

**REPUBLIQUE DU CAMEROON**

**PAIX-Travail-Patrie**

**MINISTRE DE L’ENSEIGNEMENT SUPERIEUR**

**FACULTE D’INGINERIE**

**ET TECHGNOLOGIE**

**REPUBLIC OF CAMEROON**

**Peace-Work-Fatherland**

**MINISTER OF HIGHER EDUCATION**

**FACULTY OF ENGINEERING**

**AND TECHNOLOGY**

**Presented by:**

**BITO’O BI HIEGA II DYLAN FLORENT FE20A024**

**CHEZEM DONGMEZA MIGUEL FE20A026**

**DJIOTSA DJOUAKE CHRISTIAN DARYN FE20A029**

**DJOUKENG NOUGNING YANNICK IVAN FE20A030**

**MADADJO KUITCHE MONICK CYBELLE FE20A060**

**INSTRUCTOR**: **Dr. NKEMENI VALERY**

**JUNE 2023**

Table of Contents

**No table of contents entries found.**

**INTRODUCTION:**

To solve the issue of food wastage in our society, we proposed to build a food waste management mobile application. We came out with the requirements of our system, its UML design and its User interface Design. The application we are offering will have to deal with many types of data collected from the different users and the application should be able to store and process that data. Now to solve this issue we need to implement the system’s databases after designing it.

1. **TOOLS USED FOR THIS ASSIGNMENT**

To achieve our goal, we made use of the following tools:

* **Star UML** for the design.
* **Django and Django rest framework** for the implementation.
* **Postgresql** to implement Database backup.

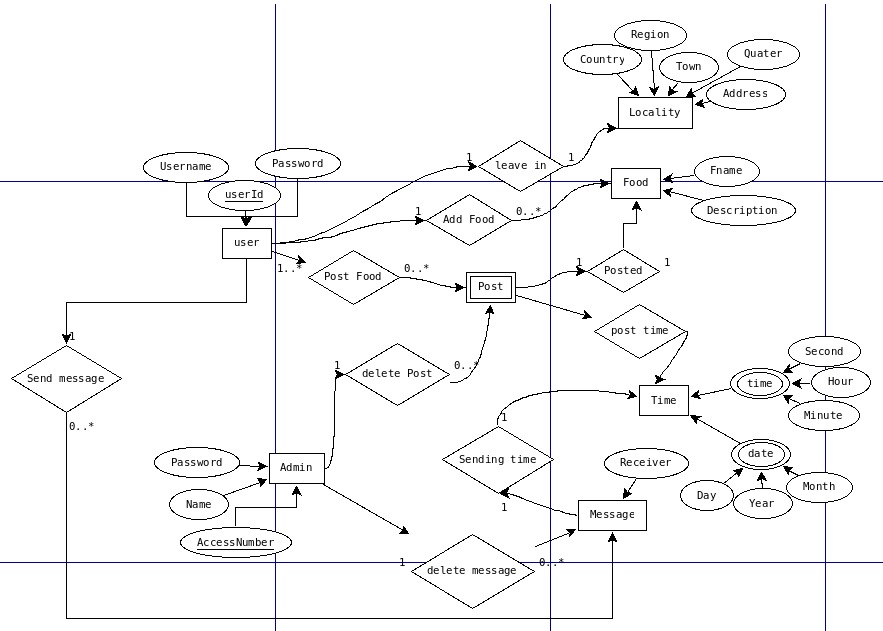
1. **DATABASE DESIGN:**

A **Database** isany collection of data, or information that is specially organized for rapid search and retrieval by a computer**.**  For the design of our database, we used:

* **The class Diagram**
* **The Entity Relational (ER) Diagram:**

1. **ENTITY RELATIONAL DIAGRAM:**

An **ER Diagram** is a type of flowchart that illustrates how **entities** such as people, objects or concepts relate to each other within a system. It is an essential element for database design.

