This is considered the more safe and default behavior

query: records will be updated in a single query

- The *QuerySet.update* method will be called and model signals will not run
- This will be fast, but may break data constraints that are controlled by signals
- This is considered unsafe but useful in certain situations

The server's successful response to a patch-all request will NOT include any individual records. Instead, the response content will contain a «meta» object with an «updated» count of updated records.

Examples:

Update one dog:

```
PATCH /dogs/1/ {  
    "fur": "white"  
}
```

Update many dogs by ID:

```
PATCH /dogs/ [  
    {"id": 1, "fur": "white"}, {"id": 2, "fur": "black"}, {"id": 3, "fur": "yellow"}  
]
```

Update all dogs in a query:

```
PATCH /dogs/?filter{fur.contains}=brown&patch-all=true {  
    "fur": "gold"  
}
```

Parameters

- **id** (*integer*) – A unique integer value identifying this document.

Status Codes

- 200 OK –

PATCH /api/v2/documents/{id}/

API endpoint that allows documents to be viewed or edited.

Parameters

- **id** (*integer*) – A unique integer value identifying this document.

Status Codes

- 200 OK –

DELETE /api/v2/documents/{id}/

Delete a single document.

Parameters

- **id** (*integer*) – A unique integer value identifying this document.

Status Codes

- 204 No Content – No response body

GET /api/v2/documents/{id}/linked_resources/
API endpoint that return all the linked resources of a specific id.

Parameters

- **id** (*integer*) – A unique integer value identifying this document.

Status Codes

- 200 OK –

GET /api/v2/geoapps/
API endpoint that show all the geoapps available.

Query Parameters

- **ordering** (*string*) – Which field to use when ordering the results.
- **page** (*integer*) – A page number within the paginated result set.
- **page_size** (*integer*) – Number of results to return per page.
- **search** (*string*) – A search term.

Status Codes

- 200 OK –

POST /api/v2/geoapps/
Either create a single or many model instances in bulk using the Serializer's many=True ability from Django REST >= 2.2.5.

The data can be represented by the serializer name (single or plural forms), dict or list.

Examples:

POST /dogs/ {

 «name»: «Fido», «age»: 2

}

POST /dogs/ {

 «dog»: { «name»: «Lucky», «age»: 3

 }

}

POST /dogs/ {

 «dogs»: [{ «name»: «Fido», «age»: 2 }, { «name»: «Lucky», «age»: 3 }

]

}

POST /dogs/ [

 { «name»: «Fido», «age»: 2 }, { «name»: «Lucky», «age»: 3 }

]

Status Codes

- 201 Created –

PATCH /api/v2/geoapps/
API endpoint that allows geoapps to be viewed or edited.

Status Codes

- 200 OK –

DELETE /api/v2/geoapps/

Either delete a single or many model instances in bulk

```
DELETE /dogs/ {  
  «dogs»: [ {«id»: 1}, {«id»: 2}  
  ]  
}  
  
DELETE /dogs/ [  
  {«id»: 1}, {«id»: 2}  
]
```

Status Codes

- 204 No Content – No response body

GET /api/v2/geoapps/{id}/

API endpoint that show details of a specific geoapp.

Parameters

- **id** (*integer*) – A unique integer value identifying this geo app.

Status Codes

- 200 OK –

PUT /api/v2/geoapps/{id}/

Update one or more model instances.

If `ENABLE_BULK_UPDATE` is set, multiple previously-fetched records may be updated in a single call, provided their IDs.

If `ENABLE_PATCH_ALL` is set, multiple records may be updated in a single `PATCH` call, even without knowing their IDs.

WARNING: `ENABLE_PATCH_ALL` should be considered an advanced feature and used with caution. This feature must be enabled at the viewset level and must also be requested explicitly by the client via the «patch-all» query parameter.

This parameter can have one of the following values:

true (or 1): records will be fetched and then updated in a transaction loop

- The *Model.save* method will be called and model signals will run
- This can be slow if there are too many signals or many records in the query
- This is considered the more safe and default behavior

query: records will be updated in a single query

- The *QuerySet.update* method will be called and model signals will not run
- This will be fast, but may break data constraints that are controlled by signals
- This is considered unsafe but useful in certain situations

The server's successful response to a patch-all request will NOT include any individual records. Instead, the response content will contain a «meta» object with an «updated» count of updated records.

Examples:

Update one dog:

```
PATCH /dogs/1/ {
  "fur": "white"
}
```

Update many dogs by ID:

```
PATCH /dogs/ [
  {"id": 1, "fur": "white"}, {"id": 2, "fur": "black"}, {"id": 3, "fur": "yellow"}
]
```

Update all dogs in a query:

```
PATCH /dogs/?filter{fur.contains}=brown&patch-all=true {
  "fur": "gold"
}
```

Parameters

- **id** (*integer*) – A unique integer value identifying this geo app.

Status Codes

- 200 OK –

PATCH /api/v2/geoapps/{id}/

API endpoint that allows geoapps to be viewed or edited.

Parameters

- **id** (*integer*) – A unique integer value identifying this geo app.

Status Codes

- 200 OK –

DELETE /api/v2/geoapps/{id}/

Delete a single instances

```
DELETE /dogs/ {
  «dogs»: [ {«id»: 1}, {«id»: 2}
]
}

DELETE /dogs/ [
  {«id»: 1}, {«id»: 2}
]
```

Parameters

- **id** (*integer*) – A unique integer value identifying this geo app.

Status Codes

- **204 No Content** – No response body

GET `/api/v2/geoapps/{id}/{field_name}/`

Fetch related object(s), as if sideloaded (used to support link objects).

This method gets mapped to `/<resource>/<pk>/<field_name>/` by `DynamicRouter` for all `DynamicRelationField` fields. Generally, this method probably shouldn't be overridden.

An alternative implementation would be to generate reverse queries. For an exploration of that approach, see:

<https://gist.github.com/ryochiji/54687d675978c7d96503>

Parameters

- **field_name** (*string*) –
- **id** (*integer*) – A unique integer value identifying this geo app.

Status Codes

- **200 OK** –

GET `/api/v2/geostories/`

API endpoint that return all the geostories available with detailed information (paginated).

Query Parameters

- **ordering** (*string*) – Which field to use when ordering the results.
- **page** (*integer*) – A page number within the paginated result set.
- **page_size** (*integer*) – Number of results to return per page.
- **search** (*string*) – A search term.

Status Codes

- **200 OK** –

POST `/api/v2/geostories/`

Either create a single or many model instances in bulk using the Serializer's `many=True` ability from Django REST >= 2.2.5.

The data can be represented by the serializer name (single or plural forms), dict or list.

Examples:

POST `/dogs/` {

 «name»: «Fido», «age»: 2

}

POST `/dogs/` {

 «dog»: { «name»: «Lucky», «age»: 3

 }

}

POST `/dogs/` {

 «dogs»: [{ «name»: «Fido», «age»: 2 }, { «name»: «Lucky», «age»: 3 }

]

```

}
POST /dogs/ [
  {«name»: «Fido», «age»: 2}, {«name»: «Lucky», «age»: 3}
]

```

Status Codes

- 201 Created –

PATCH /api/v2/geostories/

API endpoint that allows geoapps to be viewed or edited.

Status Codes

- 200 OK –

DELETE /api/v2/geostories/

Either delete a single or many model instances in bulk

```

DELETE /dogs/ {
  «dogs»: [ {«id»: 1}, {«id»: 2}
]
}
DELETE /dogs/ [
  {«id»: 1}, {«id»: 2}
]

```

Status Codes

- 204 No Content – No response body

GET /api/v2/geostories/{id}/

API endpoint that return detailed information of a specific geostory.

Parameters

- **id** (*integer*) – A unique integer value identifying this geo story.

Status Codes

- 200 OK –

PUT /api/v2/geostories/{id}/

Update one or more model instances.

If `ENABLE_BULK_UPDATE` is set, multiple previously-fetched records may be updated in a single call, provided their IDs.

If `ENABLE_PATCH_ALL` is set, multiple records may be updated in a single PATCH call, even without knowing their IDs.

WARNING: `ENABLE_PATCH_ALL` should be considered an advanced feature and used with caution. This feature must be enabled at the viewset level and must also be requested explicitly by the client via the «patch-all» query parameter.

This parameter can have one of the following values:

true (or 1): records will be fetched and then updated in a transaction loop

- The *Model.save* method will be called and model signals will run

- This can be slow if there are too many signals or many records in the query
- This is considered the more safe and default behavior

query: records will be updated in a single query

- The *QuerySet.update* method will be called and model signals will not run
- This will be fast, but may break data constraints that are controlled by signals
- This is considered unsafe but useful in certain situations

The server's successful response to a patch-all request will NOT include any individual records. Instead, the response content will contain a «meta» object with an «updated» count of updated records.

Examples:

Update one dog:

```
PATCH /dogs/1/ {  
  "fur": "white"  
}
```

Update many dogs by ID:

```
PATCH /dogs/ [  
  {"id": 1, "fur": "white"}, {"id": 2, "fur": "black"}, {"id": 3, "fur": "yellow"}  
]
```

Update all dogs in a query:

```
PATCH /dogs/?filter{fur.contains}=brown&patch-all=true {  
  "fur": "gold"  
}
```

Parameters

- **id** (*integer*) – A unique integer value identifying this geo story.

Status Codes

- 200 OK –

PATCH /api/v2/geostories/{id}/

API endpoint that allows geoapps to be viewed or edited.

Parameters

- **id** (*integer*) – A unique integer value identifying this geo story.

Status Codes

- 200 OK –

DELETE /api/v2/geostories/{id}/

Either delete a single or many model instances in bulk

```
DELETE /dogs/ {  
  «dogs»: [ {«id»: 1}, {«id»: 2}  
]
```



```

}
DELETE /dogs/ [
  {«id»: 1}, {«id»: 2}
]

```

Parameters

- **id** (*integer*) – A unique integer value identifying this geo story.

Status Codes

- **204 No Content** – No response body

GET `/api/v2/geostories/{id}/{field_name}/`

Fetch related object(s), as if sideloaded (used to support link objects).

This method gets mapped to `/<resource>/<pk>/<field_name>/` by `DynamicRouter` for all `DynamicRelationField` fields. Generally, this method probably shouldn't be overridden.

An alternative implementation would be to generate reverse queries. For an exploration of that approach, see:

<https://gist.github.com/ryochiji/54687d675978c7d96503>

Parameters

- **field_name** (*string*) –
- **id** (*integer*) – A unique integer value identifying this geo story.

Status Codes

- **200 OK** –

GET `/api/v2/groups/`

API endpoint that return all groups available with detailed information (paginated).

Query Parameters

- **ordering** (*string*) – Which field to use when ordering the results.
- **page** (*integer*) – A page number within the paginated result set.
- **page_size** (*integer*) – Number of results to return per page.

Status Codes

- **200 OK** –

POST `/api/v2/groups/`

Either create a single or many model instances in bulk using the Serializer's `many=True` ability from Django REST >= 2.2.5.

The data can be represented by the serializer name (single or plural forms), dict or list.

Examples:

```

POST /dogs/ {
  «name»: «Fido», «age»: 2
}

POST /dogs/ {

```

```
    «dog»: { «name»: «Lucky», «age»: 3
  }
}
POST /dogs/ {
  «dogs»: [ { «name»: «Fido», «age»: 2}, { «name»: «Lucky», «age»: 3}
]
}
POST /dogs/ [
  { «name»: «Fido», «age»: 2}, { «name»: «Lucky», «age»: 3}
]
```

Status Codes

- 201 Created –

PATCH /api/v2/groups/

API endpoint that allows groups to be viewed or edited.

Status Codes

- 200 OK –

DELETE /api/v2/groups/

Either delete a single or many model instances in bulk

```
DELETE /dogs/ {
  «dogs»: [ { «id»: 1}, { «id»: 2}
]
}
DELETE /dogs/ [
  { «id»: 1}, { «id»: 2}
]
```

Status Codes

- 204 No Content – No response body

GET /api/v2/groups/{id}/

API endpoint that return detailed information of a specific group.

Parameters

- **id** (*integer*) – A unique integer value identifying this group profile.

Status Codes

- 200 OK –

PUT /api/v2/groups/{id}/

Update one or more model instances.

If `ENABLE_BULK_UPDATE` is set, multiple previously-fetched records may be updated in a single call, provided their IDs.

If `ENABLE_PATCH_ALL` is set, multiple records may be updated in a single `PATCH` call, even without knowing their IDs.

WARNING: `ENABLE_PATCH_ALL` should be considered an advanced feature and used with caution. This feature must be enabled at the viewset level and must also be requested explicitly by the client via the «patch-all» query parameter.

This parameter can have one of the following values:

true (or 1): records will be fetched and then updated in a transaction loop

- The *Model.save* method will be called and model signals will run
- This can be slow if there are too many signals or many records in the query
- This is considered the more safe and default behavior

query: records will be updated in a single query

- The *QuerySet.update* method will be called and model signals will not run
- This will be fast, but may break data constraints that are controlled by signals
- This is considered unsafe but useful in certain situations

The server's successful response to a patch-all request will NOT include any individual records. Instead, the response content will contain a «meta» object with an «updated» count of updated records.

Examples:

Update one dog:

```
PATCH /dogs/1/ {
  "fur": "white"
}
```

Update many dogs by ID:

```
PATCH /dogs/ [
  {"id": 1, "fur": "white"}, {"id": 2, "fur": "black"}, {"id": 3, "fur": "yellow"}
]
```

Update all dogs in a query:

```
PATCH /dogs/?filter{fur.contains}=brown&patch-all=true {
  "fur": "gold"
}
```

Parameters

- **id** (*integer*) – A unique integer value identifying this group profile.

Status Codes

- 200 OK –

PATCH /api/v2/groups/{id}/

API endpoint that allows groups to be viewed or edited.

Parameters

- **id** (*integer*) – A unique integer value identifying this group profile.

