Covid19 Data App

Table of Contents

Project Overview

The project is a simple web application which retrieves covid 19 information from open source rapidAPI endpoint. The data is fetched based on a unique country code provided to each of the countries and populated into the database. This is a one-time activity that the user must do to perform actions on the local database setup in the project. For example, below is a list of countries and their country codes

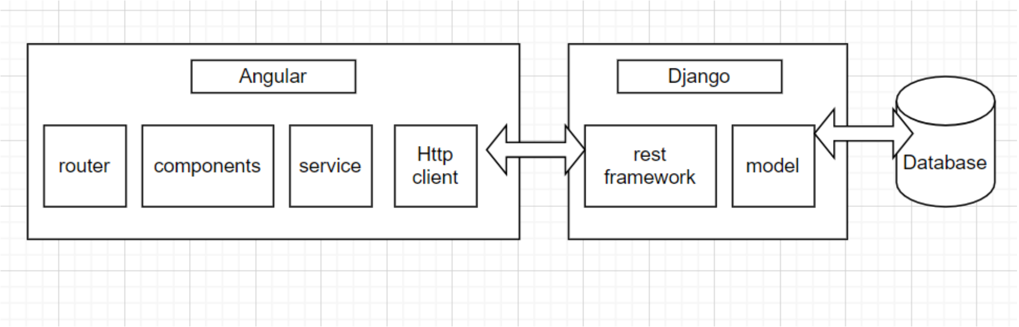
- Ireland: ie
- Germany: de
- Italy: it
- UK: gb
- Spain: es

Sample of the data received from rapiAPI is as below and the database table is designed based of this

```
{
        "country": "Ireland",
        "code": "IE",
        "confirmed": 480846,
        "recovered": 401907,
        "critical": 89,
        "deaths": 5566,
        "latitude": 53.41291,
        "longitude": -8.24389,
        "lastChange": "2021-11-11T15:49:04+01:00",
        "lastUpdate": "2021-11-11T17:15:03+01:00"
}
```

Architectural Diagram:

Django exports REST Apis using Django Rest Framework & interacts with Database using Django Model. Angular Client sends HTTP Requests and retrieve HTTP Responses using HttpClient Module, shows data on the components. We also use Angular Router for navigating to pages.



Overall Architecture

Technology Used

The web application is built using the latest version of angular and Django rest framework. A simple restful backend is setup to serve API endpoint requests. The stack was chosen since Django rest framework is battery powered and has a lot of features built in which does the heavy lifting for us, angular was chosen since there are a limited number of components involved and is an efficient framework for single page applications. Postgres database has been used to store the data. Git has been used for version control .

The complete code for the project can be found in GitHub at :
https://github.com/nkrishx/openAppRepo

Below are the links to set up these technologies.

1. https://www.django-rest-framework.org/
2. https://www.postgresql.org/download/
3. https://angular.io/guide/setup-local

Database Design

Based on the requirements and the data retrieved from rapidAPI endpoint, a database with a single table has been designed and used. If required download and install pgAdmin tool as well which is a GUI for interacting with postgres database.

The fields and of the database are:

country:char
code : char
confirmed: positive int
recovered: positive int
critical: positive int
deaths: positive int
latitude: float
longitude: float
lastChange: dateTime
lastUpdate: dateTime
created: dateTime
updated: dateTime

Once postgres is installed on the local machine, we need to set up a database for the application to interact with, this can be done by going into the postgres shell and running the create database command.

The shell can be accessed by tying ‘**psql**’ in the terminal if added as system environment variable and then create the database using

CREATE DATABASE database name OWNER username

The database name mentioned in the above command is important as this will be the same name used in the settings.py file of the Django project.

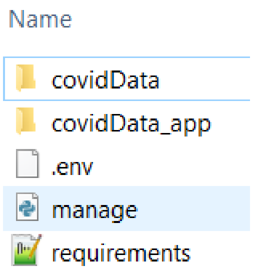
https://www.guru99.com/postgresql-create-database.html

Local Setup

- Download and install python 3 and above since the version of Django used here is the latest stable version 3.2.9.
- First setup a virtual environment for the project if you do not want to mess up your global packages.

```
python -m venv /path/to/new/virtual/environment
```

- Once the venv is setup, activate it and clone the repository from the github link at the top.
- A requirements.txt file exists stating all of the packages required.
- Run the requirements.txt with pip install requirements.txt
- Once this is done successfully you need to setup the database information and the keys used in the project.
- Create a file called as .env in the same directory as the Django project, add into the file the following info:



Name
covidData
covidData_app
.env
manage
requirements

```
SECRET_KEY=django-insecure-avy=$06uhy!(rf01bas0r9banpy8(5vx5lspt6q)f_sari26b*

DB_NAME=database name

DB_USER=user

DB_PASSWORD=password

DB_HOST=localhost

RAPID_API_KEY=3ad19a4761msh51ac800fb2e2c5ap159813jsne62cfb3dca3c
```

