Docker

En el directorio raiz del proyecto se crea el archivo docker-compose.yml

Es el que dirigirá el contenedor del backend, frontend y base de datos.

*#version: "3.9"*

services:

backend:

build: ./Backend

container\_name: ventasback\_test

command: *>*

sh -c "uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload"

volumes:

- ./Backend:/app

ports:

- "8383:8000"

env\_file:

- .env

depends\_on:

- db

restart: always

db:

image: postgres:15

container\_name: ventasdb\_test

restart: always

environment:

POSTGRES\_DB: ${POSTGRES\_DB}

POSTGRES\_USER: ${POSTGRES\_USER}

POSTGRES\_PASSWORD: ${POSTGRES\_PASSWORD}

ports:

- "5440:5432"

volumes:

- postgres\_data:/var/lib/postgresql/data

- ./DB/DB.sql:/docker-entrypoint-initdb.d/DB.sql

frontend:

build: ./Frontend

container\_name: ventasfront\_test

volumes:

- ./Frontend:/app

- /app/node\_modules

ports:

- "3003:3000"

stdin\_open: true

tty: true

command: npm start -- --host 0.0.0.0

restart: always

volumes:

postgres\_data:

En el directorio del backend se debe crear el archivo Dockerfile no tiene extensión y allí debe colocarse el código con la imagen que se descargará de python

FROM python:3.12

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 8000

En el directorio del frontend se debe crear el archivo Dockerfile no tiene extensión y allí debe colocarse el código con la imagen que se descargará de python

FROM node:20-alpine

WORKDIR /app

COPY package.json package-lock.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "run", "start", "--", "--host", "0.0.0.0"]