

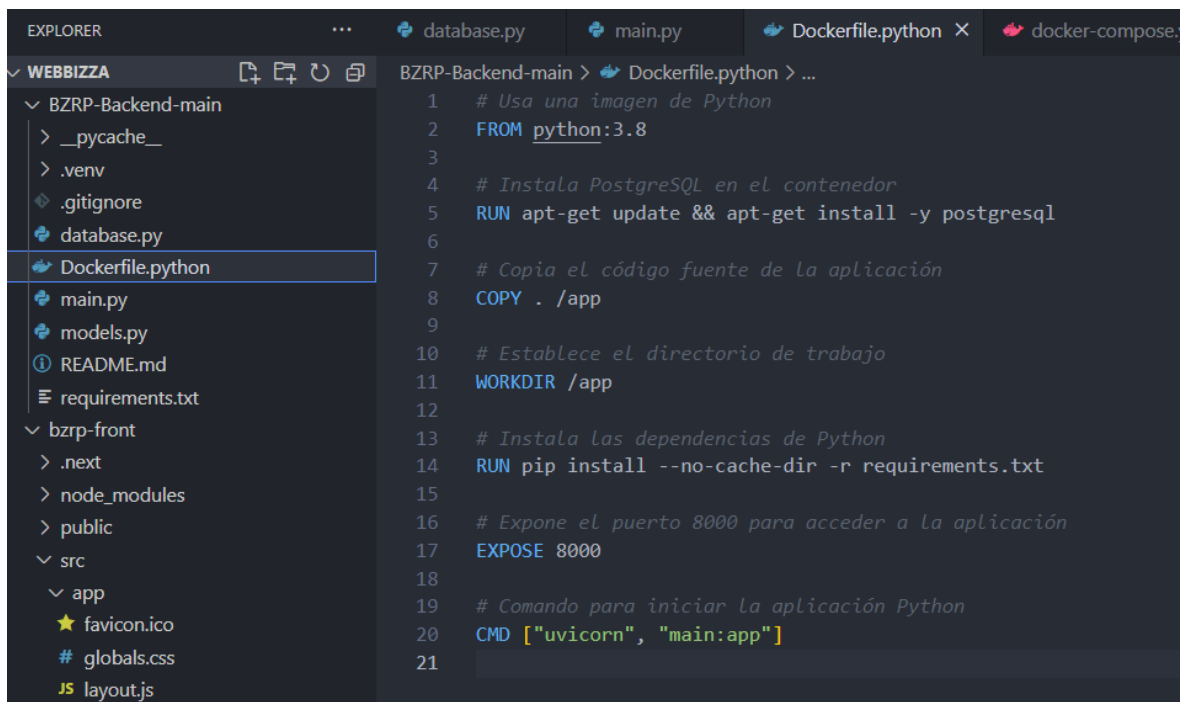
CARRERA DE INGENIERIA DE SOFTWARE

MATERIA: Gráficos y visualización

NOMBRES Y APELLIDOS: Christian Saraguro y Antonio Briones

FECHA: 11/07/2023

- Bueno para empezar ya tenía el proyecto de bizza rap funcional, con el backend y frontend funcionando.
- Luego de eso descargue el repositorio que nos dejó y hice una sola carpeta que contenía el backend y frontend.
Ahí agregué los archivos que estaban en su repo, en este caso los de Dockerfile para el respectivo back, front y la raíz de proyecto.
- Empecé instalando los requerimientos en el backend y modificando un poco el archivo pasado según la documentación que leí en internet para que funcionara en Windows y agregando el tema de la instalación de postgresql.



