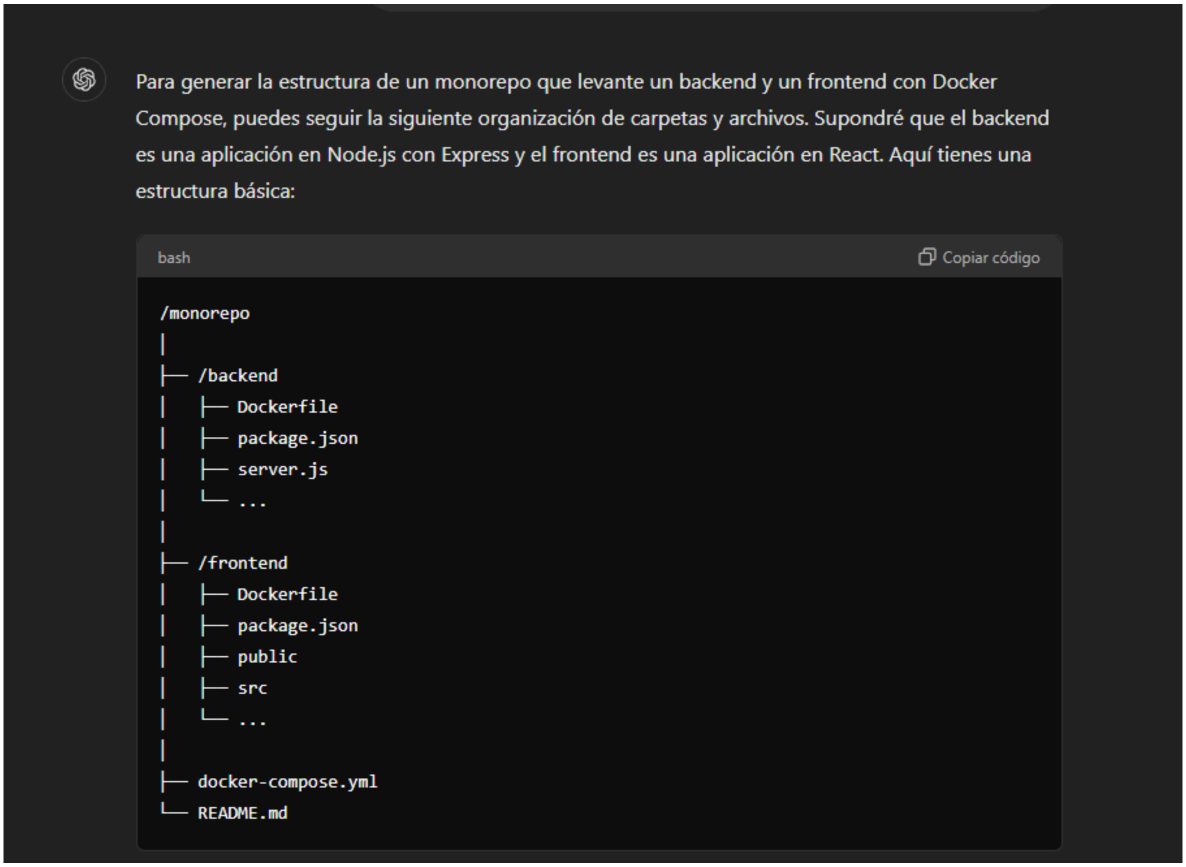
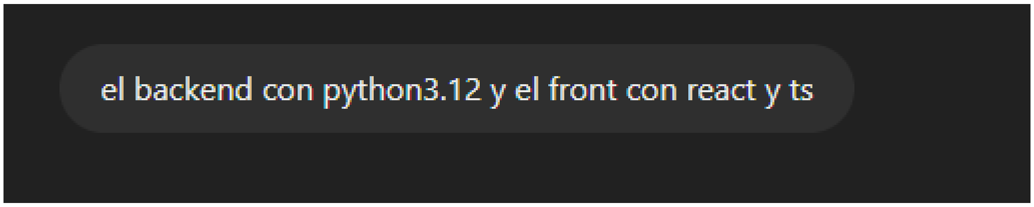
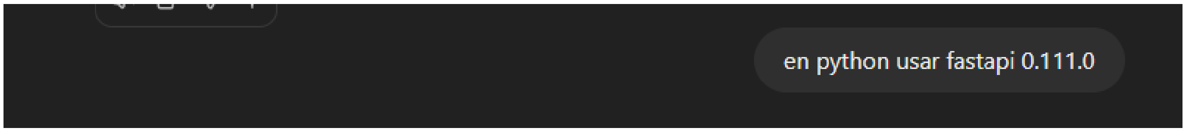


