

---

# **django-terra-geocrud**

***Release 0.3.46***

**Sep 15, 2020**



---

## Contents

---

<b>1</b>	<b>Installation</b>	<b>3</b>
1.1	Requirements . . . . .	3
1.2	With pip . . . . .	3
1.3	With git . . . . .	3
<b>2</b>	<b>Configuration</b>	<b>5</b>
<b>3</b>	<b>Example of use</b>	<b>9</b>



Backend API configurator for Geographic CRUD. Based on django-geostore



### 1.1 Requirements

Should match with [django-geostore](<https://django-geostore.readthedocs.io/en/latest/installation.html#requirements>) requirements

### 1.2 With pip

From Pypi:

```
pip install django-terra-geocrud
```

From Github:

```
pip install -e https://github.com/Terralego/django-terra-geocrud.git@master  
↪ #egg=django-terra-geocrud
```

### 1.3 With git

```
git clone https://github.com/Terralego/django-terra-geocrud.git  
cd django_terra_geocrud  
python setup.py install
```





## CHAPTER 2

---

### Configuration

---

In your project :

- settings

```
INSTALLED_APPS = [
    ...
    # apps required by CRUD
    'geostore', # store geographic data
    'template_model', # store template in model
    'template_engines', # generate odt and docx templates
    'rest_framework', # if you want to try api HTML interface
    'django_json_widget', # if you want to use django admin
    'reversion', # used to store every change on data (run ./manage.py_
    ↪ createinitialrevisions first)
    # CRUD app
    'terra_geocrud',
    ...
]
...
TEMPLATES = [
    ...
    # if you want to render odt templates
    {'BACKEND': 'template_engines.backends.odt.OdtEngine'},
    # if you want to render docx templates
    {'BACKEND': 'template_engines.backends.docx.DocxEngine'},
]
```

- urls

```
urlpatterns = [
    ...
    path('api/mapbox-baselayer/', include('mapbox_baselayer.urls')),
    path('api/crud/', include('terra_geocrud.urls')),
    ...
]
```

You can customize default url and namespace by including `terra_geocrud.views` directly

Run migrations

```
./manage.py migrate
```

- ADMIN :

you can disable and / or customize admin

```
:: from django.contrib import admin from geostore.models import Layer, Feature from mapbox_baselayer.admin import MapBaseLayerAdmin from mapbox_baselayer.models import MapBaseLayer
```

```
from terra_geocrud import admin as geocrud_admin, models
```

```
admin.site.register(models.CrudGroupView, geocrud_admin.CrudGroupViewAdmin) admin.site.register(models.CrudView, geocrud_admin.CrudViewAdmin) admin.site.register(models.AttachmentCategory, geocrud_admin.AttachmentCategoryAdmin)
```

```
# we recommend to activate MapBaseLayerAdmin to manage map base layers admin.site.register(MapBaseLayer, MapBaseLayerAdmin)
```

```
# we recommend to use this admin for this geostore models admin.site.register(Layer, geocrud_admin.CrudLayerAdmin) admin.site.register(Feature, geocrud_admin.CrudFeatureAdmin)
```

- SETTINGS :

Waiting for settings definition directly in models.

Settings should be overridden with `TERRA_GEOCRUD` settings in your project settings file:

```
...
TERRA_GEOCRUD = {
    # default value for map extent. API serialize this for layer extent if there is
    ↪no features in it (as default)
    'EXTENT': [-90.0, -180.0, 90.0, 180.0],
    # default storage for file stored in json properties. It is recommended to
    ↪configure a private web storage in your project (as S3Storage -> see django-
    ↪storages)
    'DATA_FILE_STORAGE_CLASS': 'django.core.files.storage.FileSystemStorage',
    # default mapbox style provided by api if no custom style defined in crud view
    'STYLES': {
        'line': {
            'type': 'line',
            'paint': {
                'line-color': '#000',
                'line-width': 3
            }
        },
        'point': {
            'type': 'circle',
            'paint': {
                'circle-color': '#000',
                'circle-radius': 8
            }
        },
        'polygon': {
            'type': 'fill',
            'paint': {
                'fill-color': '#000'
            }
        }
    }
}
```

(continues on next page)

(continued from previous page)

```
        },  
    }  
}  
...  

```

- If you want to generate map on your template with the geometry of your feature, and/or extra features, you should use mbglrenderer.

Check <https://github.com/consbio/mbgl-renderer>.

Change the url in the settings to use your instance of mbglrenderer :

```
MBGLRENDERER_URL = 'http://mbglrenderer'
```



## CHAPTER 3

---

### Example of use

---

- django-terra-gecorud provide its own settings url

/api/crud/settings/

- There are 4 endpoint in GEOCRUD API:

settings/	-> get ordered menu <b>with</b> views classified by group <b>or</b>
↳ <b>not</b> , <b>and</b> basic <b>map</b> settings	
groups/	-> manage groups of CRUD views
views/	-> manage CRUD views (a view creation create its
↳ associated layer)	

- A command is available to create default views for each existing layer

```
./manage.py create_default_crud_views
```

- START GUIDE
- First, you need to create crud views for your geostore layers with the command or the admin.
- These views can be grouped, and will be listed by the frontend api
- Then, you can customize default layer-schema by providing your own property groups, which will groups properties as json schema nested objects.

#### ## ADMIN

- Some classes are provided to help you to manage Crud views / groups / layers and feature through django admin.
- You need to register your wanted ModelAdmin in your project

#### ## TEMPLATES

- Check <https://github.com/Terralego/django-template-engines> to create your own template.
- In addition to the ODTEngine and DocXEngine, for odt only, you can add maps of layers with features and extra\_features. Use :

```
{% load map_tags %}
{% map_image_url_loader feature_included=True extra_features="Extra_
↪feature_slug,Extra_feature_2_slug"
  base_layer="mapbox_baselayer_slug" %}
```

You can use the other tags : width, height, anchor.