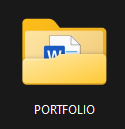
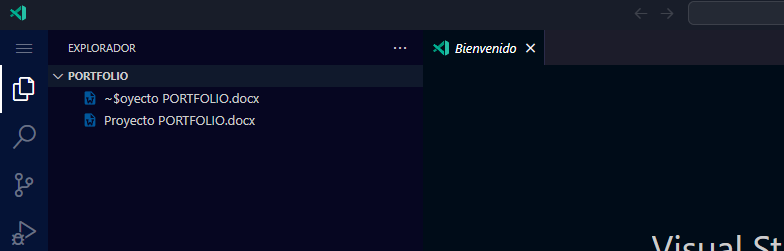
**DJANGO REST FRAMEWORK + REACT**

Crearemos un folder en el escritorio llamado PORTFOLIO 21/08/2023.



Luego abriremos el vscode para luego arrastrar el folder PORTFOLIO al vscode,



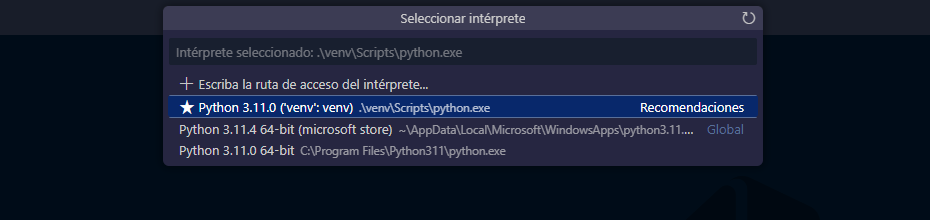
Dentro del vscode en una terminal ejecutaremos los siguientes comandos:

Instalaremos un entorno virtual si es la primera vez.

❯ pip install virtualenv

❯ python -m venv venv

Una vez creado el entorno virtual presionamos F1 en vscode, y escribimos Python interprete, para luego escoger el que tiene la estrella.



Una vez echo se activará el entorno virtual, como se puede ver en la consola.



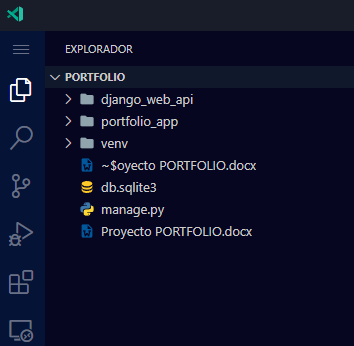
❯ pip install django

❯ django-admin startproject django\_web\_api .

❯ python manage.py runserver

❯ python manage.py startapp portfolio\_app

Una vez ejecutados los comandos anteriores debería quedar de la siguiente manera.



Ahora realizaremos configuraciones del backend, en el archivo setthings.py del folder django\_web\_api, donde incluiremos la aplicación portfolio\_app.



Luego ejecutamos los siguientes comandos.

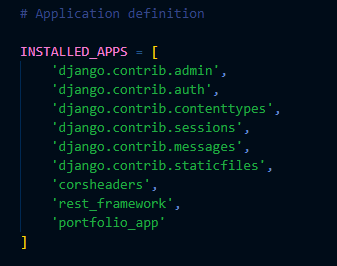
❯ python manage.py migrate

Ahora instalaremos el módulo django rest framework

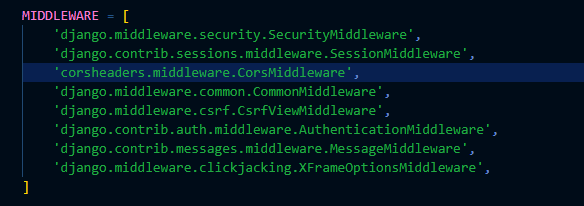
❯ pip install djangorestframework

❯ pip install django-cors-headers

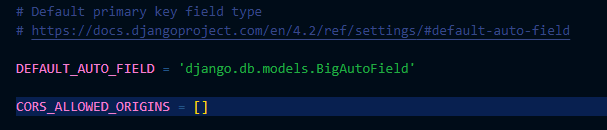
Añadimos los dos módulos instalados en el archivo setthings.py del folder django\_web\_api.



