# Technical test Django/React :

The context of this test is to provide a simple web service for storing and retrieving mountain peaks.

Using the python web framework Django and a postgresql database(postGIS can be used for geo features), implement the following features:

- models/db tables for storing a peak location and attribute: lat, lon, altitude, name

- REST api endpoints to :

    \* create/read/update/delete a peak

    \* retrieve a list of peaks in a given geographical bounding box

Then using React, implement the following features:

- Display some mountain locations on a map

- On click on a mountain location, display mountain attributes: lat, lon, altitude, name

- Allow users to add manually a mountain peak directly on the map

Deploy all this stack using docker and docker-compose

The source code should be delivered using github with detailed explanations on how to deploy and launch the project.

NOTES PROJET MFI

Tuto pour creation d’une api REST CRUD avec Django djangoframework

<https://dev.to/jkaylight/django-rest-framework-with-postgresql-a-crud-tutorial-1l34>

docker-compose run web python manage.py makemigrations

docker-compose run web python manage.py migrate

docker-compose up –build

3 services tournent:

* Restapimountainpeaks\_db\_1 qui correspond à la bdd postgresql
* Restapimountainpeaks\_react

# MFI Test - Mountain Peaks

PREREQUISITES: To have docker installed on your computer

This project is about a map displaying some mountain peaks on given coordinates.

## Features

- You can add new mountain peak by clicking on the \_\_add data\_\_ button

- You can update a mountain peak by clicking on the \_\_point\_\_ in the map

## To easily run your project

First you have to clone this repository on your computer by typing

```

git clone https://github.com/Fantemis/mfi\_test.git

```

Then move on this folder:

```

cd mfi\_test

```

Next step is to move on the folder named: \_\_back\_\_ .

```

cd back

```

You'll now find a file named: \_\_docker-compose.yml

This file allows you to get the entire project environment, including:

- postgresql database

- django rest api

- react web app (with 3D globe)

Then you'll have to type this commands:

```

docker-compose run web python manage.py makemigrations

```

Then

```

docker-compose run web python manage.py migrate

```

And then you can finish by:

```

docker-compose up --build

```

This flag is to build your environment, to download all dependencies etc.

I am personally using docker desktop on windows10.

So when your containers are built, you can find them on the docker desktop interface or by typing

```

docker container ls

```

You'll find 3 containers:

- react\_web\_app

- psql\_db

- django\_rest\_api

You can now navigate on your applications

- \_\_localhost:3000\_\_ for the react web app

- \_\_localhost:8000/mountain\_peak\_\_ for the django api (with default user web interface)

Note that you have differents api routes availables

- CREATE: \_\_localhost:8000/mountain\_peak/create/\_\_ : Allows you to create your own mountain peak

- READ: \_\_localhost:8000/mountain\_peak\_\_ : Its the default route, it will give you all mountain peaks

- UPDATE: \_\_localhost:8000/mountain\_peak/update/particular-id\_\_: To update particular mountain peak

- DELETE: \_\_localhost:8000/mountain\_peak/delete/particular-id\_\_: To delete particular mountain peak