Status Codes

- 200 OK –

DELETE /api/v2/groups/{id}/

Delete a single instance

Parameters

- **id** (*integer*) – A unique integer value identifying this group profile.

Status Codes

- 204 No Content – No response body

GET /api/v2/groups/{id}/managers/

API endpoint that show the managers for the selected group.

Parameters

- **id** (*integer*) – A unique integer value identifying this group profile.

Status Codes

- 200 OK –

GET /api/v2/groups/{id}/members/

API endpoint that show either members and managers for the selected group.

Parameters

- **id** (*integer*) – A unique integer value identifying this group profile.

Status Codes

- 200 OK –

GET /api/v2/groups/{id}/resources/

API endpoint that show all the resources connected to the selected group.

Parameters

- **id** (*integer*) – A unique integer value identifying this group profile.

Status Codes

- 200 OK –

GET /api/v2/layers/

API endpoint that return all the layers available with detailed information (paginated).

Query Parameters

- **ordering** (*string*) – Which field to use when ordering the results.
- **page** (*integer*) – A page number within the paginated result set.
- **page_size** (*integer*) – Number of results to return per page.
- **search** (*string*) – A search term.

Status Codes

- 200 OK –

POST /api/v2/layers/

Either create a single or many model instances in bulk using the Serializer's many=True ability from Django REST >= 2.2.5.

The data can be represented by the serializer name (single or plural forms), dict or list.

Examples:

```
POST /dogs/ {
    «name»: «Fido», «age»: 2
}
POST /dogs/ {
    «dog»: { «name»: «Lucky», «age»: 3
}
}
POST /dogs/ {
    «dogs»: [ { «name»: «Fido», «age»: 2}, { «name»: «Lucky», «age»: 3}
]
}
POST /dogs/ [
    { «name»: «Fido», «age»: 2}, { «name»: «Lucky», «age»: 3}
]
```

Status Codes

- [201 Created](#) –

PATCH /api/v2/layers/

API endpoint that allows layers to be viewed or edited.

Status Codes

- [200 OK](#) –

DELETE /api/v2/layers/

Either delete a single or many model instances in bulk

```
DELETE /dogs/ {
    «dogs»: [ { «id»: 1}, { «id»: 2}
]
}
DELETE /dogs/ [
    { «id»: 1}, { «id»: 2}
]
```

Status Codes

- [204 No Content](#) – No response body

GET /api/v2/layers/{id}/

API endpoint that return detailed information of a specific layer

Parameters

- **id** (*integer*) – A unique integer value identifying this layer.

Status Codes

- 200 OK –

PUT /api/v2/layers/{id}/

Update one or more model instances.

If `ENABLE_BULK_UPDATE` is set, multiple previously-fetched records may be updated in a single call, provided their IDs.

If `ENABLE_PATCH_ALL` is set, multiple records may be updated in a single PATCH call, even without knowing their IDs.

WARNING: `ENABLE_PATCH_ALL` should be considered an advanced feature and used with caution. This feature must be enabled at the viewset level and must also be requested explicitly by the client via the «patch-all» query parameter.

This parameter can have one of the following values:

true (or 1): records will be fetched and then updated in a transaction loop

- The *Model.save* method will be called and model signals will run
- This can be slow if there are too many signals or many records in the query
- This is considered the more safe and default behavior

query: records will be updated in a single query

- The *QuerySet.update* method will be called and model signals will not run
- This will be fast, but may break data constraints that are controlled by signals
- This is considered unsafe but useful in certain situations

The server's successful response to a patch-all request will NOT include any individual records. Instead, the response content will contain a «meta» object with an «updated» count of updated records.

Examples:

Update one dog:

```
PATCH /dogs/1/ {  
  "fur": "white"  
}
```

Update many dogs by ID:

```
PATCH /dogs/ [  
  {"id": 1, "fur": "white"}, {"id": 2, "fur": "black"}, {"id": 3, "fur": "yellow"}  
]
```

Update all dogs in a query:

```
PATCH /dogs/?filter{fur.contains}=brown&patch-all=true {  
  "fur": "gold"  
}
```

Parameters

- **id** (*integer*) – A unique integer value identifying this layer.

Status Codes

- 200 OK –

PATCH /api/v2/layers/{id}/

API endpoint that allows layers to be viewed or edited.

Parameters

- **id** (*integer*) – A unique integer value identifying this layer.

Status Codes

- 200 OK –

DELETE /api/v2/layers/{id}/

Delete a single model instance

```
DELETE /dogs/ {
  «dogs»: [ {«id»: 1}, {«id»: 2}
]
}
DELETE /dogs/ [
  {«id»: 1}, {«id»: 2}
]
```

Parameters

- **id** (*integer*) – A unique integer value identifying this layer.

Status Codes

- 204 No Content – No response body

GET /api/v2/layers/{id}/{field_name}/

Fetch related object(s), as if sideloaded (used to support link objects).

This method gets mapped to `/<resource>/<pk>/<field_name>/` by `DynamicRouter` for all `DynamicRelationField` fields. Generally, this method probably shouldn't be overridden.

An alternative implementation would be to generate reverse queries. For an exploration of that approach, see:

<https://gist.github.com/ryochiji/54687d675978c7d96503>

Parameters

- **field_name** (*string*) –
- **id** (*integer*) – A unique integer value identifying this layer.

Status Codes

- 200 OK –

GET /api/v2/maps/

API endpoint that return all the maps available with detailed information (paginated).

Query Parameters

- **ordering** (*string*) – Which field to use when ordering the results.
- **page** (*integer*) – A page number within the paginated result set.
- **page_size** (*integer*) – Number of results to return per page.

- **search** (*string*) – A search term.

Status Codes

- 200 OK –

POST /api/v2/maps/

Either create a single or many model instances in bulk using the Serializer's many=True ability from Django REST >= 2.2.5.

The data can be represented by the serializer name (single or plural forms), dict or list.

Examples:

POST /dogs/ {

 «name»: «Fido», «age»: 2

}

POST /dogs/ {

 «dog»: { «name»: «Lucky», «age»: 3

 }

}

POST /dogs/ {

 «dogs»: [{ «name»: «Fido», «age»: 2 }, { «name»: «Lucky», «age»: 3 }

]

}

POST /dogs/ [

 { «name»: «Fido», «age»: 2 }, { «name»: «Lucky», «age»: 3 }

]

Status Codes

- 201 Created –

PATCH /api/v2/maps/

API endpoint that allows maps to be viewed or edited.

Status Codes

- 200 OK –

DELETE /api/v2/maps/

Either delete a single or many model instances in bulk

DELETE /dogs/ {

 «dogs»: [{ «id»: 1 }, { «id»: 2 }

]

}

DELETE /dogs/ [

 { «id»: 1 }, { «id»: 2 }

]

Status Codes

- **204 No Content** – No response body

GET `/api/v2/maps/{id}/`

API endpoint that return detailed information of a specific map.

Parameters

- **id** (*integer*) – A unique integer value identifying this map.

Status Codes

- **200 OK** –

PUT `/api/v2/maps/{id}/`

Update one or more model instances.

If `ENABLE_BULK_UPDATE` is set, multiple previously-fetched records may be updated in a single call, provided their IDs.

If `ENABLE_PATCH_ALL` is set, multiple records may be updated in a single `PATCH` call, even without knowing their IDs.

WARNING: `ENABLE_PATCH_ALL` should be considered an advanced feature and used with caution. This feature must be enabled at the viewset level and must also be requested explicitly by the client via the «patch-all» query parameter.

This parameter can have one of the following values:

true (or 1): records will be fetched and then updated in a transaction loop

- The *Model.save* method will be called and model signals will run
- This can be slow if there are too many signals or many records in the query
- This is considered the more safe and default behavior

query: records will be updated in a single query

- The *QuerySet.update* method will be called and model signals will not run
- This will be fast, but may break data constraints that are controlled by signals
- This is considered unsafe but useful in certain situations

The server's successful response to a patch-all request will NOT include any individual records. Instead, the response content will contain a «meta» object with an «updated» count of updated records.

Examples:

Update one dog:

```
PATCH /dogs/1/ {
  "fur": "white"
}
```

Update many dogs by ID:

```
PATCH /dogs/ [
  {"id": 1, "fur": "white"}, {"id": 2, "fur": "black"}, {"id": 3, "fur": "yellow"}
]
```

Update all dogs in a query:

```
PATCH /dogs/?filter{fur.contains}=brown&patch-all=true {  
  "fur": "gold"  
}
```

Parameters

- **id** (*integer*) – A unique integer value identifying this map.

Status Codes

- 200 OK –

PATCH /api/v2/maps/{id}/

API endpoint that allows maps to be viewed or edited.

Parameters

- **id** (*integer*) – A unique integer value identifying this map.

Status Codes

- 200 OK –

DELETE /api/v2/maps/{id}/

Delete a single instance.

Parameters

- **id** (*integer*) – A unique integer value identifying this map.

Status Codes

- 204 No Content – No response body

GET /api/v2/maps/{id}/layers/

Return the list of the layer that compose the map with some information like opacity, name, visibility. In the layer_params field, are available the information regarding the layer

Parameters

- **id** (*integer*) – A unique integer value identifying this map.

Status Codes

- 200 OK –

GET /api/v2/maps/{id}/local_layers/

API endpoint allowing to retrieve a detailed list of the local MapLayers

Parameters

- **id** (*integer*) – A unique integer value identifying this map.

Status Codes

- 200 OK –

GET /api/v2/resources/

API endpoint that show all resources available in GeoNode, includes maps, layers, geostories and documents (paginated).

Query Parameters

- **ordering** (*string*) – Which field to use when ordering the results.
- **page** (*integer*) – A page number within the paginated result set.

- **page_size** (*integer*) – Number of results to return per page.
- **search** (*string*) – A search term.

Status Codes

- 200 OK –

POST /api/v2/resources/

Either create a single or many model instances in bulk using the Serializer's many=True ability from Django REST >= 2.2.5.

The data can be represented by the serializer name (single or plural forms), dict or list.

Examples:

POST /dogs/ {

 «name»: «Fido», «age»: 2

}

POST /dogs/ {

 «dog»: { «name»: «Lucky», «age»: 3

 }

}

POST /dogs/ {

 «dogs»: [{ «name»: «Fido», «age»: 2 }, { «name»: «Lucky», «age»: 3 }

]

}

POST /dogs/ [

 { «name»: «Fido», «age»: 2 }, { «name»: «Lucky», «age»: 3 }

]

Status Codes

- 201 Created –

PATCH /api/v2/resources/

API endpoint that allows base resources to be viewed or edited.

Status Codes

- 200 OK –

DELETE /api/v2/resources/

Either delete a single or many model instances in bulk

DELETE /dogs/ {

 «dogs»: [{ «id»: 1 }, { «id»: 2 }

]

}

DELETE /dogs/ [

 { «id»: 1 }, { «id»: 2 }

]

Status Codes

- **204 No Content** – No response body

GET `/api/v2/resources/{id}/`

API endpoint that return detailed information of a specific resource, including keywords, owner and bbox_polygon.

Parameters

- **id** (*integer*) – A unique integer value identifying this resource base.

Status Codes

- **200 OK** –

PUT `/api/v2/resources/{id}/`

Update one or more model instances.

If `ENABLE_BULK_UPDATE` is set, multiple previously-fetched records may be updated in a single call, provided their IDs.

If `ENABLE_PATCH_ALL` is set, multiple records may be updated in a single `PATCH` call, even without knowing their IDs.

WARNING: `ENABLE_PATCH_ALL` should be considered an advanced feature and used with caution. This feature must be enabled at the viewset level and must also be requested explicitly by the client via the «patch-all» query parameter.

This parameter can have one of the following values:

true (or 1): records will be fetched and then updated in a transaction loop

- The *Model.save* method will be called and model signals will run
- This can be slow if there are too many signals or many records in the query
- This is considered the more safe and default behavior

query: records will be updated in a single query

- The *QuerySet.update* method will be called and model signals will not run
- This will be fast, but may break data constraints that are controlled by signals
- This is considered unsafe but useful in certain situations

The server's successful response to a patch-all request will NOT include any individual records. Instead, the response content will contain a «meta» object with an «updated» count of updated records.

Examples:

Update one dog:

```
PATCH /dogs/1/ {  
  "fur": "white"  
}
```

Update many dogs by ID:

```
PATCH /dogs/ [  
  {"id": 1, "fur": "white"}, {"id": 2, "fur": "black"}, {"id": 3, "fur": "yellow"}  
]
```

Update all dogs in a query:

```
PATCH /dogs/?filter{fur.contains}=brown&patch-all=true {
  "fur": "gold"
}
```

Parameters

- **id** (*integer*) – A unique integer value identifying this resource base.

Status Codes

- 200 OK –

PATCH /api/v2/resources/{id}/

API endpoint that allows base resources to be viewed or edited.

Parameters

- **id** (*integer*) – A unique integer value identifying this resource base.

Status Codes

- 200 OK –

DELETE /api/v2/resources/{id}/

Delete a single instance

Parameters

- **id** (*integer*) – A unique integer value identifying this resource base.

Status Codes

- 204 No Content – No response body

GET /api/v2/resources/{id}/get_perms/

API endpoint that show all the permissions defined for a specific resource, splitted by groups and users.

Parameters

- **id** (*integer*) – A unique integer value identifying this resource base.

Status Codes

- 200 OK –

PUT /api/v2/resources/{id}/set_perms/

API endpoint that allows the user to set the permissions to a specific resource.

Parameters

- **id** (*integer*) – A unique integer value identifying this resource base.

Status Codes

- 200 OK –

GET /api/v2/resources/approved/

Will return a list of approved resource available.

Status Codes

- 200 OK –

GET `/api/v2/resources/featured/`
Will return a list of featured resource available.

Status Codes

- 200 OK –

GET `/api/v2/resources/published/`
Will return a list of published resource available.

Status Codes

- 200 OK –

GET `/api/v2/resources/resource_types/`
Will return a list of the resource types that are managed by GeoNode.