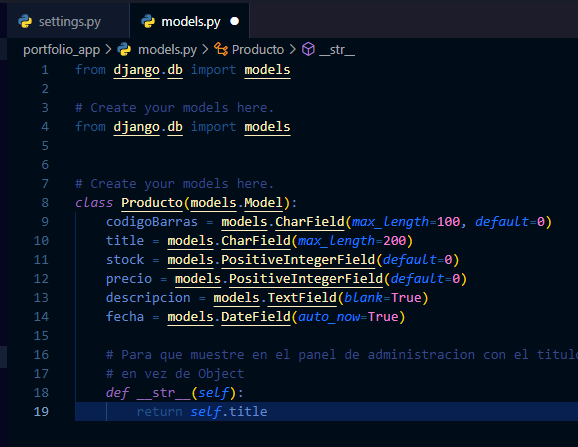
Añadimos el siguiente código en el mismo archivo, "corsheaders.middleware.CorsMiddleware",



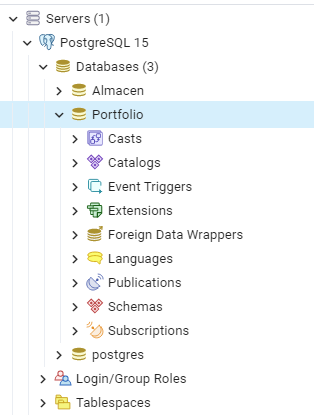
También lo siguiente en el mismo archivo.



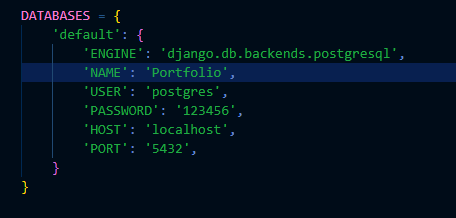
En el folder portfolio\_app, en el archivo models.py incluiremos nuestras clases de nuestro modelo que serán la representación de las tablas de nuestra base de datos.



Antes de realizar los modelos configuraremos en settings.py la conexión de nuestra base de datos con Postgresql, antes se debe crear la base de datos en Postgresql.



Ahora en el archivo setthings.py, incluir la conexión.



Ahora ejecutamos los siguientes comandos.

❯ pip install psycopg2

❯ python manage.py makemigrations

❯ python manage.py migrate

❯ python manage.py makemigrations portfolio\_app

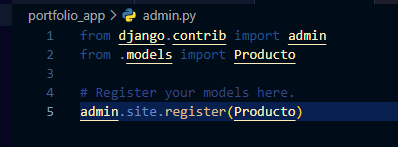
❯ python manage.py migrate portfolio\_app

Crearemos un usuario y su respectiva contraseña para el panel de administración de django.

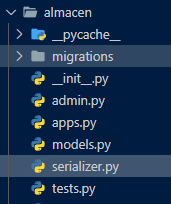
❯ python manage.py createsuperuser

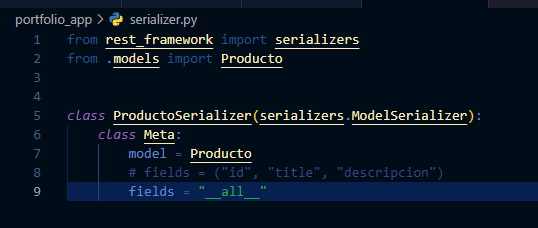
❯ python manage.py runserver

Ahora en folder portfolio\_app en el archivo admin.py añadiremos la tabla Producto para que se añada al panel de administración.

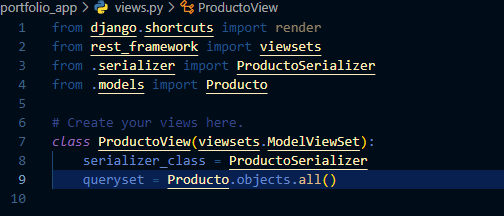


Crearemos un archivo llamado serializer.py en el folder portfolio\_app, para serializar la información en formato json.