Fig: ER diagram of **FoodsApp**

1. **CLASS DIAGRAM:**

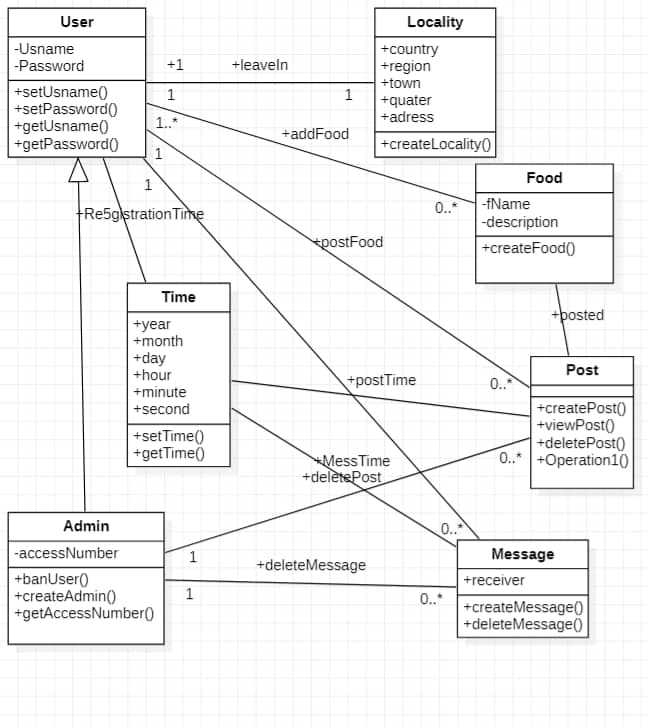


fig: **class Diagram of FoodsApp**

1. **DATABASE IMPLEMENTATION:**
2. **Creation of the Database.**

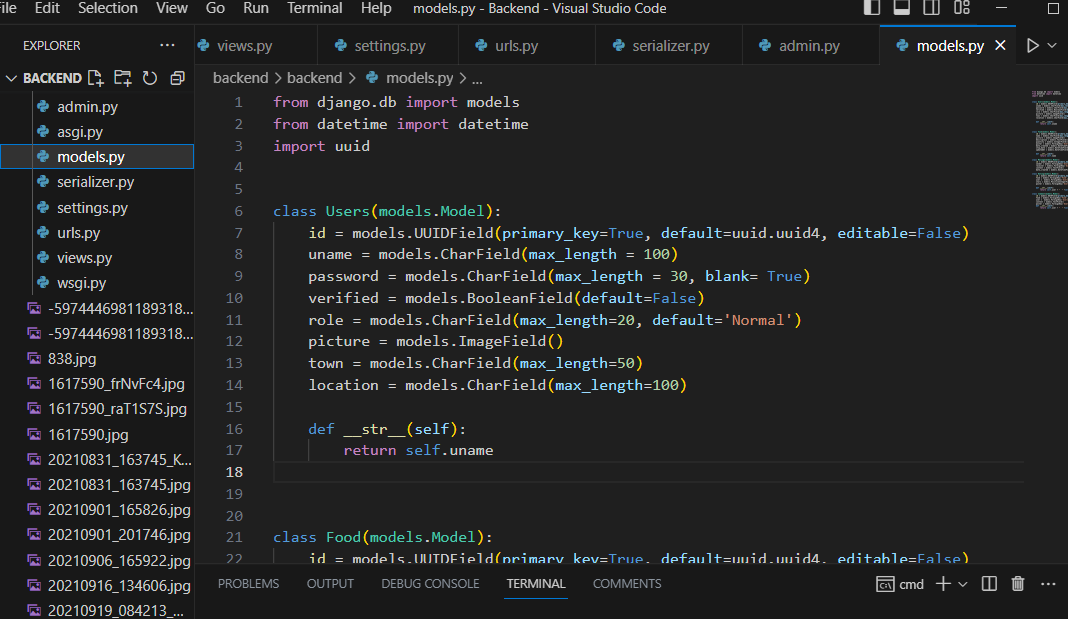
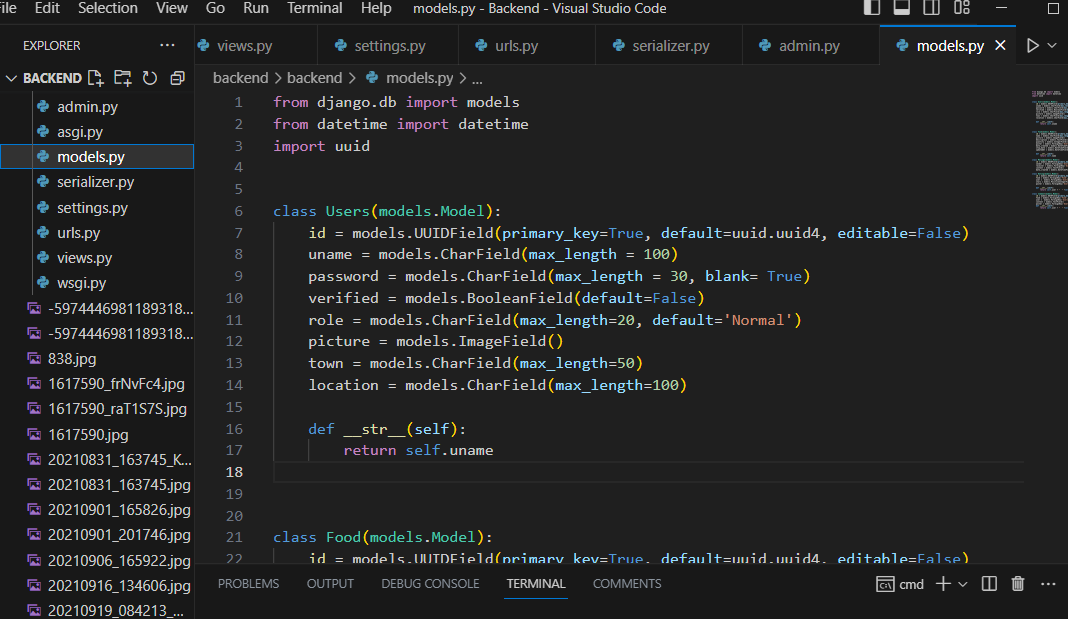
To create our Database, we used **Django** framework which is based on python. This is because Django eases backend implementation and deployment of big projects with a huge amount of libraries. A Django project is made up of multiple python files as shown in fig1 below