Status Codes

- 200 OK –

GET `/api/v2/schema/`
OpenApi3 schema for this API. Format can be selected via content negotiation.

- YAML: application/vnd.oai.openapi
- JSON: application/vnd.oai.openapi+json

Query Parameters

- **format** (*string*) –
- **lang** (*string*) –

Status Codes

- 200 OK –

GET `/api/v2/uploads/`
List of the upload instances with their status.

Query Parameters

- **ordering** (*string*) – Which field to use when ordering the results.
- **page** (*integer*) – A page number within the paginated result set.
- **page_size** (*integer*) – Number of results to return per page.
- **search** (*string*) – A search term.

Status Codes

- 200 OK –

POST `/api/v2/uploads/`
Either create a single or many model instances in bulk using the Serializer's many=True ability from Django REST >= 2.2.5.

The data can be represented by the serializer name (single or plural forms), dict or list.

Examples:

```
POST /dogs/ {  
  «name»: «Fido», «age»: 2
```

```

}
POST /dogs/ {
  «dog»: { «name»: «Lucky», «age»: 3
}
}
POST /dogs/ {
  «dogs»: [ { «name»: «Fido», «age»: 2}, { «name»: «Lucky», «age»: 3}
]
}
POST /dogs/ [
  { «name»: «Fido», «age»: 2}, { «name»: «Lucky», «age»: 3}
]

```

Status Codes

- 201 Created –

PATCH /api/v2/uploads/

API endpoint that allows uploads to be viewed or edited.

Status Codes

- 200 OK –

DELETE /api/v2/uploads/

Either delete a single or many model instances in bulk

```

DELETE /dogs/ {
  «dogs»: [ { «id»: 1}, { «id»: 2}
]
}
DELETE /dogs/ [
  { «id»: 1}, { «id»: 2}
]

```

Status Codes

- 204 No Content – No response body

GET /api/v2/uploads/{id}/

Authenticated API endpoint that will show details about the upload status. Main informations available are:

- ID: upload id
- name: layer name
- date: start date of the import process
- state: the actual state of the import process. When the state is PROCESSED, means that the layer has been process by GeoNode and Geoserver
- progress: % of progress

- `uploadfile_set`: list of the uploaded files
- `delete_url`: url to be called for delete the import
- `import_url`: to monitoring the import status in geoserver.

Example of still in progress import:

```
{
  «upload»: { «id»: 1119, «name»: «layer_name», «date»: «2021-04-26T09:50:41.233543Z», «create_date»: «2021-04-26T09:50:40.515866Z», «user»: 1167, «state»: «COMPLETE», «progress»: 80.0, «complete»: true, «import_id»: 1089, «uploadfile_set»: [], «resume_url»: null, «delete_url»: «/upload/delete/1119», «import_url»: «http://geoserver:8080/geoserver/rest/imports/1089», «detail_url»: null
  }
}
```

Parameters

- `id` (*integer*) – A unique integer value identifying this upload.

Status Codes

- **200 OK** –

PUT `/api/v2/uploads/{id}/`

Update one or more model instances.

If `ENABLE_BULK_UPDATE` is set, multiple previously-fetched records may be updated in a single call, provided their IDs.

If `ENABLE_PATCH_ALL` is set, multiple records may be updated in a single PATCH call, even without knowing their IDs.

WARNING: `ENABLE_PATCH_ALL` should be considered an advanced feature and used with caution. This feature must be enabled at the viewset level and must also be requested explicitly by the client via the `«patch-all»` query parameter.

This parameter can have one of the following values:

true (or 1): records will be fetched and then updated in a transaction loop

- The `Model.save` method will be called and model signals will run
- This can be slow if there are too many signals or many records in the query
- This is considered the more safe and default behavior

query: records will be updated in a single query

- The `QuerySet.update` method will be called and model signals will not run
- This will be fast, but may break data constraints that are controlled by signals
- This is considered unsafe but useful in certain situations

The server's successful response to a patch-all request will NOT include any individual records. Instead, the response content will contain a `«meta»` object with an `«updated»` count of updated records.

Examples:

Update one dog:

```
PATCH /dogs/1/ {
```



```
    "fur": "white"
  }
}
```

Update many dogs by ID:

```
PATCH /dogs/ [
  { "id": 1, "fur": "white" }, { "id": 2, "fur": "black" }, { "id": 3, "fur": "yellow" }
]
```

Update all dogs in a query:

```
PATCH /dogs/?filter{fur.contains}=brown&patch-all=true {
  "fur": "gold"
}
```

Parameters

- **id** (*integer*) – A unique integer value identifying this upload.

Status Codes

- **200 OK** –

PATCH /api/v2/uploads/{id}/

API endpoint that allows uploads to be viewed or edited.

Parameters

- **id** (*integer*) – A unique integer value identifying this upload.

Status Codes

- **200 OK** –

DELETE /api/v2/uploads/{id}/

Either delete a single or many model instances in bulk

```
DELETE /dogs/ {
  «dogs»: [ { «id»: 1 }, { «id»: 2 }
]
}

DELETE /dogs/ [
  { «id»: 1 }, { «id»: 2 }
]
```

Parameters

- **id** (*integer*) – A unique integer value identifying this upload.

Status Codes

- **204 No Content** – No response body

GET /api/v2/uploads/{id}/{field_name}/

Fetch related object(s), as if sideloaded (used to support link objects).

This method gets mapped to `/<resource>/<pk>/<field_name>/` by `DynamicRouter` for all `DynamicRelationField` fields. Generally, this method probably shouldn't be overridden.

An alternative implementation would be to generate reverse queries. For an exploration of that approach, see:

<https://gist.github.com/ryochiji/54687d675978c7d96503>

Parameters

- **field_name** (*string*) –
- **id** (*integer*) – A unique integer value identifying this upload.

Status Codes

- 200 OK –

PUT /api/v2/uploads/upload/

API endpoint that allows user to upload new layers into GeoNode. The endpoint is protected via authentication and the request should always come from the same session. The API accept the request with form-data. Is also possible to define some information via json. Behind the scene, geonode will:

- Create an upload session
- Save the layer into GeoNode
- Call async geoserver to save the layer

Example: `client = requests.session() response = client.put(`

```
    «http://localhost:8000/api/v2/uploads/upload/»,    auth=HTTPBasicAuth(username,  
    password), data=params, files=files,
```

```
) Where:
```

```
params = { «permissions»: “{    «users»:    {«AnonymousUser»:    [«view_resourcebase»]} , «groups»:{}” , # layer permissions «time»: «false»,  
    «layer_title»: «layer_title», «time»: «false», «charset»: «UTF-8»,  
    } files = {  
        «filename»: <_io.BufferedReader name=>filename>>  
    }  
}
```

Status Codes

- 201 Created –

PUT /upload/final?id={import_id}

API endpoint that allows user to check the status of new layers into GeoNode. The endpoint is protected via authentication and the request should always come from the same session. Example:

`client = requests.session() response = client.get(«http://localhost:8000/upload/final?id=1») Where:`

```
params = { «permissions»: “{ «users»: {«AnonymousUser»: [«view_resourcebase»]}  
    , «groups»:{}” , # layer permissions «time»: «false», «layer_title»: «layer_title»,  
    «time»: «false», «charset»: «UTF-8»,  
    }  
}
```

Status Codes

- 200 OK –

```
{ «status»: «finished», «id»: 131, «url»: «file_name_path», «bbox»: «-180.0000000,-  
    90.0000000,180.0000000,90.0000000», «crs»: {
```

```

        »type»: «name», «properties»: «EPSG:4326»
      }, «success»: true
    }

```

GET /api/v2/users/

API endpoint that show the list of users registered in GeoNode. if the logged user is the administrator will show all the user available, otherwise only the information of his own account are provided

Query Parameters

- **ordering** (*string*) – Which field to use when ordering the results.
- **page** (*integer*) – A page number within the paginated result set.
- **page_size** (*integer*) – Number of results to return per page.

Status Codes

- 200 OK –

POST /api/v2/users/

Either create a single or many model instances in bulk using the Serializer's many=True ability from Django REST >= 2.2.5.

The data can be represented by the serializer name (single or plural forms), dict or list.

Examples:

POST /dogs/ {

```

    «name»: «Fido», «age»: 2

```

```

}

```

POST /dogs/ {

```

    «dog»: { «name»: «Lucky», «age»: 3

```

```

    }

```

```

}

```

POST /dogs/ {

```

    «dogs»: [ { «name»: «Fido», «age»: 2 }, { «name»: «Lucky», «age»: 3 }

```

```

    ]

```

```

}

```

POST /dogs/ [

```

    { «name»: «Fido», «age»: 2 }, { «name»: «Lucky», «age»: 3 }

```

```

]

```

Status Codes

- 201 Created –

DELETE /api/v2/users/

Either delete a single or many model instances in bulk

DELETE /dogs/ {

```
    «dogs»: [ { «id»: 1 }, { «id»: 2 }
  ]
}
DELETE /dogs/ [
  { «id»: 1 }, { «id»: 2 }
]
```

Status Codes

- **204 No Content** – No response body

GET `/api/v2/users/{id}/`

API endpoint that return detailed information of a specific user.

Parameters

- **id** (*integer*) – A unique integer value identifying this user.

Status Codes

- **200 OK** –

PUT `/api/v2/users/{id}/`

Update one or more model instances.

If `ENABLE_BULK_UPDATE` is set, multiple previously-fetched records may be updated in a single call, provided their IDs.

If `ENABLE_PATCH_ALL` is set, multiple records may be updated in a single `PATCH` call, even without knowing their IDs.

WARNING: `ENABLE_PATCH_ALL` should be considered an advanced feature and used with caution. This feature must be enabled at the viewset level and must also be requested explicitly by the client via the «patch-all» query parameter.

This parameter can have one of the following values:

true (or 1): records will be fetched and then updated in a transaction loop

- The *Model.save* method will be called and model signals will run
- This can be slow if there are too many signals or many records in the query
- This is considered the more safe and default behavior

query: records will be updated in a single query

- The *QuerySet.update* method will be called and model signals will not run
- This will be fast, but may break data constraints that are controlled by signals
- This is considered unsafe but useful in certain situations

The server's successful response to a patch-all request will NOT include any individual records. Instead, the response content will contain a «meta» object with an «updated» count of updated records.

Examples:

Update one dog:

```
PATCH /dogs/1/ {
  "fur": "white"
```

```
}
```

Update many dogs by ID:

```
PATCH /dogs/ [
  {"id": 1, "fur": "white"}, {"id": 2, "fur": "black"}, {"id": 3, "fur": "yellow"}
]
```

Update all dogs in a query:

```
PATCH /dogs/?filter{fur.contains}=brown&patch-all=true {
  "fur": "gold"
}
```

Parameters

- **id** (*integer*) – A unique integer value identifying this user.

Status Codes

- 200 OK –

DELETE /api/v2/users/{id}/

Delete a model instance.

```
DELETE /dogs/ {
  «dogs»: [ {«id»: 1}, {«id»: 2}
]
}
```

```
DELETE /dogs/ [
  {«id»: 1}, {«id»: 2}
]
```

Parameters

- **id** (*integer*) – A unique integer value identifying this user.

Status Codes

- 204 No Content – No response body

GET /api/v2/users/{id}/groups/

API endpoint that show the list of the groups that the user belong to.

Parameters

- **id** (*integer*) – A unique integer value identifying this user.

Status Codes

- 200 OK –

GET /api/v2/users/{id}/resources/

API endpoint that show the list of the resources that belong to the user.

Parameters

- **id** (*integer*) – A unique integer value identifying this user.

Status Codes

- 200 OK –

GET /mapstore/rest/resources/

Only Authenticate User perform CRUD Operations on Respective Data

Status Codes

- 200 OK – No response body

POST /mapstore/rest/resources/

Only Authenticate User perform CRUD Operations on Respective Data

Status Codes

- 201 Created – No response body

GET /mapstore/rest/resources/{id}/

Only Authenticate User perform CRUD Operations on Respective Data

Parameters

- **id** (*integer*) – A unique value identifying this map store resource.

Status Codes

- 200 OK – No response body

PUT /mapstore/rest/resources/{id}/

Only Authenticate User perform CRUD Operations on Respective Data

Parameters

- **id** (*integer*) – A unique value identifying this map store resource.

Status Codes

- 200 OK – No response body

PATCH /mapstore/rest/resources/{id}/

Only Authenticate User perform CRUD Operations on Respective Data

Parameters

- **id** (*integer*) – A unique value identifying this map store resource.

Status Codes

- 200 OK – No response body

DELETE /mapstore/rest/resources/{id}/

Only Authenticate User perform CRUD Operations on Respective Data

Parameters

- **id** (*integer*) – A unique value identifying this map store resource.

Status Codes

- 204 No Content – No response body

GET /mapstore/rest/users/

API endpoint that show the users available with some details.

Status Codes

- 200 OK –

POST /mapstore/rest/users/

API endpoint that allows users to be viewed or edited.

Status Codes

- 201 Created –

GET /mapstore/rest/users/{id}/

API endpoint that show the users available with some details.

Parameters

- **id** (*integer*) – A unique integer value identifying this user.

Status Codes

- 200 OK –

PUT /mapstore/rest/users/{id}/

API endpoint that allows users to be viewed or edited.

Parameters

- **id** (*integer*) – A unique integer value identifying this user.

Status Codes

- 200 OK –

DELETE /mapstore/rest/users/{id}/

API endpoint that allows users to be viewed or edited.

Parameters

- **id** (*integer*) – A unique integer value identifying this user.