RAPID_API_HOST=covid-19-data.p.rapidapi.com

RAPID_API_QUERY_URL=https://covid-19-data.p.rapidapi.com/country/code

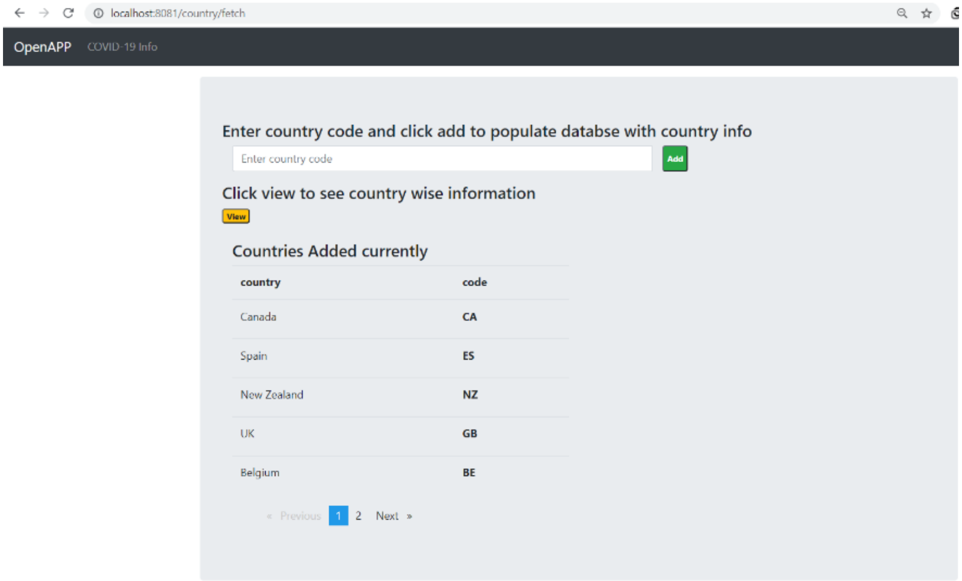
This step is important since we do not want our local passwords to be exposed, the .env file is added to the .gitignore file and will not be pushed to the repository.

*If you have a separate rapidAPI key please use that instead of the one provided above *use the same name and user as that when setting up the data base

- Next navigate to the frontend/covidData directory and run the command **npm install** to install all of the angular required packages.
- Once the installation of the packages is done, please run the Django server in a terminal/cmd window. Navigate to the directory that has the manage.py file and run **python manage.py runserver**, this will start the Django server to serve the api endpoints.
- Next in a separate terminal/cmd window run the angular server, navigate to frontend/covidData directory, and run **ng serve --port 8081**. The port 8081 is important since in the settings.py file of the Django project we have configured to allow requests to our application from other origins.
- Once both the servers are running you can navigate to http://localhost:8081/ to view the web app. Visit http://localhost:8080/ to view the browsable API interface of Django rest framework.

The overall idea of the app is that we do a onetime insertion of data into the database in the home/landing page of the application, this will fetch us data from rapidAPI and populate the db. Once populated we have the fine grain control of performing CRUD operations on the populated data. The creation of a new country is only possible by entering the country code for a country in the app home page. This is since we want to fetch data from the external rapidAPI endpoint and perform operations on top of the record after fetched and inserted. The updated and created fields of the data for a single country is the local operations timestamp we perform on the record.

Sample screen grab of the web app, follow the instructions on the screen to populate the db and interact with the countries that have been added.