Para generar la estructura de un monorepo que levante un backend y un frontend con Docker Compose, puedes seguir la siguiente organización de carpetas y archivos. Supondré que el backend es una aplicación en Node.js con Express y el frontend es una aplicación en React. Aquí tienes una estructura básica:

```bash
/monorepo
|
├── /backend
|   ├── Dockerfile
|   ├── package.json
|   ├── server.js
|   └── ...
|
├── /frontend
|   ├── Dockerfile
|   ├── package.json
|   ├── public
|   ├── src
|   └── ...
|
├── docker-compose.yml
└── README.md
```

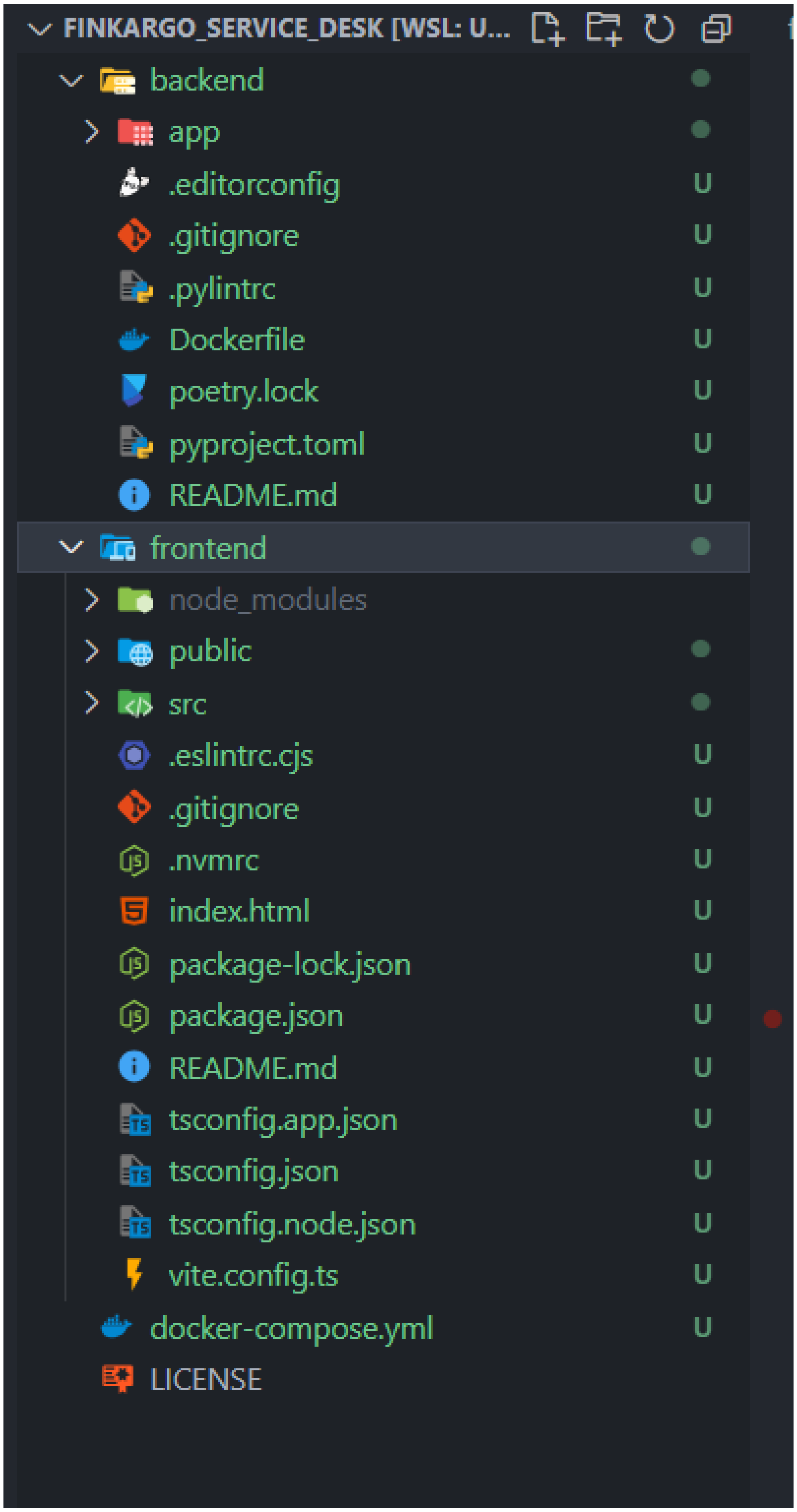


el backend con python3.12 y el front con react y ts

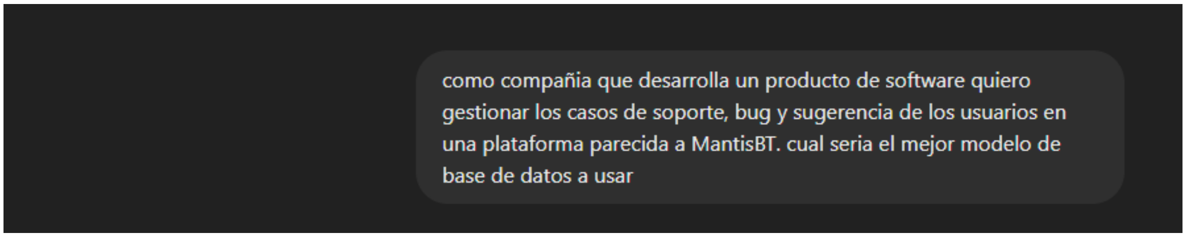


en python usar fastapi 0.111.0

Basado en la estructura propuesta procedí a cambiar el contenido de las carpetas por mis proyectos base



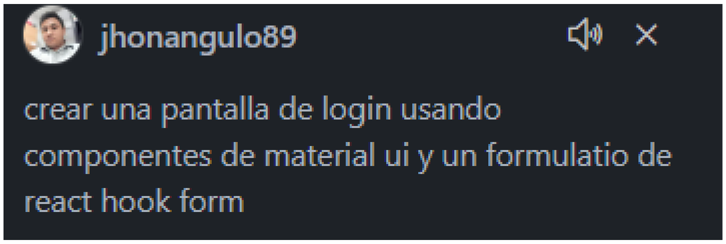
```
FINKARGO_SERVICE_DESK [WSL: U...
  backend                  ●
    app                    ●
    .editorconfig          U
    .gitignore             U
    .pylintrc              U
    Dockerfile             U
    poetry.lock            U
    pyproject.toml         U
    README.md              U
  frontend                 ●
    node_modules
    public                 ●
    src                    ●
    .eslintrc.cjs          U
    .gitignore             U
    .nvmrc                 U
    index.html             U
    package-lock.json      U
    package.json           U
    README.md              U
    tsconfig.app.json      U
    tsconfig.json          U
    tsconfig.node.json     U
    vite.config.ts         U
  docker-compose.yml       U
  LICENSE
```



como compañia que desarrolla un producto de software quiero gestionar los casos de soporte, bug y sugerencia de los usuarios en una plataforma parecida a MantisBT. cual seria el mejor modelo de base de datos a usar