Fig1. **Django project files**

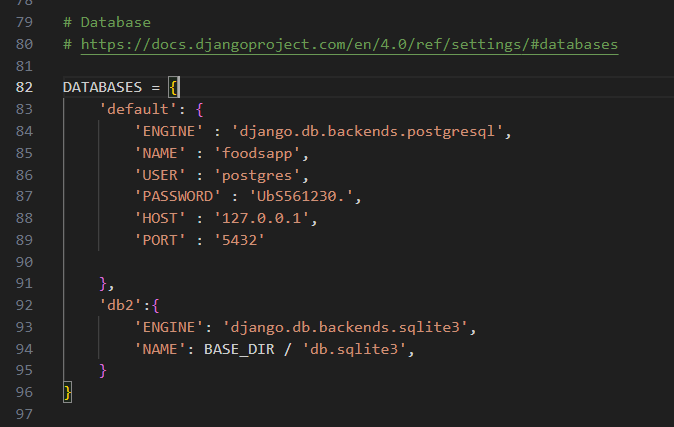
The data base creation is done in the models.py file where each table is considered as a python class and extends the django model class. Django will then convert this python file into SQL queries and create the database which will be stored in a file called db.sqlite.

In our case, tables were created in the following way

**Fig2. Database creation**

1. **Accessing the Database**

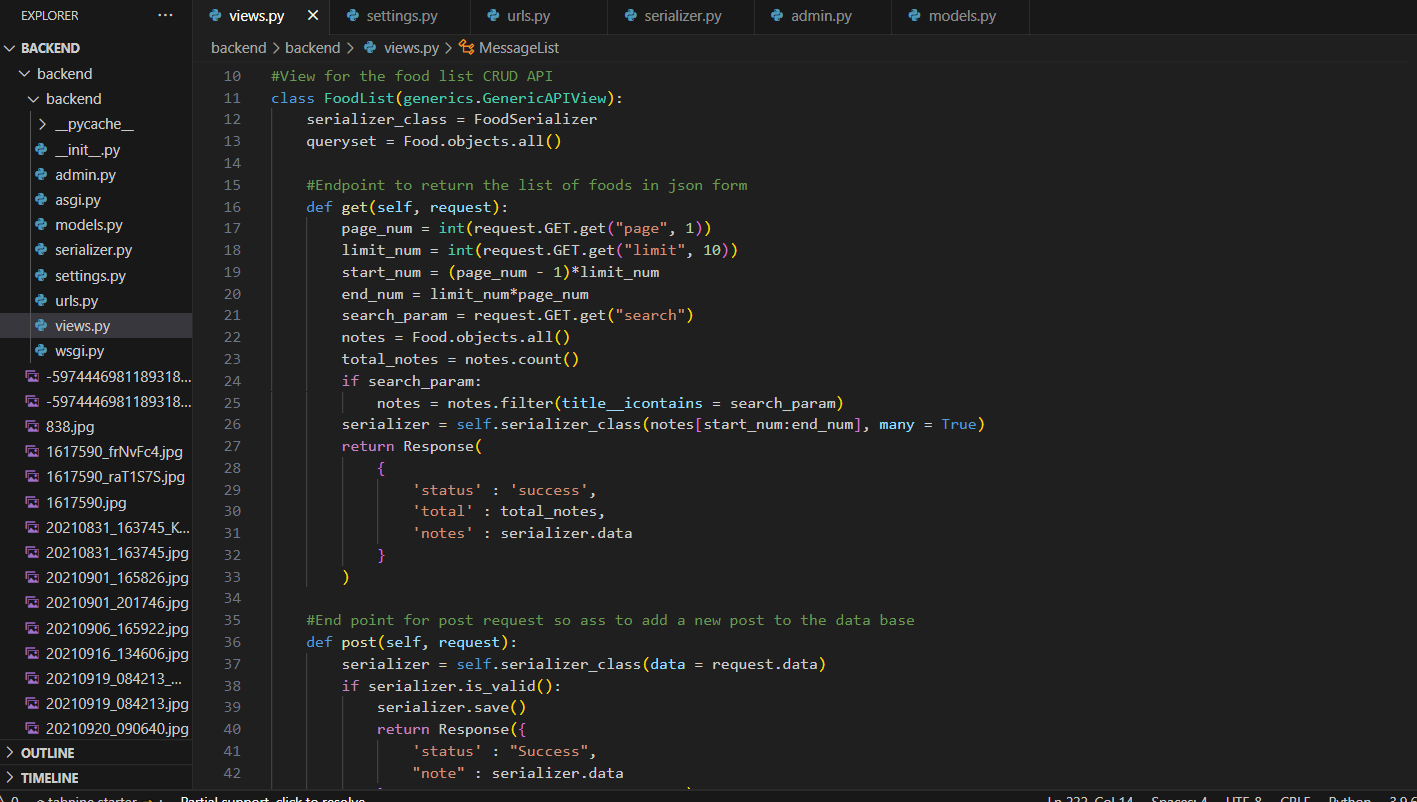
After haven created the database, it won’t serve if the application is not connected to it. In our work, we implemented a development schema where the frontend is completely separated from the backend. Therefore, we needed to create some API’s in other for the frontend to interact with our database.

To develop this APIs, we used Django Rest Framework and we connected this framework to our database as shown in fig3 below

**Fig3. Connection to database**

After the connection was established, the API development then started.

Django rest provides a default API view from which we can build our APIs by overwriting the methods of this view with our own specific service to be provided e.g. (For POST, GET, UPDATE, DELETE) requests.

This method can be shown below in Fig4

This APIs follow the REST (Representational State Transfer) architecture where every request and response is archived in the form of a JSON file before transiting from the client to the server and vice-versa. To convert our data to JSON, we used a Serializer class that will convert our database tables into a JSON file which can be manipulated by the frontend. One of the serializer class is shown in Fig5 below.