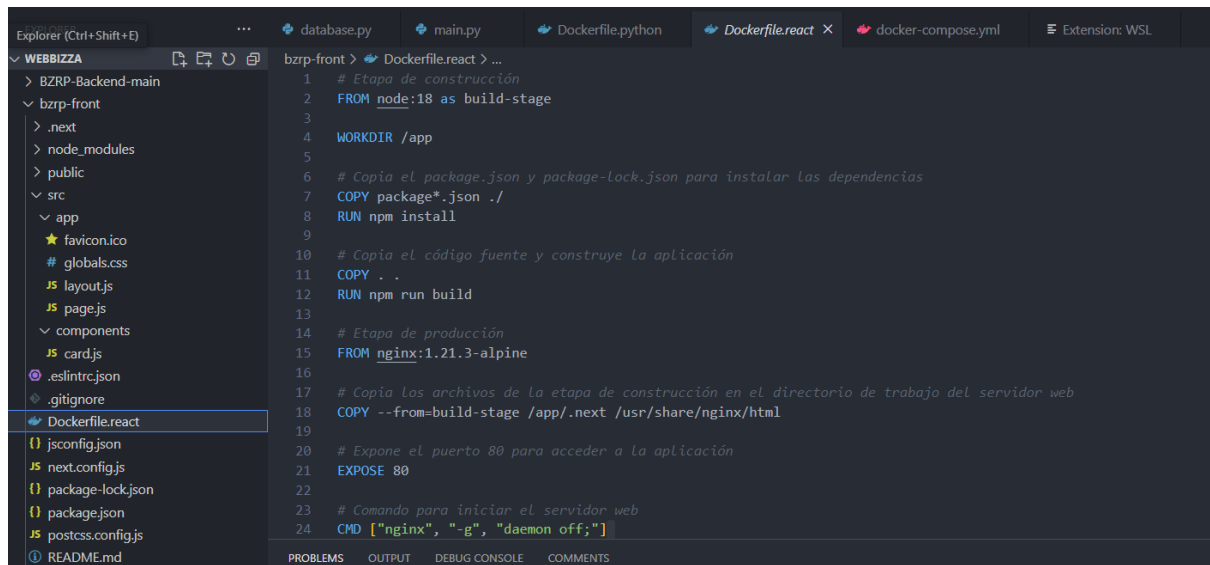
```
EXPLORER
...
database.py
main.py
Dockerfile.python X
docker-compose.

WEBBIZZA
  BZRP-Backend-main
    > __pycache__
    > .venv
    > .gitignore
    > database.py
    > Dockerfile.python
    > main.py
    > models.py
    > README.md
    > requirements.txt
  bzrp-front
    > .next
    > node_modules
    > public
    > src
    > app
    > favicon.ico
    > globals.css
    > layout.js

BZRP-Backend-main > Dockerfile.python > ...
1  # Usa una imagen de Python
2  FROM python:3.8
3
4  # Instala PostgreSQL en el contenedor
5  RUN apt-get update && apt-get install -y postgresql
6
7  # Copia el código fuente de la aplicación
8  COPY . /app
9
10 # Establece el directorio de trabajo
11 WORKDIR /app
12
13 # Instala las dependencias de Python
14 RUN pip install --no-cache-dir -r requirements.txt
15
16 # Expone el puerto 8000 para acceder a la aplicación
17 EXPOSE 8000
18
19 # Comando para iniciar la aplicación Python
20 CMD ["uvicorn", "main:app"]
21
```

Acá hice lo mismo, pero para el Front, instando según los pasos que decía, el npm install y construyendo el proyecto con el código que se muestra en el archivo.

Como dato tenías que estar en la ruta correcta dentro del terminal para ejecutar cada uno de los códigos.



The screenshot shows the Visual Studio Code interface with a file explorer on the left and a code editor on the right. The file explorer shows a project structure for 'WEBBIZZA' with a 'bzipr-front' directory. The code editor displays the 'Dockerfile.react' file, which contains the following content:

```
bzipr-front > Dockerfile.react > ...
1  # Etapa de construcción
2  FROM node:18 as build-stage
3
4  WORKDIR /app
5
6  # Copia el package.json y package-lock.json para instalar las dependencias
7  COPY package*.json ./
8  RUN npm install
9
10 # Copia el código fuente y construye la aplicación
11 COPY . .
12 RUN npm run build
13
14 # Etapa de producción
15 FROM nginx:1.21.3-alpine
16
17 # Copia los archivos de la etapa de construcción en el directorio de trabajo del servidor web
18 COPY --from=build-stage /app/.next /usr/share/nginx/html
19
20 # Expone el puerto 80 para acceder a la aplicación
21 EXPOSE 80
22
23 # Comando para iniciar el servidor web
24 CMD ["nginx", "-g", "daemon off;"]
```