Status Codes

- 204 No Content – No response body

GET /o/userinfo/

View used to show Claims about the authenticated End-User

Status Codes

- 200 OK – No response body

1.36 How to Develop

1.36.1 Start to develop with Docker

How to run the instance for development

Set the variable **SET_DOCKER_ENV** for development

```
vi .env
```

Change to

```
SET_DOCKER_ENV=development
```

Use dedicated docker-compose files while developing

Nota: In this example we are going to keep `localhost` as the target IP for GeoNode

```
docker-compose -f docker-compose.async.yml -f docker-compose.development.yml up
```

How to debug

Nota: We are supposing to use `ipdb` for debugging which is already available as package from the container

Stop the container for the `django` service:

```
docker-compose stop django
```

Run the container again with the option for *service ports*:

```
docker-compose run \
  -e DOCKER_ENV=development \
  -e IS_CELERY=False \
  -e DEBUG=True \
  -e GEONODE_LB_HOST_IP=localhost \
  -e GEONODE_LB_PORT=80 \
  -e SITEURL=http://localhost/ \
  -e ALLOWED_HOSTS="['localhost', ]" \
  -e GEOSERVER_PUBLIC_LOCATION=http://localhost/geoserver/ \
  -e GEOSERVER_WEB_UI_LOCATION=http://localhost/geoserver/ \
  --rm --service-ports django python manage.py runserver --settings=geonode.
↳ settings 0.0.0.0:8000
```

Access the site on <http://localhost/>

Nota: If you set an `ipdb` debug point with `import ipdb ; ipdb.set_trace()` then you should be facing its console and you can see the `django` server which is restarting at any change of your code from your local machine.

1.36.2 How to Install GeoNode-Core for development

Summary of installation

This section demonstrates a summarization of the steps to be followed in order to install GeoNode-Core for development using Ubuntu 18.04. The following steps will be customized to fit both GeoNode-Project and GeoNode-Core for development purpose.

The steps to be followed are:

- 1- Install build tools and libraries
- 2- Install dependencies and supporting tools
- 3- Setup Python virtual environment
- 4- Clone and install GeoNode from Github
- 5- Install and start Geoserver
- 6- Start GeoNode

Nota: The following commands/steps will be executed on your terminal

Aviso: If you have a running GeoNode service, you will need to stop it before starting the following steps. To stop GeoNode you will need to run:

```
service apache2 stop    # or your installed server
service tomcat7 stop    # or your version of tomcat
```

Install GeoNode-Core for development

GeoNode-Core installation is considered the most basic form of GeoNode. It doesn't require any external server to be installed and it can run locally against a file-system based Spatialite database.

Installation steps

- 1- Install build tools and libraries

Aviso: Those instructions might be outdated. Please refer to *1. Install the dependencies*

```
$ sudo apt-get install -y build-essential libxml2-dev libxslt1-dev libpq-dev zlib1g-
↳dev
```

- 2- Install dependencies and supporting tools
 - Install python native libraries and tools

Aviso: Those instructions might be outdated. Please refer to *1. Install the dependencies*

```
$ sudo apt-get install -y python3-dev python3-pil python3-lxml python3-pyproj python3-
↳shapely python3-nose python3-httpplib2 python3-pip software-properties-common
```

Install python virtual environment

Aviso: Those instructions might be outdated. Please refer to [2. GeoNode Installation](#)

```
$ sudo pip install virtualenvwrapper
```

Install postgresql and postgis

Aviso: Those instructions might be outdated. Please refer to [3. Postgis database Setup](#)

```
$ sudo apt-get install postgresql-10 postgresql-10-postgis-2.4
```

Change postgres password expiry and set a password

```
$ sudo passwd -u postgres # change password expiry information
$ sudo passwd postgres # change unix password for postgres
```

Create geonode role and database

```
$ su postgres
$ createdb geonode_dev
$ createdb geonode_dev-imports
$ psql
$ postgres=#
$ postgres=# CREATE USER geonode_dev WITH PASSWORD 'geonode_dev'; # should be same as
↳ password in setting.py
$ postgres=# GRANT ALL PRIVILEGES ON DATABASE "geonode_dev" to geonode_dev;
$ postgres=# GRANT ALL PRIVILEGES ON DATABASE "geonode_dev-imports" to geonode_dev;
$ postgres=# \q
$ psql -d geonode_dev-imports -c 'CREATE EXTENSION postgis;'
$ psql -d geonode_dev-imports -c 'GRANT ALL ON geometry_columns TO PUBLIC;'
$ psql -d geonode_dev-imports -c 'GRANT ALL ON spatial_ref_sys TO PUBLIC;'
$ exit
```

Edit PostgreSQL configuration file

```
sudo gedit /etc/postgresql/10/main/pg_hba.conf
```

Scroll to the bottom of the file and edit this line

```
# "local" is for Unix domain socket connections only
local    all             all                                peer
```

To be as follows

```
# "local" is for Unix domain socket connections only
local    all             all                                trust
```

Then restart PostgreSQL to make the changes effective

```
sudo service postgresql restart
```

Java dependencies

```
$ sudo apt-get install -y openjdk-11-jdk --no-install-recommends
```

Install supporting tools

```
$ sudo apt-get install -y ant maven git gettext
```

3- Setup Python virtual environment (Here is where Geonode will be running)

Add the virtualenvwrapper to your new environment.

Since we are using Ubuntu, you can add the following settings to your .bashrc file. Please note that the Ubuntu account here is called «geonode». So you will need to change it according to the name you picked.

```
$ echo export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python >> ~/.bashrc
$ echo export WORKON_HOME=/home/geonode/dev/.venvs >> ~/.bashrc
$ echo source /usr/local/bin/virtualenvwrapper.sh >> ~/.bashrc
$ echo export PIP_DOWNLOAD_CACHE=$HOME/.pip-downloads >> ~/.bashrc
```

And reload the settings by running

```
$ source ~/.bashrc
```

Set up the local virtual environment for Geonode

```
$ vim ~/.bashrc
# add the following line to the bottom
$ source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
```

```
$ source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
$ mkvirtualenv --python=/usr/bin/python3 geonode
$ workon geonode # or $ source /home/geonode/dev/.venvs/geonode/bin/activate
This creates a new directory where you want your project to be and creates a new
↳ virtualenvironment
```

Alternatively you can also create the virtual env like below

```
$ python3.8 -m venv /home/geonode/dev/.venvs/geonode
$ source /home/geonode/dev/.venvs/geonode/bin/activate
```

4- Download/Clone GeoNode from Github

To download the latest geonode version from github, the command «git clone» is used

Nota: If you are following the GeoNode training, skip the following command. You can find the cloned repository in /home/geonode/dev

```
$ git clone https://github.com/GeoNode/geonode.git -b 3.2.x
```

Install Nodejs PPA and other tools required for static development

This is required for static development

Nota: If you are following GeoNode's training, nodejs is already installed in the Virtual Machine skip the first three command and jump to cd geonode/geonode/static

```
$ sudo apt-get install nodejs npm
$ cd geonode/geonode/static
$ npm install --save-dev
```

Nota: Every time you want to update the static files after making changes to the sources, go to geonode/static and run ‘grunt production’.

Aviso: Starting from the following step, you have to make sure that you installed GDAL correctly according to the documentation page «Install GDAL for Development»

Install GeoNode in the new active local virtualenv

```
$ cd /home/geonode/dev # or to the directory containing your cloned GeoNode
$ pip install -e geonode
$ cd geonode/geonode
```

Create local_settings.py

Copy the sample file /home/geonode/dev/geonode/geonode/local_settings.py.geoserver.sample and rename it to be local_settings.py

```
$ cd /home/geonode/dev/geonode
$ cp geonode/local_settings.py.geoserver.sample geonode/local_settings.py
$ gedit geonode/local_settings.py
```

In the local_settings.py file, add the following line after the import statements:

```
SITEURL = "http://localhost:8000/"
```

In the DATABASES dictionary under the “default” key, change only the values for the keys NAME, USER and PASSWORD to be as follows:

```
DATABASES = {
'default': {
    'ENGINE': 'django.db.backends.postgresql_psycopg2',
    'NAME': 'geonode_dev',
    'USER': 'geonode_dev',
    'PASSWORD': 'geonode_dev',
    .....
    .....
    ....
    ....
    ...
}...}
```

In the DATABASES dictionary under the “datastore” key, change only the values for the keys NAME, USER and PASSWORD to be as follows:

```
# vector datastore for uploads
'datastore' : {
    'ENGINE': 'django.contrib.gis.db.backends.postgis',
    # 'ENGINE': '', # Empty ENGINE name disables
    'NAME': 'geonode_dev-imports',
    'USER' : 'geonode_dev',
    'PASSWORD' : 'geonode_dev',
    .....
    .....
    .....
```

(continues on next page)

(continuação da página anterior)

```

    ...
    ...
}

```

In the CATALOGUE dictionary under the “default” key, uncomment the USER and PASSWORD keys to activate the credentials for GeoNetwork as follows:

```

CATALOGUE = {
'default': {
    # The underlying CSW implementation
    # default is pyctsw in local mode (tied directly to GeoNode Django DB)
    'ENGINE': 'geonode.catalogue.backends.pyctsw_local',
    # pyctsw in non-local mode
    # 'ENGINE': 'geonode.catalogue.backends.pyctsw_http',
    # GeoNetwork opensource
    # 'ENGINE': 'geonode.catalogue.backends.geonetwork',
    # deegree and others
    # 'ENGINE': 'geonode.catalogue.backends.generic',
    # The FULLY QUALIFIED base url to the CSW instance for this GeoNode
    'URL': urljoin(SITEURL, '/catalogue/csw'),
    # 'URL': 'http://localhost:8080/geonetwork/srv/en/csw',
    # 'URL': 'http://localhost:8080/deegree-csw-demo-3.0.4/services',
    # login credentials (for GeoNetwork)
    'USER': 'admin',
    'PASSWORD': 'admin',
    # 'ALTERNATES_ONLY': True,
}}

```

5- Install and Start Geoserver

From the virtual environment, first you need to align the database structure using the following command :

```

$ cd /home/geonode/dev/geonode
$ python manage.py migrate

```

Aviso: If the start fails because of an import error related to osgeo or libgeos, then please consult the [Install GDAL for Development](#)

then setup GeoServer using the following command:

```

$ paver setup
$ paver sync

```

6- Now we can start our geonode instance

Aviso: Don't forget to stop the GeoNode Production services if enabled

```

service apache2 stop
service tomcat7 stop

```

```
$ paver start
```

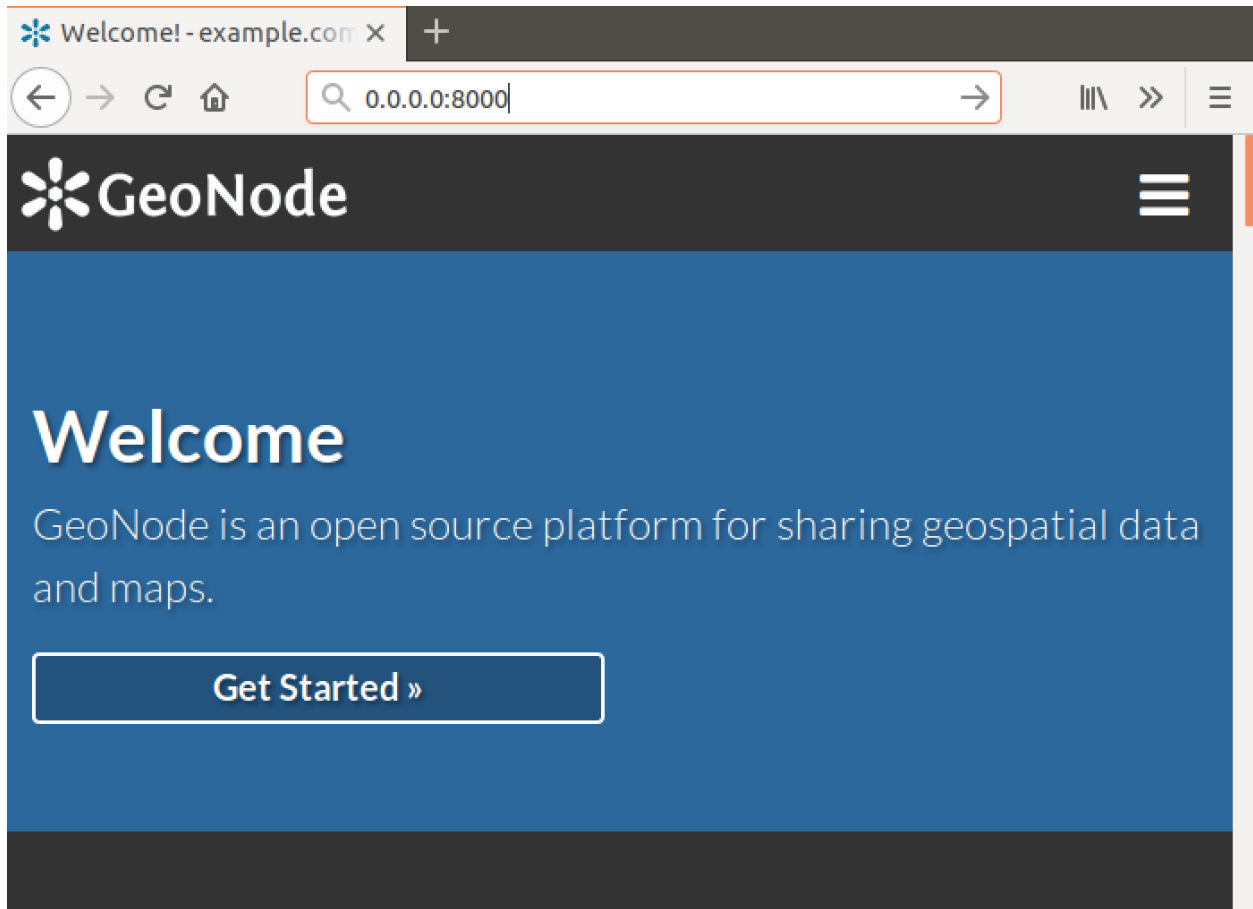
The starting process will take around 20 seconds (depends on your machine) and at the end it shows the following message:

```
[tasks]
. geonode.documents.tasks.create_document_thumbnail
. geonode.documents.tasks.delete_orphaned_document_files
. geonode.documents.tasks.delete_orphaned_thumbnails
. geonode.geoserver.tasks.geoserver_update_layers
. geonode.layers.tasks.delete_layer
. geonode.maps.tasks.delete_map
. geonode.security.tasks.synch_guardian
. geonode.services.tasks.update.harvest_resource
. geonode.tasks.email.send_mail
. geonode.tasks.notifications.send_queued_notifications
. imagekit.cachefiles.backends._generate_file

Performing system checks...

System check identified no issues (1 silenced).
January 21, 2020 - 22:49:50
Django version 1.11.27, using settings 'geonode.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
GeoNode is now available.
```

Now you can visit the geonode site by typing <http://0.0.0.0:8000> into your browser window



Install GeoNode-Project for development after installing GeoNode-Core

Geonode-Project gives the user flexibility to customize the installation of the GeoNode. Geonode itself will be installed as a requirement of your project. Inside the project structure it is possible to extend, replace or modify all geonode components (e.g. css and other static files, templates, models..) and even register new django apps without touching the original Geonode code. In order to install GeoNode-Project, the following steps need to be executed alongside the previous GeoNode-Core installation steps.