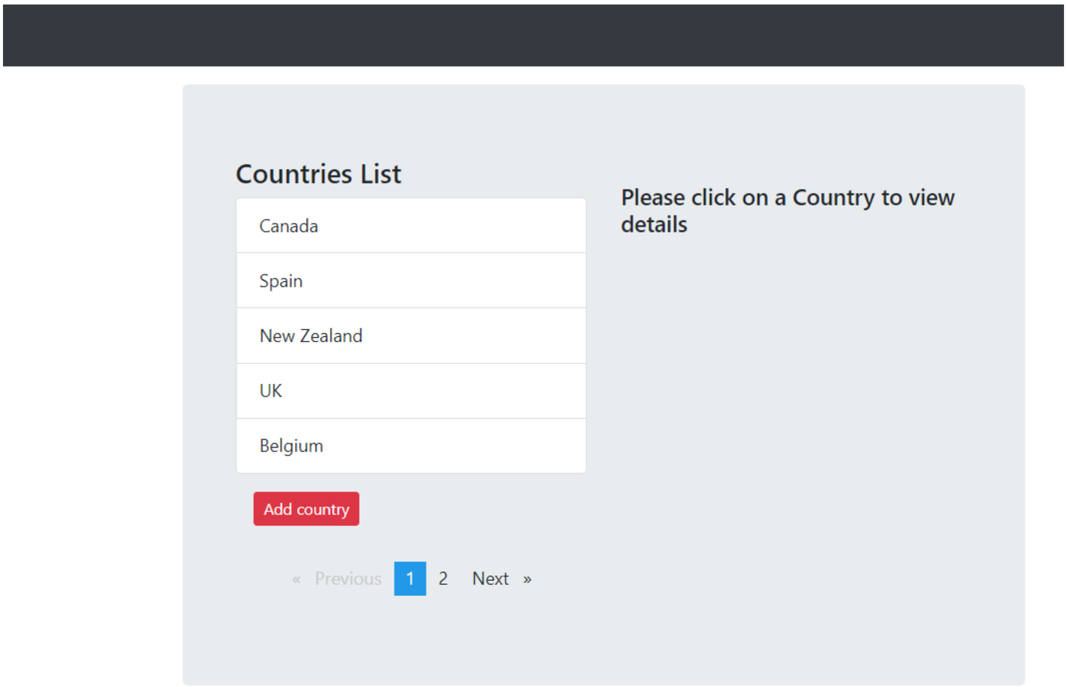
Enter country code and click add to populate databse with country info