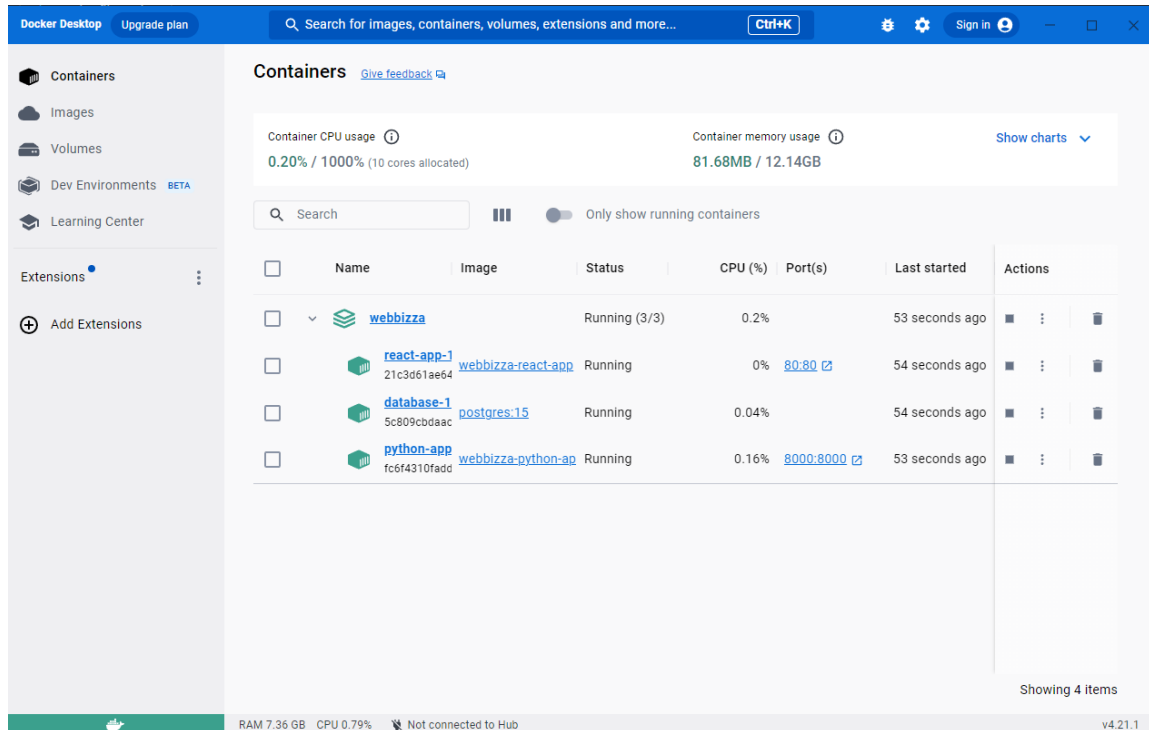
The bottom of the interface shows tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'COMMENTS'.

En el directorio raíz de mi proyecto, puse el Docker-compose y rellené los datos según mi proyecto.