1- Use `django-admin.py` to create a project «my_geonode» from a GeoNode-Project template as follows:

Nota: Before running the following command, make sure that you are currently working on the virtual environment and just outside geonode directory. The command will create a new project called «my_geonode» which should be located at the level of geonode-core installation directory «inside `/home/geonode/dev`»

```
$ django-admin.py startproject my_geonode --template=https://github.com/GeoNode/
→geonode-project/archive/master.zip -e py,rst,json,yml,ini,env,sample -n Dockerfile
$ ls /home/geonode/dev # should output: geonode my_geonode
```

Nota: Although the following command might show that the majority of requirements are already satisfied «because GeoNode-Core was already installed», it is recommended to still execute it as it might update or install any missing

package.

2- Install all the required packages/tools for GeoNode-Project as follows:

```
$ pip install -e my_geonode
```

Nota: As mentioned earlier, GeoNode will be installed as requirement for the GeoNode-Project in order to be able to extend it

Install GeoNode-Project directly from scratch

If you didn't install GeoNode-Core earlier and you wanted to install GeoNode-Project directly, please follow these steps

1- Create a virtual environment as follows:

```
$ vim ~/.bashrc
# add the following line to the bottom
$ source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
```

```
$ source /usr/share/virtualenvwrapper/virtualenvwrapper.sh
$ mkvirtualenv --python=/usr/bin/python3 my_geonode
```

Alternatively you can also create the virtual env like below

```
$ python3.8 -m venv /home/geonode/dev/.venvs/my_geonode
$ source /home/geonode/dev/.venvs/my_geonode/bin/activate
```

2- Clone the geonode-project repo from Github

```
$ git clone https://github.com/GeoNode/geonode-project.git -b 3.2.x
```

3- Install Django framework as follows

```
$ pip install Django==3.2
```

4- Use django-admin.py to create a project «my_geonode» from a GeoNode-Project template as follows:

```
$ django-admin startproject --template=./geonode-project -e py,sh,md,rst,json,yml,ini,
↪env,sample,properties -n monitoring-cron -n Dockerfile my_geonode
```

5- Install all the requirements for the GeoNode-Project and install the GeoNode-Project using pip

```
$ cd my_geonode
$ pip install -r requirements.txt --upgrade
$ pip install -e . --upgrade
```

6- Install GDAL Utilities for Python

```
$ pip install pygdal=="`gdal-config --version`.`" # or refer to the link <Install_
↪GDAL for Development <https://training.geonode.geo-solutions.it/005_dev_workshop/
↪004_devel_env/gdal_install.html>
```

7- Install GeoServer and Tomcat using paver


```
$ paver setup
$ paver sync
$ paver start
```

8- Visit <http://localhost:8000/>

1.36.3 How to run GeoNode Core for development

In order to start Geonode Core for development, you need to make sure that no Geonode instance is running first. This can be done by running the following commands:

```
$ cd /home/user/geonode
$ paver stop_geoserver
$ paver stop_django
```

Then you need to start both geoserver and django services as follows:

```
$ paver start_geoserver
$ paver start_django
```

Now you can visit your Geonode GUI by typing <http://localhost:8000> into your browser window

1.36.4 How to run GeoNode Project for development

In order to run a project for development, the following steps have to be followed:

1- Make sure there is no running instance of Geonode first by running the following command:

```
$ cd /home/user/my_geonode
$ paver stop
```

The above command will stop all services related to Geonode if running

2- Start the servers by running paver start as follows:

```
$ paver start
```

Now you can visit your geonode project site by typing <http://localhost:8000> into your browser window

1.36.5 Start MapStore2 client in development mode

Pre-requisites

1. You need a running instance of GeoNode somewhere; in this specific example we assume GeoNode is running on `:http://localhost:8000`

Install needed packages

```
sudo apt install nodejs npm
```

Prepare the source code

```
git clone --recursive https://github.com/GeoNode/geonode-mapstore-client.git geonode-  
↪mapstore-client-dev
```

Compile MapStore2 Client

```
cd geonode-mapstore-client/geonode_mapstore_client/client/  
npm update  
npm install  
npm run compile
```

Edit the file `env.json`

```
vim env.json
```

```
{  
  "DEV_SERVER_HOST": "localhost:8000",  
  "DEV_SERVER_HOST_PROTOCOL": "http"  
}
```

Run MapStore2 in Development mode

```
npm run start
```

Connect to `:http://localhost:8081`

This is a proxied version of GeoNode from MapStore2 client. **To upload new layers use the original GeoNode.**

Everytime you render a map, from GeoNode layers details page or map creation, you will access to the MapStore2 dev mode running code.

You can now update the code on the fly.

Example 1: Disable the PrintPlugin from the Layer Details small map

```
vim js/previewPlugins.js
```

```
...
BurgerMenuPlugin: require('../MapStore2/web/client/plugins/BurgerMenu'),
ScaleBoxPlugin: require('../MapStore2/web/client/plugins/ScaleBox'),
MapFooterPlugin: require('../MapStore2/web/client/plugins/MapFooter'),
// PrintPlugin: require('../MapStore2/web/client/plugins/Print'),
TimelinePlugin: require('../MapStore2/web/client/plugins/Timeline'),
PlaybackPlugin: require('../MapStore2/web/client/plugins/Playback'),
...
```

Example 2: Disable the MousePositionPlugin from the big maps

```
vim js/plugins.js
```

```
...
SaveAsPlugin: require('../MapStore2/web/client/plugins/SaveAs').default,
MetadataExplorerPlugin: require('../MapStore2/web/client/plugins/MetadataExplorer'),
GridContainerPlugin: require('../MapStore2/web/client/plugins/GridContainer'),
StyleEditorPlugin: require('../MapStore2/web/client/plugins/StyleEditor'),
TimelinePlugin: require('../MapStore2/web/client/plugins/Timeline'),
PlaybackPlugin: require('../MapStore2/web/client/plugins/Playback'),
// MousePositionPlugin: require('../MapStore2/web/client/plugins/MousePosition'),
SearchPlugin: require('../MapStore2/web/client/plugins/Search'),
SearchServicesConfigPlugin: require('../MapStore2/web/client/plugins/
↪SearchServicesConfig'),
...
```

1.36.6 Workshops

The workshops documentation demonstrates few examples on how to utilize GeoNode-Project in order to extend/customize GeoNode's functionalities according to your business. The covered topics include the following:

- 1- Customize your GeoNode with the geonode-project
- 2- Customize the look and feel
- 3- Create your ResourceBase Metadata
- 4- Create your own django app
- 5- Add a custom model
- 6- Permissions and APIs
- 7- Deploy your GeoNode

1- Customize your GeoNode with the geonode-project

In this example, GeoNode-Project is cloned to create a template instance in which the rest of the examples will be building on top of it.

1- Assuming you already installed GeoNode-Core, firstly we need to create a GeoNode-Project template and this can be achieved from the following command:

```
$ django-admin.py startproject my_geonode --template=https://github.com/GeoNode/  
↳geonode-project/archive/master.zip -e py,rst,json,yml,ini,env,sample -n Dockerfile
```

Here, django-admin is used with startproject option to create my_geonode project copying the template which is passed as GeoNode-project Github repo. It also includes «py,rst,json,yml,ini,env,sample» extensions

2- Once the cloning finished, the next step is to install the GeoNode-Project we just downloaded as follows:

```
$ pip install -e my_geonode
```

3- Install geoserver using paver as follows

```
$ cd /home/geonode/my_geonode  
$ paver setup
```

4- Note the GeoNode database connection parameters mentioned in the local_settings.py configuration file. If not found, copy local_settings.py.sample and rename it to local_settings.py then use psql to create the required user and grant the required privileges as follows:

```
$ su postgres  
$ createdb geonode  
$ psql  
postgres=# CREATE USER geonode WITH PASSWORD 'geonode';  
CREATE ROLE  
postgres=# GRANT ALL PRIVILEGES ON DATABASE "geonode" to geonode;  
GRANT  
postgres=# \q
```

Aviso: Don't forget to exit from postgres user before executing the following commands

5- Run GeoNode using paver

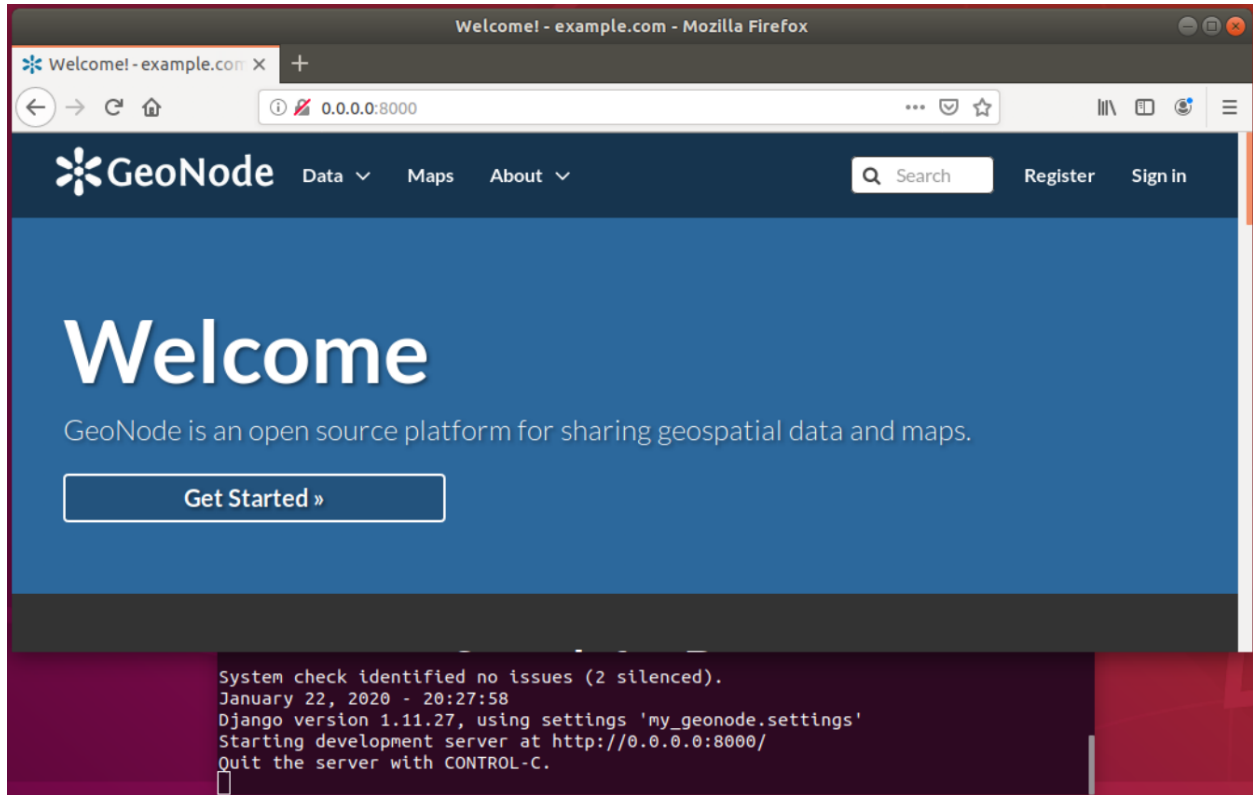
```
$ cd /home/geonode/my_geonode  
$ paver start
```

Nota: You may find this warning message: You have 132 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): account, actstream, admin, agon_ratings, announcements, auth, avatar, base, contenttypes, dialogos, documents, favorite, geonode_client, geonode_themes, groups, guardian, invitations, layers, maps, mapstore2_adapter, monitoring, oauth2_provider, people, pinax_notifications, services, sessions, sites, socialaccount, taggit, tastypie, upload, user_messages. Run “python manage.py migrate” to apply them.