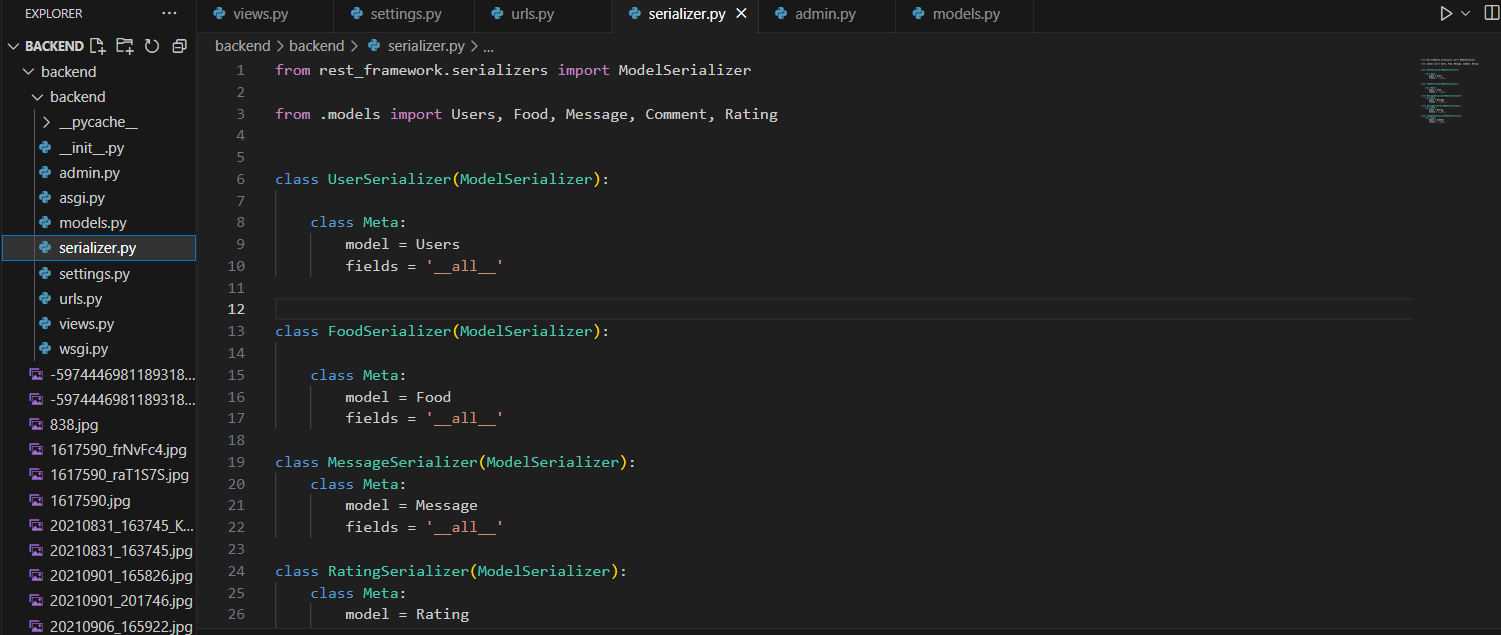
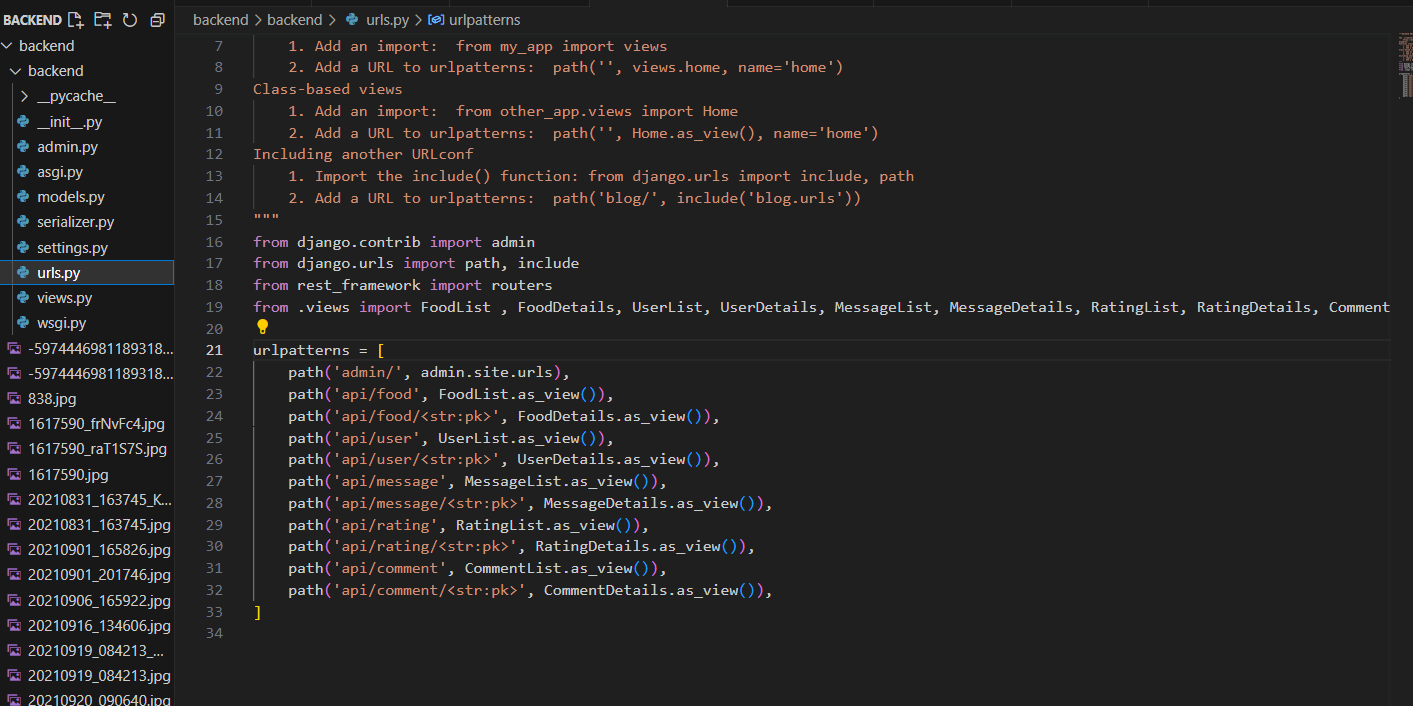