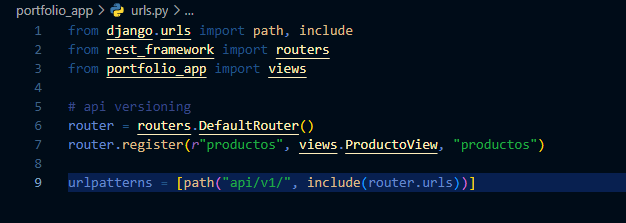




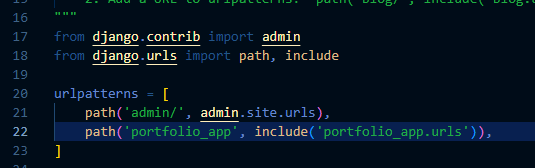
En el archivo views.py estarán todas las vistas o el CRUD de vistas.



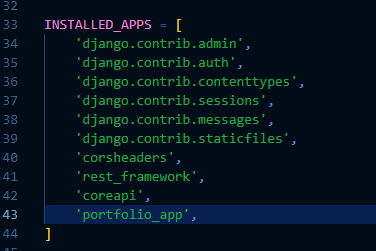
Crearemos el archivo de rutas llamado urls.py en el folder portfolio\_app.



En el folder django\_web\_api en el archivo urls.py debemos añadir los siguiente.

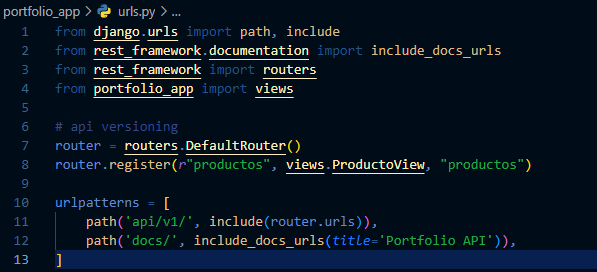


Para poder documentar nuestro web\_api, instalamos el siguiente modulo, en el archivo settings.py del folder django\_web\_api.

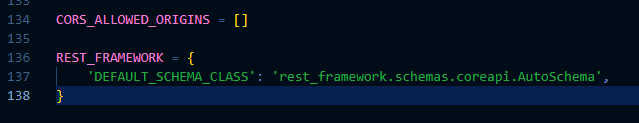


❯ pip install coreapi

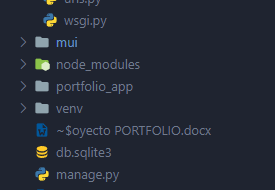
Ahora configuramos en el archivo urls.py del folder portfolio\_app.



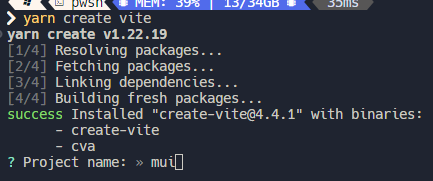
Archivo settings.py del folder django\_web\_api.

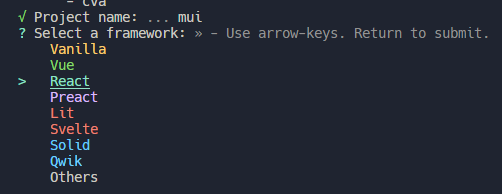


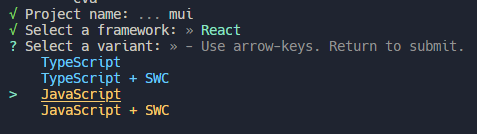
Ahora crearemos el frondend de nuestra aplicación en react, con la aplicación vitejs, le daremos un nombre llamado mui de material ui.