Resultado la consulta

```
-- Enable the pgcrypto extension for UUID generation
CREATE EXTENSION IF NOT EXISTS "pgcrypto";

CREATE TABLE Users (
   user_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
   username VARCHAR(50) NOT NULL,
   email VARCHAR(100) NOT NULL UNIQUE,
   password VARCHAR(255) NOT NULL,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE Projects (
   project_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
   name VARCHAR(100) NOT NULL,
   description TEXT,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE Issues (
   issue_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
   title VARCHAR(255) NOT NULL,
   description TEXT,
   status VARCHAR(50) NOT NULL,
   type VARCHAR(50) NOT NULL,
   priority VARCHAR(50) NOT NULL,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   user_id UUID,
   project_id UUID,
   FOREIGN KEY (user_id) REFERENCES Users(user_id),
   FOREIGN KEY (project_id) REFERENCES Projects(project_id)
);

CREATE TABLE Comments (
   comment_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
   content TEXT NOT NULL,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   user_id UUID,
   issue_id UUID,
   FOREIGN KEY (user_id) REFERENCES Users(user_id),
   FOREIGN KEY (issue_id) REFERENCES Issues(issue_id)
);

CREATE TABLE Attachments (
   attachment_id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
   filename VARCHAR(255) NOT NULL,
   filepath VARCHAR(255) NOT NULL,
   uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   issue_id UUID,
   user_id UUID,
   FOREIGN KEY (issue_id) REFERENCES Issues(issue_id),
   FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```



jhonangulo89

crear una pantalla de login usando componentes de material ui y un formulatio de react hook form