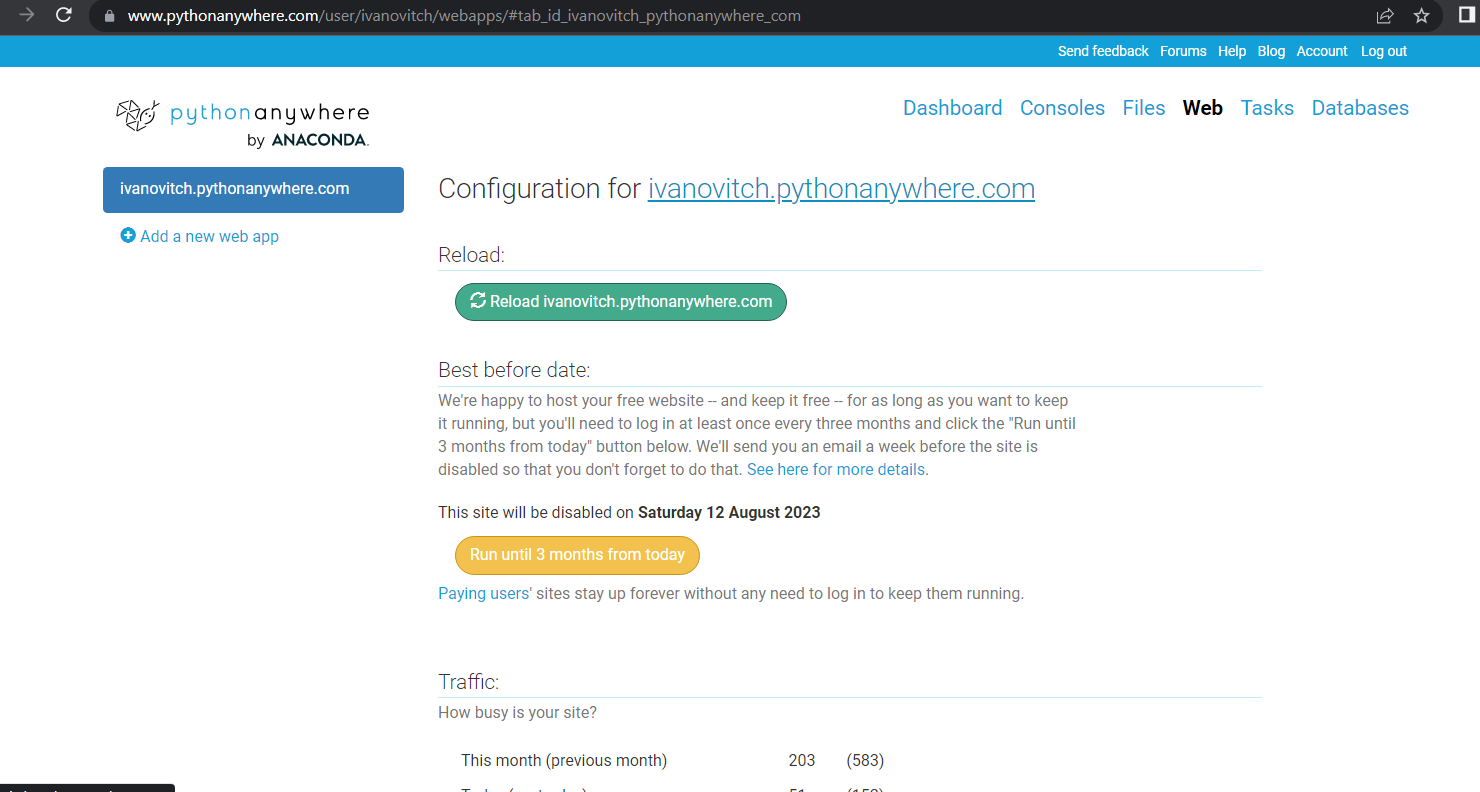