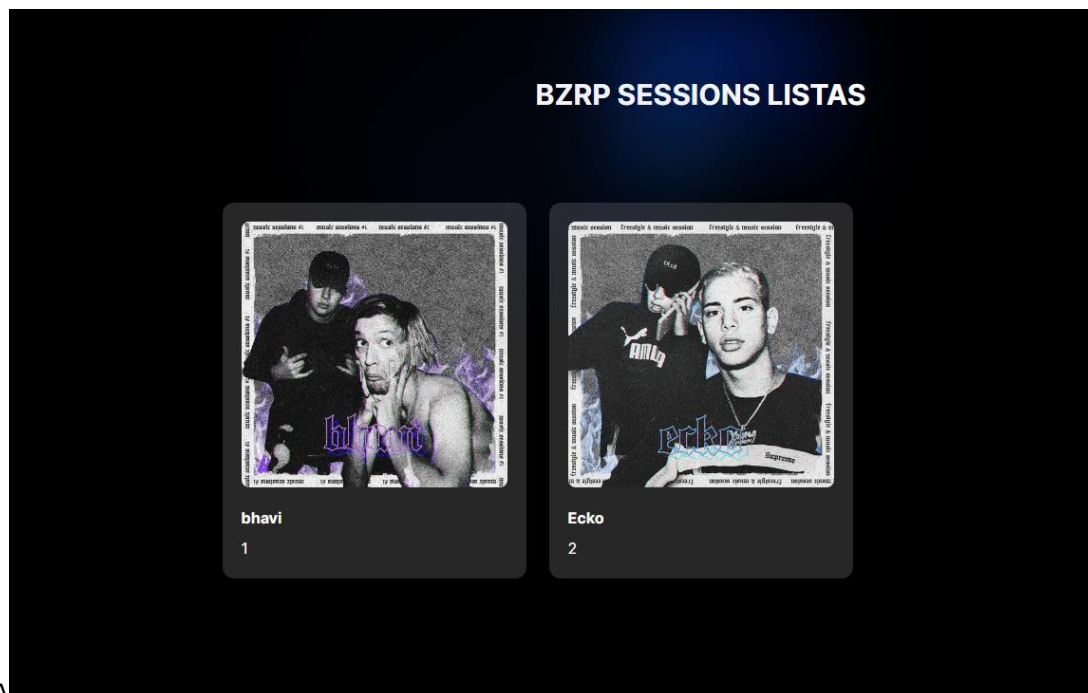
Y agregue la parte de database.

```
🐳 docker-compose.yml
1  version: '3'
2  services:
3    react-app:
4      build:
5        context: ./bzrp-front
6        dockerfile: Dockerfile.react
7      ports:
8        - "80:80"
9      volumes:
10       - react-data:/app/data
11     networks:
12       - my-network
13
14   python-app:
15     build:
16       context: ./BZRP-Backend-main
17       dockerfile: Dockerfile.python
18     ports:
19       - "8000:8000"
20     volumes:
21       - python-data:/app
22     networks:
23       - my-network
24     depends_on:
25       - database
26
27   database:
28     image: postgres:15
29     environment:
30       - POSTGRES_USER=postgres
31       - POSTGRES_PASSWORD=1234567
32       - POSTGRES_DB=bizza-rap
33     volumes:
34       - db-data:/var/lib/postgresql/data
35     networks:
36       - my-network
37
38   networks:
39     my-network:
40
```

Ahora con el comando de docker-compose up –build, lo use estando en la ruta del directorio raíz y se empezó a descargar todo para crear el contenedor en Docker, abajo se ve la imagen una vez finalizo el proceso como se ve en el Docker desktop.



Aquí esta la pagina web funcionando.



Finalmente le di control C para cerrar los servicios y terminar el trabajo.

```
TERMINAL
y gcc (Debian 12.2.0-14) 12.2.0, 64-bit
webbizza-database-1 | 2023-07-12 03:14:18.917 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
webbizza-database-1 | 2023-07-12 03:14:18.917 UTC [1] LOG: listening on IPv6 address ":::", port 5432
webbizza-database-1 | 2023-07-12 03:14:18.925 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.
webbizza-database-1 | 2023-07-12 03:14:18.934 UTC [64] LOG: database system was shut down at 2023-07-12 03:14:18 U
webbizza-database-1 | 2023-07-12 03:14:18.940 UTC [1] LOG: database system is ready to accept connections
Gracefully stopping... (press Ctrl+C again to force)
Aborting on container exit...
[+] Stopping 3/3
 ✓ Container webbizza-react-app-1 Stopped
 ✓ Container webbizza-python-app-1 Stopped
 ✓ Container webbizza-database-1 Stopped
canceled
PS C:\Users\Christian Encalada\Desktop\WEBBIZZA> |
```