Which means you have some sql statements not executed yet and you need to run the «migrate» to sync your database first then «paver start» again as follows:

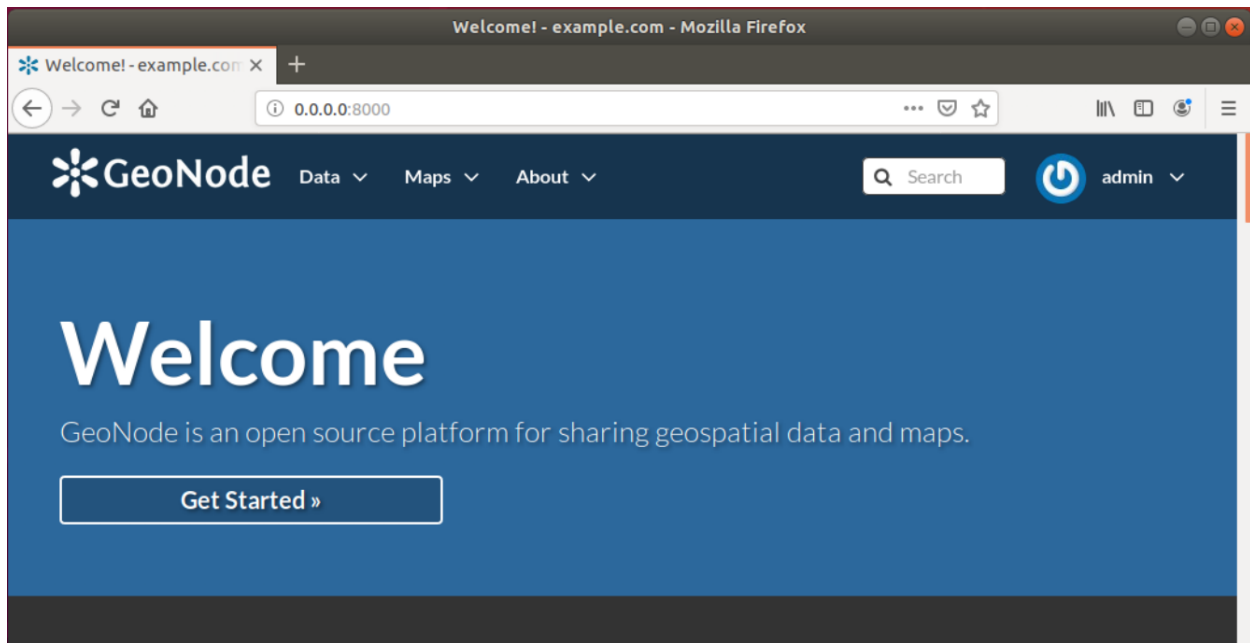
```
$ python manage.py migrate  
$ paver start
```

Aviso: If encountered this message: (Invalid HTTP_HOST header: '0.0.0.0:8000'. You may need to add u'0.0.0.0' to ALLOWED_HOSTS) It can be fixed in the settings.py file. You will need to add: `ALLOWED_HOSTS = ['0.0.0.0']` in settings.py



6- Once the previous step is done, you can visit 0.0.0.0:8000 to view the GUI of GeoNode. However, we still don't have an account in order to login from the GUI. This can be done using «paver sync». The command will create sync with latest fixtures and also creates a superuser «admin» with default password «admin»

7- Use the created account to login from the GUI through localhost:8000 or 0.0.0.0:8000



2- Customize the look and feel

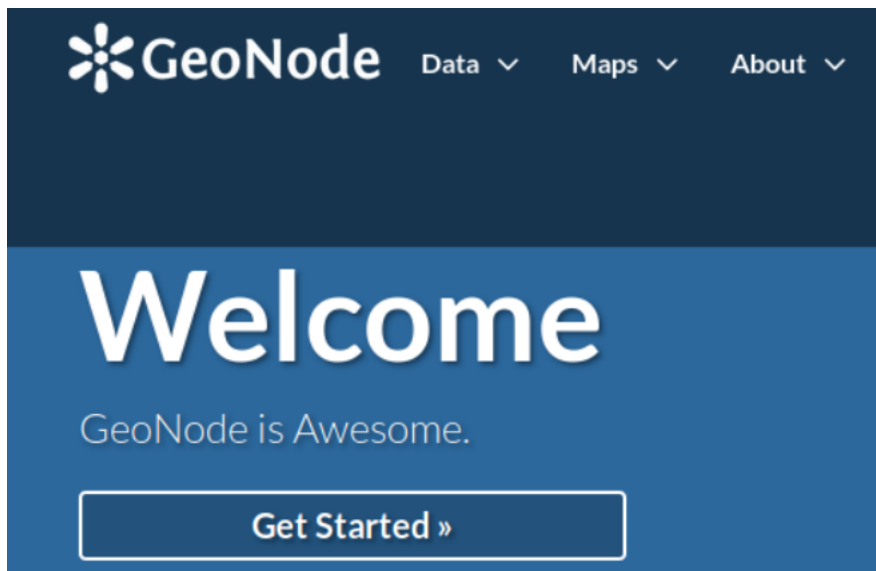
In this section we will change the look and feel of GeoNode, in particular we will do some customization to help understanding how the template inheritance works and how to add new stuff to your GeoNode. The changes will include the home page, the top menu, the footer and a generic GeoNode page.

Homepage:

The geonode-project provides some predefined templates to change the home page and the general site content.

In the «my_geonode/my_geonode/templates» directory we can edit the site_index.html.

Try to edit the content of the «jumbotron» box in the page, save and refresh your browser to see the changes.



The theme:

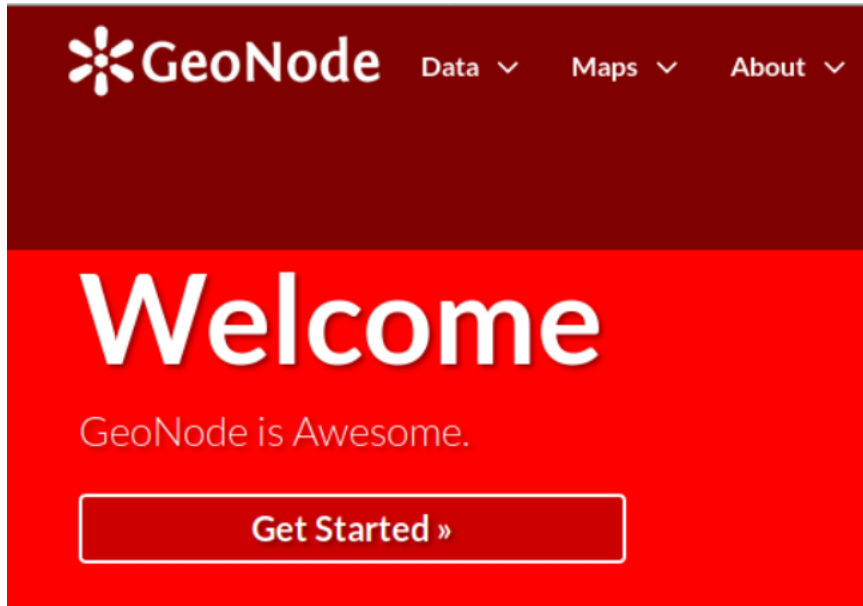
To change the theme of our geonode-project we can act on the `site_base.css` file available in the «`my_geonode/my_geonode/static/css`» folder.

The file is empty so we can inspect elements of the home page with the browser's developer tools and define css rules in there.

For example, if we want to change the background of the jumbotron, in this file we can add

```
.home .jumbotron { background: red }
```

Then once we refreshed the browser, we should see the change as follows:



Adding the `".home"` class is necessary in order to let the rule have precedence/priority over the GeoNode's one. We can see this by inspecting the element in the developer console.

The top menu:

Now we can make some changes that will apply to the whole site. We can add a Geocollections entry in the top menu bar.

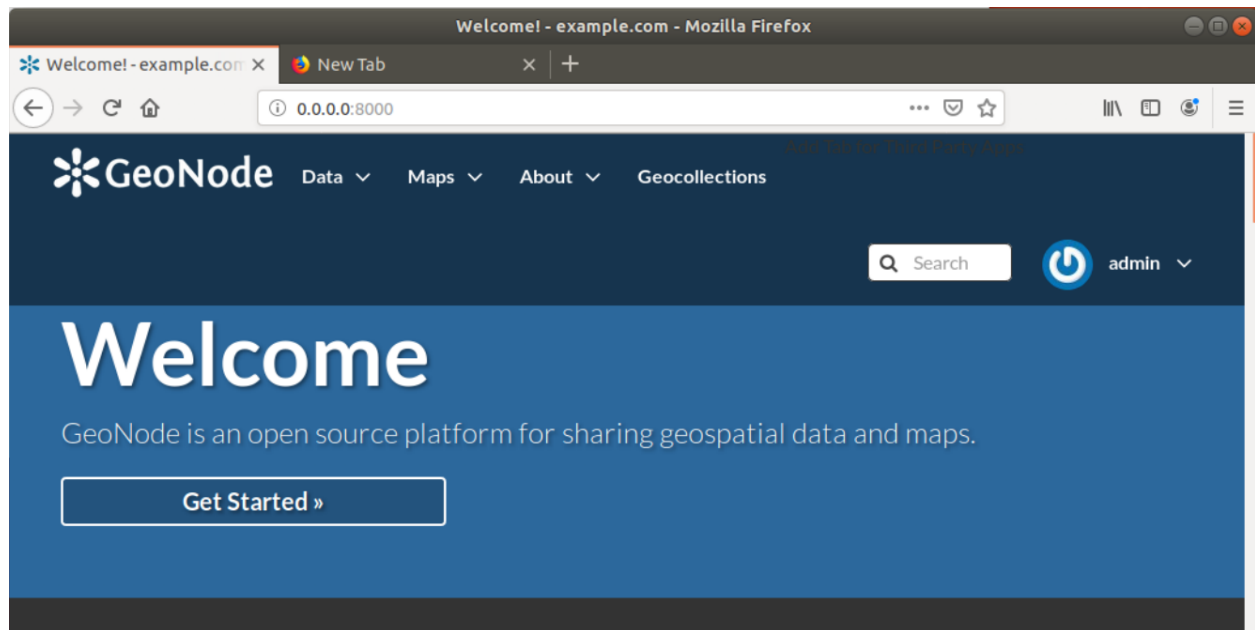
Edit the `site_base.html` file in the templates folder and uncomment the list item adapting the text as well from:

```
{% comment %}
Add Tab for Third Party Apps
<li>
  <a href="{{ PROJECT_ROOT }}/app">App</a>
</li>
{% endcomment %}
```

To:

```
<li>
  <a href="{{ PROJECT_ROOT }}/geocollections">Geocollections</a>
</li>
```

On browser refresh you will see a new entry in the nav bar which is persistent to the whole site.



GeoNode generic page

As you can see in the templates folder there are only the `site_index.html` and the `site_base.html` files. In order to customize another GeoNode page, for example the layers list page, you need to recreate the same folder structure of the GeoNode templates folder and add a file with the same name.

For the layers list page we can create a directory named «layers» inside the template directory and a file named «layer_list.html» inside layers. The changes made in this file will only affect the layer list page.

```
mkdir -p my_geonode/templates/layers/

cp geonode/geonode/layers/templates/layers/layer_list.html my_geonode/templates/
↳layers/layer_list.html

vim my_geonode/templates/layers/layer_list.html
```

For example change in page title to be:

```
<h2 class="page-title">{% trans "Explore My Layers" %}</h2>
```

then refresh the browser to see the update.



Modify functionality

In this section, we will patch the ResourceBase of GeoNode and update the Templates in order to add one more field to the Metadata Schema.

We will add a DOI field to the ResourceBase model and modify the Templates in order to show the new field both into the Metadata Wizard and the Layer Details page.

Nota: Make sure to be inside «my_geonode» directory to execute the following commands

Customizing metadata can be achieved from the model which is defined in the core at «geonode/geonode/base/models.py» as follows:

```
# internal fields
uuid = models.CharField(max_length=36)
owner = models.ForeignKey(
    settings.AUTH_USER_MODEL,
    blank=True,
    null=True,
    related_name='owned_resource',
    verbose_name=_("Owner"))
contacts = models.ManyToManyField(
    settings.AUTH_USER_MODEL,
    through='ContactRole')
title = models.CharField(_('title'), max_length=255, help_text=_(
    'name by which the cited resource is known'))
alternate = models.CharField(max_length=128, null=True, blank=True)
date = models.DateTimeField(
    _('date'),
    default=now,
    help_text=date_help_text)
date_type = models.CharField(
    _('date type'),
    max_length=255,
    choices=VALID_DATE_TYPES,
    default='publication',
    help_text=date_type_help_text)
edition = models.CharField(
```

(continues on next page)

(continuação da página anterior)

```

_('edition'),
max_length=255,
blank=True,
null=True,
help_text=edition_help_text)
abstract = models.TextField(
_('abstract'),
max_length=2000,
blank=True,
help_text=abstract_help_text)
purpose = models.TextField(
_('purpose'),
max_length=500,
null=True,
blank=True,
help_text=purpose_help_text)
maintenance_frequency = models.CharField(
_('maintenance frequency'),
max_length=255,
choices=UPDATE_FREQUENCIES,
blank=True,
null=True,
help_text=maintenance_frequency_help_text)

```

To add fields directly to the ResourceBase Class without actually modifying it, this can be done from «my_geonode/my_geonode/apps.py» file

The «ready» method is invoked at initialization time and can be currently used to tweak your app in several ways

```

class AppConfig(BaseAppConfig):

    name = "my_geonode"
    label = "my_geonode"

    def ready(self):
        super(AppConfig, self).ready()
        run_setup_hooks()

```

Now we will add the «patch_resource_base» method to the AppConfig and execute it from the ready method as follows:

```

from django.db import models
from django.utils.translation import ugettext_lazy as _

class AppConfig(BaseAppConfig):

    name = "my_geonode"
    label = "my_geonode"

    def _get_logger(self):
        import logging
        return logging.getLogger(self.__class__.__module__)

    def patch_resource_base(self, cls):
        self._get_logger().info("Patching Resource Base")
        doi_help_text = _('a DOI will be added by Admin before publication.')

```

(continues on next page)

(continuação da página anterior)

```

doi = models.TextField(
    _('DOI'),
    blank=True,
    null=True,
    help_text=doi_help_text)
cls.add_to_class('doi', doi)

def ready(self):
    super(AppConfig, self).ready()
    run_setup_hooks()

from geonode.base.models import ResourceBase
self.patch_resource_base(ResourceBase)

```

Nota: you will need to perform migrations as follows: - Add field doi to resourcebase

Once you run python manage.py migrate:

```

Running migrations:
Applying announcements.0002_auto_20200119_1257... OK
Applying base.0031_resourcebase_doi... OK
Applying people.0027_auto_20200119_1257... OK

```

Till now, we have patched the DB. however, it is not yet sufficient as we still need to display the added field.