Fig5**. Serializer**

After this implantation the database was the fully accessible. In order to implement this access through the APIs, URLs where attached to each of this API-ENDPOINTS in other for the client to consume the available services. This was done as follows in the urls.py file

**Fig6. URLS connection**

1. **Backend Deployment**

The backend which is made up of the database, all the APIs and their Endpoints was deployed on https://www.pythonanywhere.com which is a hosting platform for python projects. The administrator view is as follows

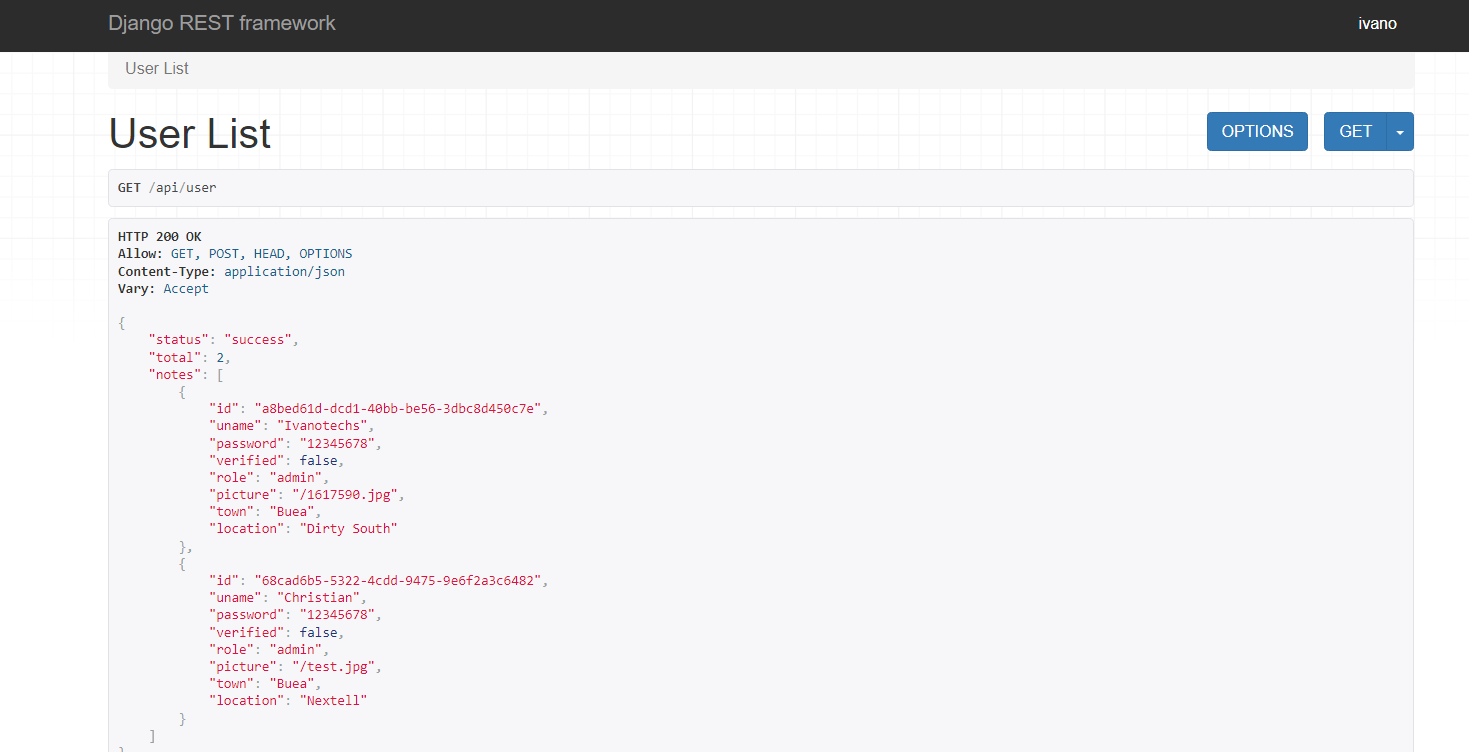
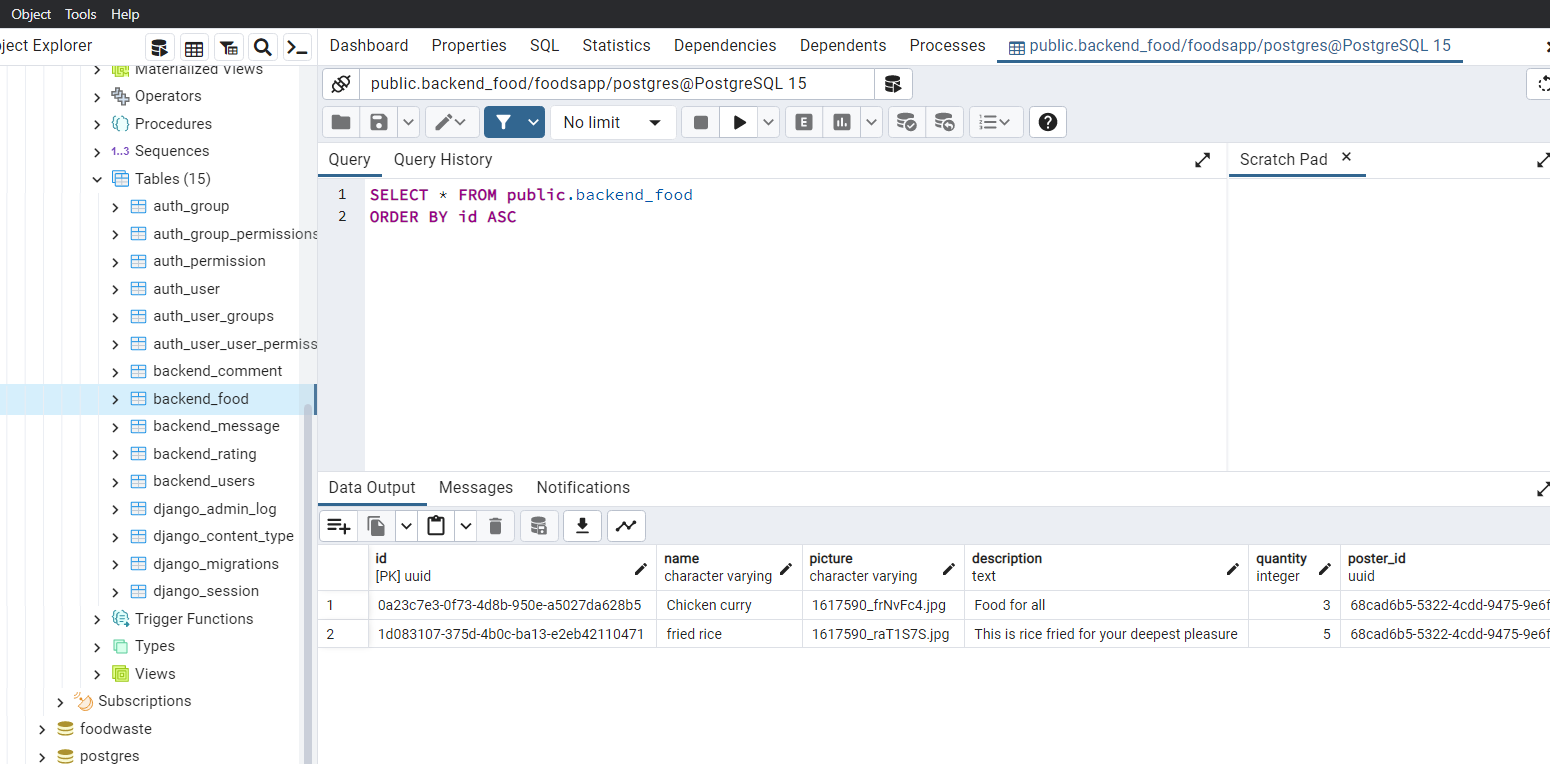
With our Django Rest APIView, we have a default panel to test our API endpoints before our frontend consume them. It can be seen below and available on the link [www.ivanovitch.pythonanywhere.com](http://www.ivanovitch.pythonanywhere.com/)

Fig7. User API Endpoint

1. **Database Backup**

In other to implement a backup for our database, a second database was implementing in Postgresql and connected to our project such that each query passes through the two databases. This is a protection so as to keep track of the original database for backup in case of failure. This is to implement reliability and data integrity. It can be seen below

**CONCLUSION:**

A database is a file saved on a server that can be accessed by clients, in other to implement one, we have to ensure

* Integrity : this was done by implementing a backup database
* Reliability : this was done by deploying the database online for 24/7 access and to ensure it won’t fail
* Accessibility: This was done by implementing APIs attaching URLs to the different endpoints.

Therefore we can conclude that our database was implemented successively and is secured for usage by any front-end device through easy REST APIs which can perform Create Read Update and Delete(CRUD) requests.