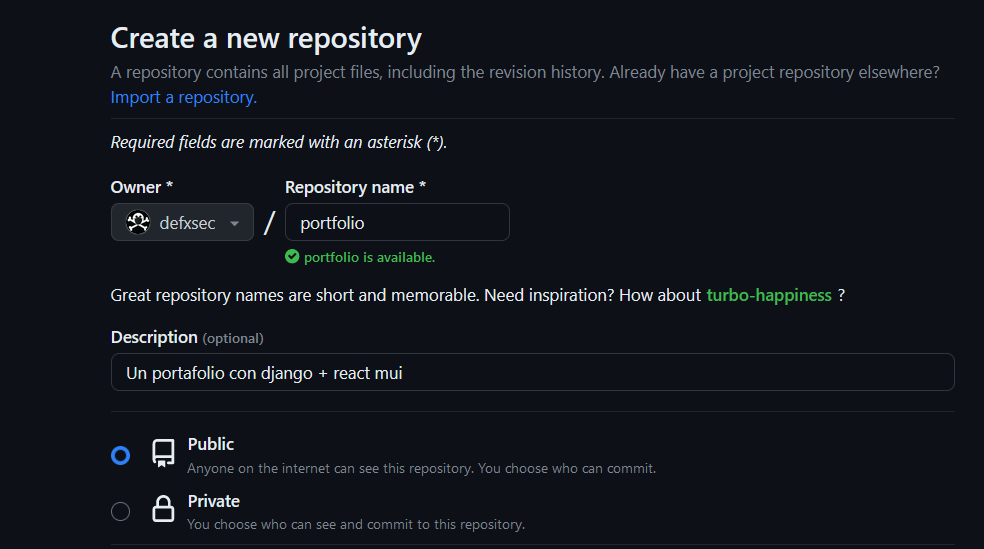
❯ yarn create vite







Ahora crearemos un repositorio en github.



❯ git init

❯ git add .

❯ git commit -m "first commit"

❯ git remote add origin https://github.com/defxsec/portfolio.git

❯ git push -u origin master

**Para un segundo commit**

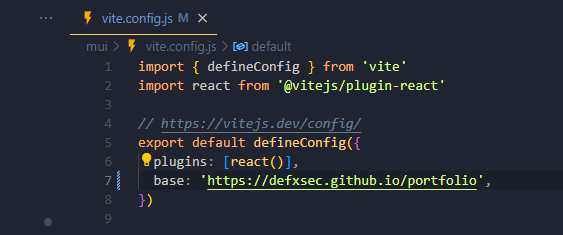
❯ git status

❯ git add .

❯ git commit -m "proyecto mui react"

❯ git push

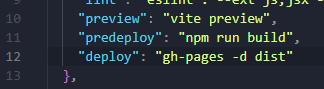
Ahora procedemos a configurar nuestro proyecto, en el archivo vite.config.js del folder mui.



Instamos la librería

❯ npm i --save-dev gh-pages

Ahora nos vamos al archivo package.json de mui.



❯ npm run deploy

Procederemos a instalar los demás paquetes.

❯ cd mui

❯ npm install

❯ npm run dev

❯ yarn add @mui/material @emotion/react @emotion/styled

❯ yarn add @mui/material @mui/styled-engine-sc styled-components

❯ yarn add @fontsource/roboto

❯ yarn add @mui/icons-material

❯ yarn add install react-router-dom react-hot-toast axios react-hook-form

❯ yarn add @mui/styles

❯ npm install --legacy-peer-deps

❯ npm config set legacy-peer-deps true

❯ npm i @mui/material @emotion/react @emotion/styled @mui/x-data-grid @mui/icons-material react-router-dom@6 react-pro-sideba

r formik yup @fullcalendar/core @fullcalendar/daygrid @fullcalendar/timegrid @fullcalendar/list @nivo/core @nivo/pie @nivo/lin

e @nivo/bar @nivo/geo

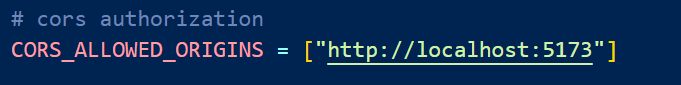
❯ npm i react-pro-sidebar@0.7.1

❯ yarn add react-router-dom

❯ npm install --save @fullcalendar/react @fullcalendar/core @fullcalendar/daygrid

❯ npm install @fullcalendar/interaction

Para permitir que se comuniue entre el front-end y back-end, debemos añadir la siguiente línea entre archivo settings.py de django\_web\_api.



❯

❯