Let's extend the default templates so that we can show the newly added field

Overriding the Metadata Wizard Template Page

Similar to what we have done before in the Templates directory, we will need to create «layouts» directory under «my_geonode/my_geonode/templates». This directory will contain a copy from «geonode/src/geonode/geonode/layers/templates/layouts/panels.html» as follows:

```

$ mkdir -p my_geonode/templates/layouts
$ cp ~/geonode/src/geonode/geonode/layers/templates/layouts/panels.html my_geonode/
↳ templates/layouts/panels.html
$ vim my_geonode/templates/layouts/panels.html

```

Inside panels.html, we will add a new div with text input as follows:

```

{{ layer_form.data_quality_statement }}
</div>
<div>
    <span><label for="{{ layer_form.doi.id }}">{{ layer_form.doi.label }}</
↳ label></span>
    <input id="id_resource-doi" name="resource-doi"
        type="text"
        class="has-external-popover"
        data-container="body"
        data-content="a DOI will be added by Admin before publication." data-
↳ html="true" data-placement="right"
        placeholder="a DOI will be added by Admin before publication."
        value="{{ layer_form.doi.value }}">
    </div>
</div>

```

In addition, we will override the Layer Detail template page as follows:

```
mkdir -p my_geonode/templates/base

cp /home/geo/Envs/geonode/src/geonode/geonode/base/templates/base/_resourcebase_info_
panel.html my_geonode/templates/base/

vim my_geonode/templates/base/_resourcebase_info_panel.html
```

```
<dd><a href="/groups/group/{{ resource.group.name }}/activity/">{{ group }}</a> </dd>

<dt>DOI</dt>
<dd>{{ resource.doi }}</dd>

</dl>
```

Now from the layer details page, you can see the DOI metadata entry per layer

The screenshot shows the GeoNode interface. At the top is the navigation bar with the GeoNode logo and links for Data, Maps, About, and Geocollections. A search bar and a user profile icon labeled 'admin' are also present. The main content area displays the layer title 'low_res_test_dataset_areas64_id_91_2_0' above a map of Africa. To the right of the map is a sidebar with buttons for 'Download Layer', 'Metadata Detail', 'Editing Tools', 'View Layer', and 'Download Metadata'. Below these buttons is a 'Legend' section showing 'Araster style' with a blue square icon. Further down is a 'Maps using this layer' section stating 'This layer is not currently used in any maps.' Below that is a 'Create a map using this layer' section with a 'Create a Map' button. At the bottom of the sidebar is a 'Styles' section stating 'The following styles are associated with this layer. Choose a style to view it in the preview'. Below the map, there is a metadata section with tabs for 'Info', 'Attributes', 'Share', 'Ratings', 'Comments', and 'Favorite'. The 'Info' tab is active, showing details: Title 'low_res_test_dataset_areas64_id_91_2_0', License 'Not Specified', Abstract 'No abstract provided', Publication Date 'Jan. 22, 2020, 9:50 p.m.', Type 'Raster Data', Keywords 'low_res_test_dataset_areas64_id_91_2_0, WCS, WorldImage', Regions 'Global', Owner 'admin', and DOI 'None'.

3- Create your own django app

In this section, we will demonstrate how to create and setup the skeleton of a custom app using the django facilities. The app will add a geocollections functionality to our GeoNode.

The Geocollections app allows to present in a single page, resources and users grouped by a GeoNode Group. We can assign arbitrary resources to a Geocollection, a Group and a name that will be also used to build a dedicated URL.

Nota: Make sure to be inside «my_geonode» directory to execute the following commands

Create the django app

Django gives us an handy command to create apps. We already used startproject to create our geonode-project, now we can use startapp to create the app.

```
python manage.py startapp geocollections
```

This will create a folder named geocollections that contains empty models and views.

We need to add the new app to the INSTALLED_APPS of our project. inside «my_geonode/settings.py» line 54 change:

```
INSTALLED_APPS += (PROJECT_NAME,) to be: INSTALLED_APPS += (PROJECT_NAME,
↳ 'geocollections',)
```

Add a custom model

In this section, we will add a custom model and the related logic as follows:

- Add a new model
- Add urls and views
- Add admin panel
- Add the template

```
vim geocollections/models.py
```

```
from django.db import models

from geonode.base.models import ResourceBase
from geonode.groups.models import GroupProfile

class Geocollection(models.Model):
    """
    A collection is a set of resources linked to a GeoNode group
    """
    group = models.ForeignKey(GroupProfile, related_name='group_collections')
    resources = models.ManyToManyField(ResourceBase, related_name='resource_
↳ collections')
    name = models.CharField(max_length=128, unique=True)
    slug = models.SlugField(max_length=128, unique=True)

    def __unicode__(self):
        return self.name
```

At this point we need to ask django to create the database table. Django since version 1.8 has embedded migrations mechanism and we need to use them in order to change the state of the db.

Nota: Make sure to be inside «my_geonode» directory to execute the following commands

```
python manage.py makemigrations

# the above command informs you with the migrations to be executed on the database

python manage.py migrate
```

Next we will use django generic view to show the collections detail. Add the following code in the views.py file:

```
vim geocollections/views.py
```

```
from django.views.generic import DetailView

from .models import Geocollection

class GeocollectionDetail(DetailView):
    model = Geocollection
```

Add url configuration

In order to access the created view we also need url mapping. We can create a `urls.py` file containing a url mapping to our generic view:

```
vim geocollections/urls.py
```

```
from django.conf.urls import url

from .views import GeocollectionDetail

urlpatterns = [
    url(r'^(?P<slug>[-\w]+)/$',
        GeocollectionDetail.as_view(),
        name='geocollection-detail'),
]
```

We also need to register the app urls in the project urls. So let's modify the «my_geonode» `urls.py` file adding the following:

```
vim my_geonode/urls.py
```

```
...
urlpatterns += [
    ## include your urls here
    url(r'^geocollections/', include('geocollections.urls')),
]
...
```

Enable the admin panel

We need a user interface where we can create geocollections. Django makes this very easy, we just need the `admin.py` file as follows:

```
vim geocollections/admin.py
```

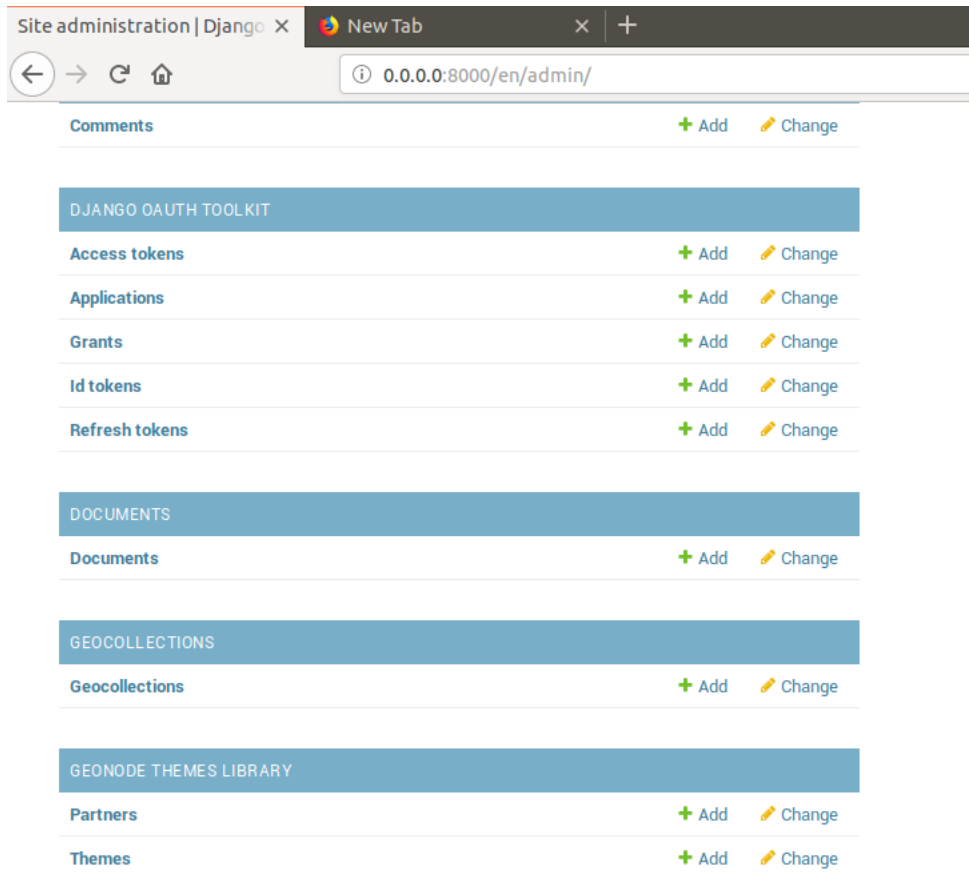
```
from django.contrib import admin

from .models import Geocollection

class GeocollectionAdmin(admin.ModelAdmin):
    prepopulated_fields = {"slug": ("name",)}
    filter_horizontal = ('resources',)

admin.site.register(Geocollection, GeocollectionAdmin)
```

Now we can visit the admin page and create a geocollection from there as follows:



Adding the template

Now we need the template where the geocollection detail will be rendered. Let's create a geocollections directory inside the «my_geonode/templates» directory with a file named geocollection_detail.html:

```
mkdir -p my_geonode/templates/geocollections/

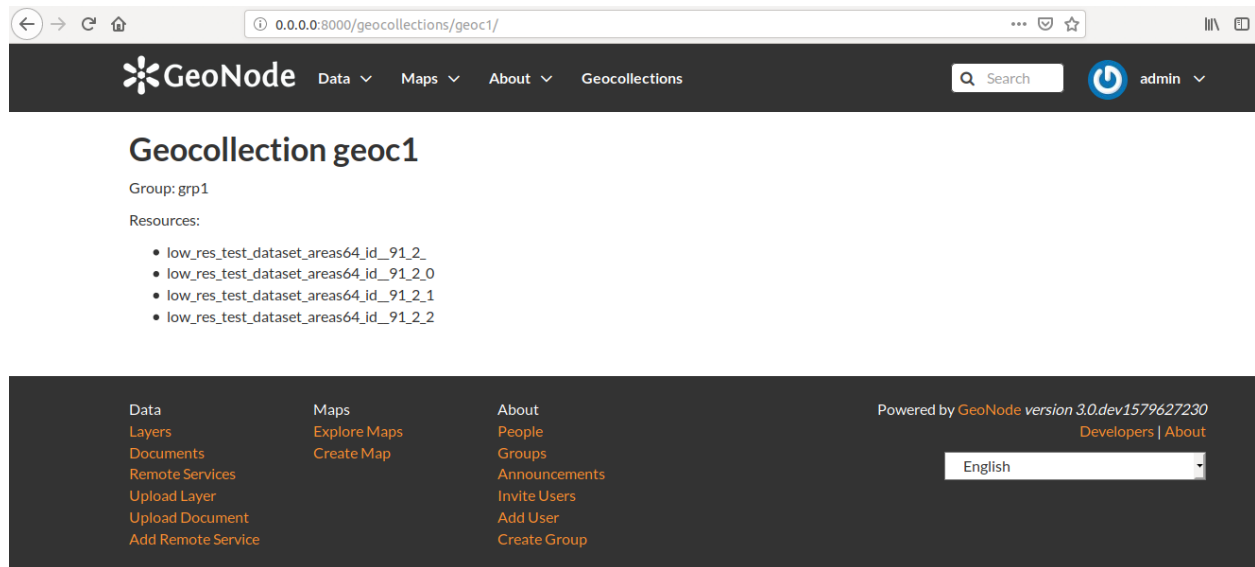
vim my_geonode/templates/geocollections/geocollection_detail.html
```

```
{% extends "geonode_base.html" %}
{% block body %}
    <h2>Geocollection {{ object.name }}</h2>
    <p>Group: {{ object.group.title }}</p>
    <p>Resources:</p>
    <ul>
        {% for resource in object.resources.all %}
            <li>{{ resource.title }}</li>
        {% endfor %}
    </ul>
{% endblock %}
```

To check the results, create a group in the geonode ui interface and load one or more layers/documents

login into the admin panel -> geocollections and create a geocollections

Visit <http://localhost:8000/geocollections/<the-name-of-the-created-geocollection>> and view the results.



Now you know how to customize an html template so you can tune this page as you prefer.

Permissions and APIs

In this section we will add some more advanced logic like permissions and APIs. The permissions in GeoNode are managed with django-guardian, a library which allow to set object level permissions (django has table level authorization).

The APIs are implemented through django-tastypie.