Click view to see country wise information

Countries Added currently

country	code
Canada	CA
Spain	ES
New Zealand	NZ
UK	GB
Belgium	BE

Home Page

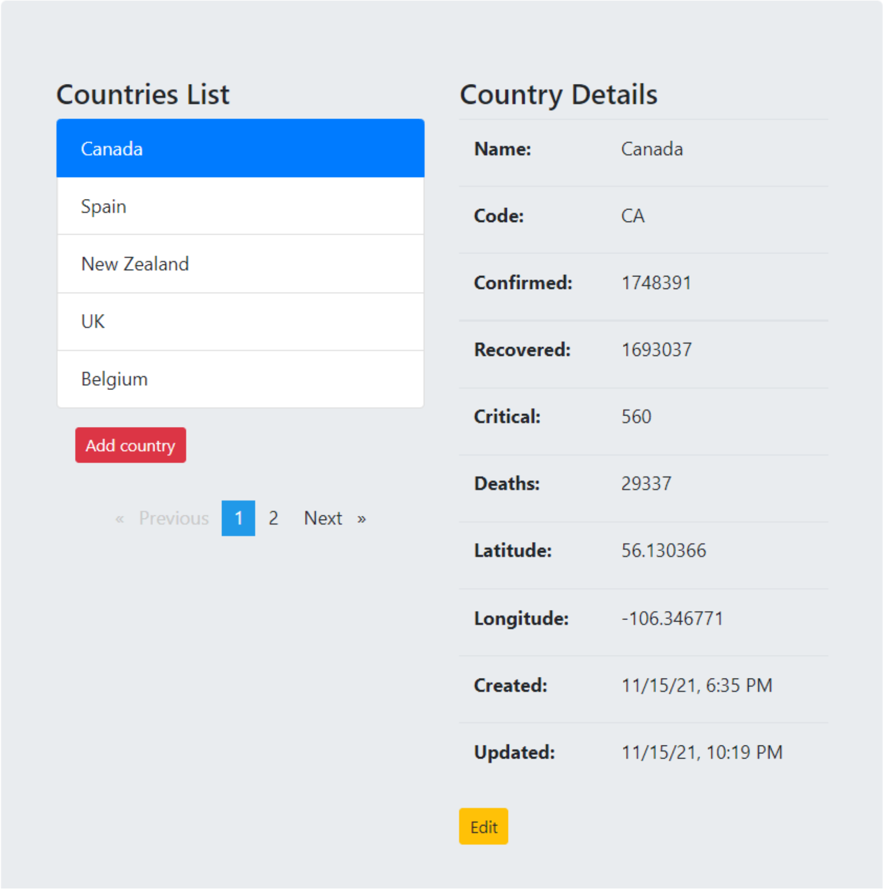


Countries List

- Canada
- Spain
- New Zealand
- UK
- Belgium

Please click on a Country to view details

List View



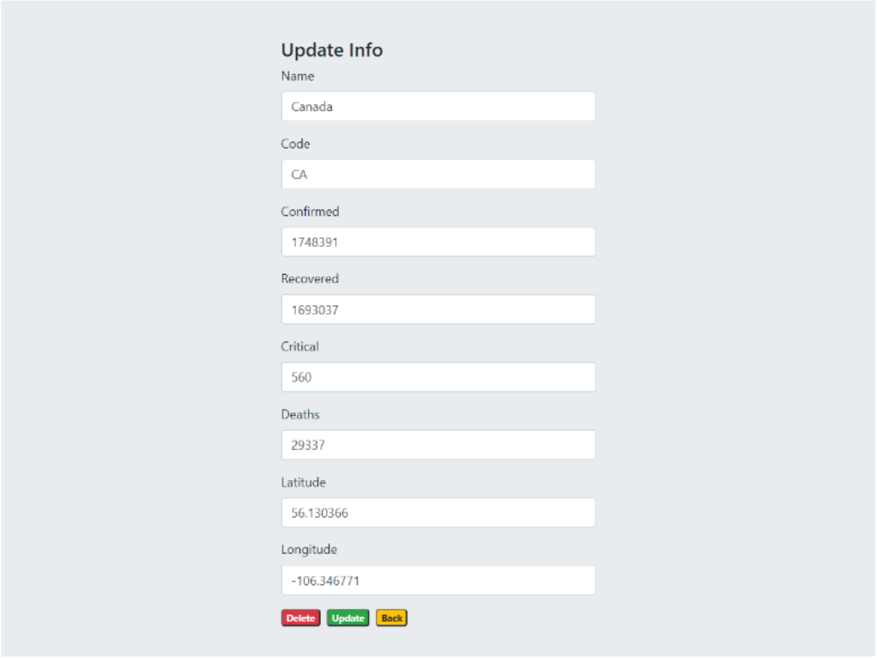
Countries List

- Canada
- Spain
- New Zealand
- UK
- Belgium

Country Details

Name:	Canada
Code:	CA
Confirmed:	1748391
Recovered:	1693037
Critical:	560
Deaths:	29337
Latitude:	56.130366
Longitude:	-106.346771
Created:	11/15/21, 6:35 PM
Updated:	11/15/21, 10:19 PM

Detail View



Update Info

Name: Canada

Code: CA

Confirmed: 1748391

Recovered: 1693037

Critical: 560

Deaths: 29337

Latitude: 56.130366

Longitude: -106.346771

Update/edit View

API Documentation

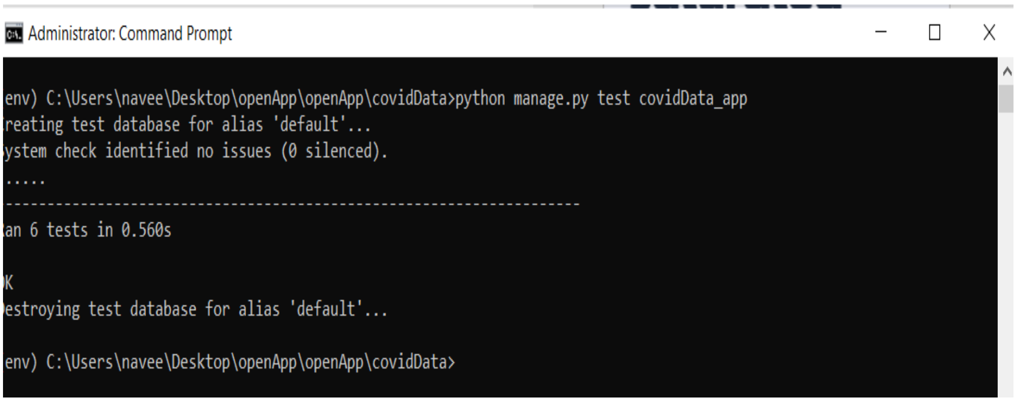
Methods	Urls	Actions
GET	/api/v1/country/data/	retrieve all countries
GET	/api/v1/country/data/:id	retrieve a country by :id
PUT	/api/v1/country/data/:id	update a country by :id
DELETE	/api/v1/country/data/:id	delete a country by :id
GET	/api/v1/country?code=[country_code]	Fetches data from rapidAPI for the country and adds to db
GET	/api/v1/country/	retrieve all countries with ordering, searching, pagination and filter capabilities

The /api/v1/country/ endpoint is something that is not used by the angular frontend, it’s an endpoint that can used in browsable API page of Django or postman app to see the various kind of ordering, searching, and filtering that is setup in the view

Unit Tests

Unit tests covering all of the above mentioned actions are written and is available to run the tests. It can be run by navigating to the directory with manage.py file and running the command “python manage.py test covidData_app”.

A postman app collection has also been setup in the project and can be used to test the endpoints for latency.



```
env) C:\Users\navee\Desktop\openApp\openApp\covidData>python manage.py test covidData_app
reating test database for alias 'default'...
ystem check identified no issues (0 silenced).
.....
---------------------------------------------------------------------
an 6 tests in 0.560s

K
estroying test database for alias 'default'...

env) C:\Users\navee\Desktop\openApp\openApp\covidData>
```

Unit Test