The topics to be covered include:

- Permissions on who can view the geocollection
- How to add templated and js to embed a permission ui in our geocollection detail page
- API to serve json serialized searchable endpoint

Permissions logic (permissions objects)

We need to add the permissions object to the database. We can do this by adding the following meta class to our Geocollection model, guardian will take care of creating the objects for us.

```
vim geocollections/models.py
```

```
class Meta:
    permissions = (
        ('view_geocollection', 'Can view geocollection'),
    )
```

Then run «python manage.py makemigrations» and «python manage.py migrate» to install them

Permissions logic (set_default)

Let's add a method that will be used to set the default permissions on the Geocollections. We can add this logic to the Geocollection model but could also be a generic Mix-in similar to how it is implemented in GeoNode.

```
vim geocollections/models.py
```



```

from django.contrib.auth.models import Group
from django.contrib.auth import get_user_model
from django.contrib.contenttypes.models import ContentType
from django.conf import settings
from guardian.shortcuts import assign_perm

def set_default_permissions(self):
    """
    Set default permissions.
    """

    self.remove_object_permissions()

    # default permissions for anonymous users
    anonymous_group, created = Group.objects.get_or_create(name='anonymous')

    if settings.DEFAULT_ANONYMOUS_VIEW_PERMISSION:
        assign_perm('view_geocollection', anonymous_group, self)

    # default permissions for group members
    assign_perm('view_geocollection', self.group, self)

```

Permissions logic (methods)

Now we need a method to add generic permissions, we want to be able to assign view permissions to groups and single users. We can add this to our Geocollection model

```
vim geocollections/models.py
```

```

def set_permissions(self, perm_spec):
    anonymous_group = Group.objects.get(name='anonymous')
    self.remove_object_permissions()
    if 'users' in perm_spec and "AnonymousUser" in perm_spec['users']:
        assign_perm('view_geocollection', anonymous_group, self)
    if 'users' in perm_spec:
        for user, perms in perm_spec['users'].items():
            user = get_user_model().objects.get(username=user)
            assign_perm('view_geocollection', user, self)
    if 'groups' in perm_spec:
        for group, perms in perm_spec['groups'].items():
            group = Group.objects.get(name=group)
            assign_perm('view_geocollection', group, self)

def remove_object_permissions(self):
    from guardian.models import UserObjectPermission, GroupObjectPermission
    UserObjectPermission.objects.filter(content_type=ContentType.objects.get_for_
↵model(self),
                                object_pk=self.id).delete()
    GroupObjectPermission.objects.filter(content_type=ContentType.objects.get_for_
↵model(self),
                                object_pk=self.id).delete()

```

Permissions logic (views.py)

We can add now a view to receive and set our permissions, in views.py:

```
vim geocollections/views.py
```

```

import json
from django.core.exceptions import PermissionDenied
from django.http import HttpResponse
from django.contrib.auth import get_user_model

User = get_user_model()

def geocollection_permissions(request, collection_id):

    collection = Geocollection.objects.get(id=collection_id)
    user = User.objects.get(id=request.user.id)

    if user.has_perm('view_geocollection', collection):
        return HttpResponse(
            'You have the permission to view. please customize a template for this view'
            ↪,
            content_type='text/plain')

    if request.method == 'POST':
        success = True
        message = "Permissions successfully updated!"
        try:
            permission_spec = json.loads(request.body)
            collection.set_permissions(permission_spec)

            return HttpResponse(
                json.dumps({'success': success, 'message': message}),
                status=200,
                content_type='text/plain'
            )
        except:
            success = False
            message = "Error updating permissions :("
            return HttpResponse(
                json.dumps({'success': success, 'message': message}),
                status=500,
                content_type='text/plain'
            )

```

Permissions logic (url)

Lastly we need a url to map our client to our view, in urls.py

```
vim geocollections/urls.py
```

```

from django.conf.urls import url

from .views import GeocollectionDetail, geocollection_permissions

urlpatterns = [
    url(r'^(?P<slug>[-\w]+)/$',
        GeocollectionDetail.as_view(),
        name='geocollection-detail'),

    url(r'^permissions/(?P<collection_id>\d+)$',
        geocollection_permissions,

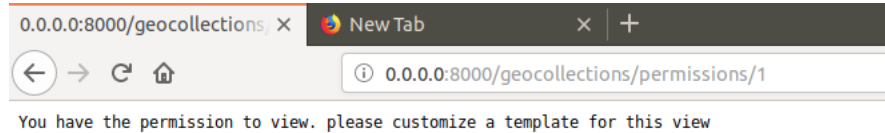
```

(continues on next page)

(continuação da página anterior)

```
name='geocollection_permissions'),
]
```

This url will be called with the id of the geocollection, the id will be passed to the view in order to get the permissions.



Aviso: A note on the client part, the server side logic is just one part necessary to implement permissions.

A checklist of what is necessary:

- A template snippet that can be embedded in the geocollection_detail.html, you can copy and simplify: _permissions_form.html and _permissions.html (in geonode/templates)
- A javascript file that will collect permissions settings and send them to the server, you can copy and simplify: _permissions_form_js.html (in geonode/templates)

API

The GeoNode API system easily allows to plug in new APIs. This section demonstrates the required steps:

We need first to create an api.py file in our geocollection app.

```
vim geocollections/api.py
```

```
import json
from tastypie.resources import ModelResource
from tastypie import fields
from tastypie.constants import ALL_WITH_RELATIONS, ALL

from geonode.api.api import ProfileResource, GroupResource
from geonode.api.resourcebase_api import ResourceBaseResource

from .models import Geocollection
class GeocollectionResource(ModelResource):

    users = fields.ToManyField(ProfileResource, attribute=lambda bundle: bundle.obj.
    group.group.user_set.all(), full=True)
    group = fields.ToOneField(GroupResource, 'group', full=True)
    resources = fields.ToManyField(ResourceBaseResource, 'resources', full=True)

    class Meta:
        queryset = Geocollection.objects.all().order_by('-group')
        ordering = ['group']
        allowed_methods = ['get']
        resource_name = 'geocollections'
        filtering = {
            'group': ALL_WITH_RELATIONS,
            'id': ALL
        }
```

API authorization

We want the API to respect our custom permissions, we can easily achieve this by adding the following to the beginning of `api.py`:

```
vim geocollections/api.py
```

```
from tastypie.authorization import DjangoAuthorization
from guardian.shortcuts import get_objects_for_user

class GeocollectionAuth(DjangoAuthorization):

    def read_list(self, object_list, bundle):
        permitted_ids = get_objects_for_user(
            bundle.request.user,
            'geocollections.view_geocollection').values('id')

        return object_list.filter(id__in=permitted_ids)

    def read_detail(self, object_list, bundle):
        return bundle.request.user.has_perm(
            'view_geocollection',
            bundle.obj)
```

And this to the `GeocollectionResource` Meta class:

```
authorization = GeocollectionAuth()
```

Add a url for our API

In order to publish our API we need a url and we want that url to appear under the GeoNode's `/api` domain.

The final url for our API has to be `/api/geocollections`.

We can inject the url into the GeoNode API by adding the following lines to «`my_geonode/urls.py`» file:

```
vim my_geonode/urls.py
```

```
from geonode.api.urls import api

from geocollections.api import GeocollectionResource

api.register(GeocollectionResource())
```

And add the following in the `urlpatterns`:

```
url(r'', include(api.urls)),
```

The final result will be:

```
from django.conf.urls import url, include
from django.views.generic import TemplateView

from geonode.urls import urlpatterns

from geonode.api.urls import api
from geocollections.api import GeocollectionResource
```

(continues on next page)

(continuação da página anterior)

```
api.register(GeocollectionResource())

urlpatterns += [
    ## include your urls here
    url(r'', include(api.urls)),
    url(r'^geocollections/', include('geocollections.urls')),
]
```

Let's test permissions on API

We can test the permissions on API by manually set a permission from the command line and check that the API respects it.

With running `python manage.py shell` from inside our «my_geonode» folder, it opens a geonode shell.

A perm spec could look like this:

```
perms = {
    'users': {
        'AnonymousUser': ['view_geocollection'],
        'alessio': ['view_geocollection']}
}
```

and we can assign the permissions with:

```
from geocollections.models import Geocollection

Geocollection.objects.first().set_permissions(perms)
```

our <http://localhost:8000/api/geocollections> should now list the geocollection.

If you remove the “AnonymousUser” line from perms and assign again the permissions it will disappear.

```
perms = {
    'users': {
        'alessio': ['view_geocollection']}
}
```

Deploy your GeoNode

So far we demonstrated how to modify, extend and style our GeoNode in dev mode but now it's time to go on production. In this section we will clarify how to:

- commit your work on GitHub
- setup your server
- setup your GeoNode for production

Push to GitHub It is always a good practice to keep your code in a remote repository, GitHub is one of the options and is indeed the most used.

It is assumed that you already have a GitHub account and that you have git installed and configured with your name and email.

We will push only the my_geonode folder to GitHub and as we knew earlier, GeoNode for us is a dependency and we'll just reinstall it as it is on the server.

Steps to push your code to GitHub:

- Create an empty repository in GitHub and copy it's address

- In my_geonode, run `git init` to initialize an empty repository
- Add your remote repository address with `git remote add yourname yourremoteaddress`
- edit `.gitignore` adding all `*.pyc` files
- `git add *` to add all content of my_geonode
- `git commit -m 'initial import'` to make the initial commit
- `git push yourname master` to push the code to the GitHub repository

Setup the server

There are several options for deploying GeoNode projects on servers. In this section, we explain how to deploy it on Ubuntu server 18.04 using system-wide installation

Nota: For quick installation, follow the INSTALLING documentation at <http://docs.geonode.org/en/master/install/core/index.html>

Setup our my_geonode

We need now to install the developed «my_geonode» project following these steps:

- `git clone` from your repository (in the folder of your preference)
- `sudo pip install -e my_geonode`
- edit the settings where needed
- edit `/etc/apache2/sites-enabled/geonode.conf` replacing the wsgi path to the `my_geonode/my_geonode/wsgi.py` file
- add the apache rights to the «my_geonode» folder with a directory like

```
<Directory "/path/to/my_geonode/">  
    Order allow,deny  
    Require all granted  
</Directory>
```

- Test your server.

This documentation helps developers to install GeoNode-Core and GeoNode-Project from different scenarios. GeoNode-Project can be installed on top of GeoNode-Core if already installed. Also GeoNode-Project can be installed from scratch as it has GeoNode-Core as a prerequisite.

HTTP Routing Table

/api

GET /api/v2/, 652	GET /api/v2/uploads/{id}/{field_name}/, 677
GET /api/v2/documents/, 652	GET /api/v2/users/, 679
GET /api/v2/documents/{id}/, 653	GET /api/v2/users/{id}/, 680
GET /api/v2/documents/{id}/linked_resources/, 654	GET /api/v2/users/{id}/groups/, 681
GET /api/v2/geoapps/, 655	GET /api/v2/users/{id}/resources/, 681
GET /api/v2/geoapps/{id}/, 656	POST /api/v2/documents/, 652
GET /api/v2/geoapps/{id}/{field_name}/, 658	POST /api/v2/geoapps/, 655
GET /api/v2/geostories/, 658	POST /api/v2/geostories/, 658
GET /api/v2/geostories/{id}/, 659	POST /api/v2/groups/, 661
GET /api/v2/geostories/{id}/{field_name}/, 661	POST /api/v2/layers/, 664
GET /api/v2/groups/, 661	POST /api/v2/maps/, 668
GET /api/v2/groups/{id}/, 662	POST /api/v2/resources/, 671
GET /api/v2/groups/{id}/managers/, 664	POST /api/v2/uploads/, 674
GET /api/v2/groups/{id}/members/, 664	POST /api/v2/users/, 679
GET /api/v2/groups/{id}/resources/, 664	PUT /api/v2/documents/{id}/, 653
GET /api/v2/layers/, 664	PUT /api/v2/geoapps/{id}/, 656
GET /api/v2/layers/{id}/, 665	PUT /api/v2/geostories/{id}/, 659
GET /api/v2/layers/{id}/{field_name}/, 667	PUT /api/v2/groups/{id}/, 662
GET /api/v2/maps/, 667	PUT /api/v2/layers/{id}/, 666
GET /api/v2/maps/{id}/, 669	PUT /api/v2/maps/{id}/, 669
GET /api/v2/maps/{id}/layers/, 670	PUT /api/v2/resources/{id}/, 672
GET /api/v2/maps/{id}/local_layers/, 670	PUT /api/v2/resources/{id}/set_perms/, 673
GET /api/v2/resources/, 670	PUT /api/v2/uploads/upload/, 678
GET /api/v2/resources/approved/, 673	PUT /api/v2/uploads/{id}/, 676
GET /api/v2/resources/featured/, 673	PUT /api/v2/users/{id}/, 680
GET /api/v2/resources/published/, 674	DELETE /api/v2/documents/, 653
GET /api/v2/resources/resource_types/, 674	DELETE /api/v2/documents/{id}/, 654
GET /api/v2/resources/{id}/, 672	DELETE /api/v2/geoapps/, 656
GET /api/v2/resources/{id}/get_perms/, 673	DELETE /api/v2/geoapps/{id}/, 657
GET /api/v2/schema/, 674	DELETE /api/v2/geostories/, 659
GET /api/v2/uploads/, 674	DELETE /api/v2/geostories/{id}/, 660
GET /api/v2/uploads/{id}/, 675	DELETE /api/v2/groups/, 662
	DELETE /api/v2/groups/{id}/, 664
	DELETE /api/v2/layers/, 665
	DELETE /api/v2/layers/{id}/, 667
	DELETE /api/v2/maps/, 668
	DELETE /api/v2/maps/{id}/, 670
	DELETE /api/v2/resources/, 671

DELETE /api/v2/resources/{id}/, 673
DELETE /api/v2/uploads/, 675
DELETE /api/v2/uploads/{id}/, 677
DELETE /api/v2/users/, 679
DELETE /api/v2/users/{id}/, 681
PATCH /api/v2/documents/{id}/, 654
PATCH /api/v2/geoapps/, 655
PATCH /api/v2/geoapps/{id}/, 657
PATCH /api/v2/geostories/, 659
PATCH /api/v2/geostories/{id}/, 660
PATCH /api/v2/groups/, 662
PATCH /api/v2/groups/{id}/, 663
PATCH /api/v2/layers/, 665
PATCH /api/v2/layers/{id}/, 667
PATCH /api/v2/maps/, 668
PATCH /api/v2/maps/{id}/, 670
PATCH /api/v2/resources/, 671
PATCH /api/v2/resources/{id}/, 673
PATCH /api/v2/uploads/, 675
PATCH /api/v2/uploads/{id}/, 677

/mapstore

GET /mapstore/rest/resources/, 682
GET /mapstore/rest/resources/{id}/, 682
GET /mapstore/rest/users/, 682
GET /mapstore/rest/users/{id}/, 683
POST /mapstore/rest/resources/, 682
POST /mapstore/rest/users/, 682
PUT /mapstore/rest/resources/{id}/, 682
PUT /mapstore/rest/users/{id}/, 683
DELETE /mapstore/rest/resources/{id}/,
682
DELETE /mapstore/rest/users/{id}/, 683
PATCH /mapstore/rest/resources/{id}/,
682

/o

GET /o/userinfo/, 683

/upload

PUT /upload/final?id={import_id}, 678