Chatbot IA Closy

Para la realización de este supuesto práctico, se recomendaba utilizar Botpress, pero yo personalmente me resulta más cómodo relizar el desarrollo desde 0 por mi cuenta, por lo que he tomado la decisión de realizarlo así.

Entorno y desarrollo

Primero de todo, comentar que he utilizado para el desarrollo un MacBook.
Tengo dos equipos, y el otro es un Lenovo ThinkPad, pero consideraba que en el Mac iba a ser más fácil de gestionar todo con las herramientas que tenía pensado utilizar.

Base de datos

Para crear mi base de datos **tienda**, he utiizado **PostgreSQL**, ya que he trabajado otras veces con él y me resulta bastante cómodo.

- Dejo una captura para que se vea mi base de datos con el contenido que le he creado para la prueba del ChatBot. He creado una tabla llamada **stock** para introducir los datos.

```
postgres=# \l
                                  List of databases
   Name    |    Owner    | Encoding | Collate | Ctype |      Access privileges
-----------+-------------+----------+---------+-------+-----------------------------
 postgres  | dariogarcia | UTF8     | C       | C     |
 template0 | dariogarcia | UTF8     | C       | C     | =c/dariogarcia             +
           |             |          |         |       | dariogarcia=CTc/dariogarcia
 template1 | dariogarcia | UTF8     | C       | C     | =c/dariogarcia             +
           |             |          |         |       | dariogarcia=CTc/dariogarcia
 tienda    | postgres    | UTF8     | C       | C     | =Tc/postgres               +
           |             |          |         |       | postgres=CTc/postgres
(4 rows)
```

```
dariogarcia — psql postgres — 126×38
Last login: Sat Mar  1 12:34:01 on ttys003
dariogarcia@MacBook-Air-de-Dario ~ % psql postgres
psql (14.17 (Homebrew))
Type "help" for help.

postgres=# SELECT * FROM stock;
 id |  prenda  | talla | color  | cantidad
----+----------+-------+--------+----------
  1 | camiseta | M     | negro  |       10
  2 | pantalon | L     | azul   |        5
  3 | sudadera | S     | rojo   |        8
  4 | falda    | XL    | blanco |        3
(4 rows)

postgres=#
```

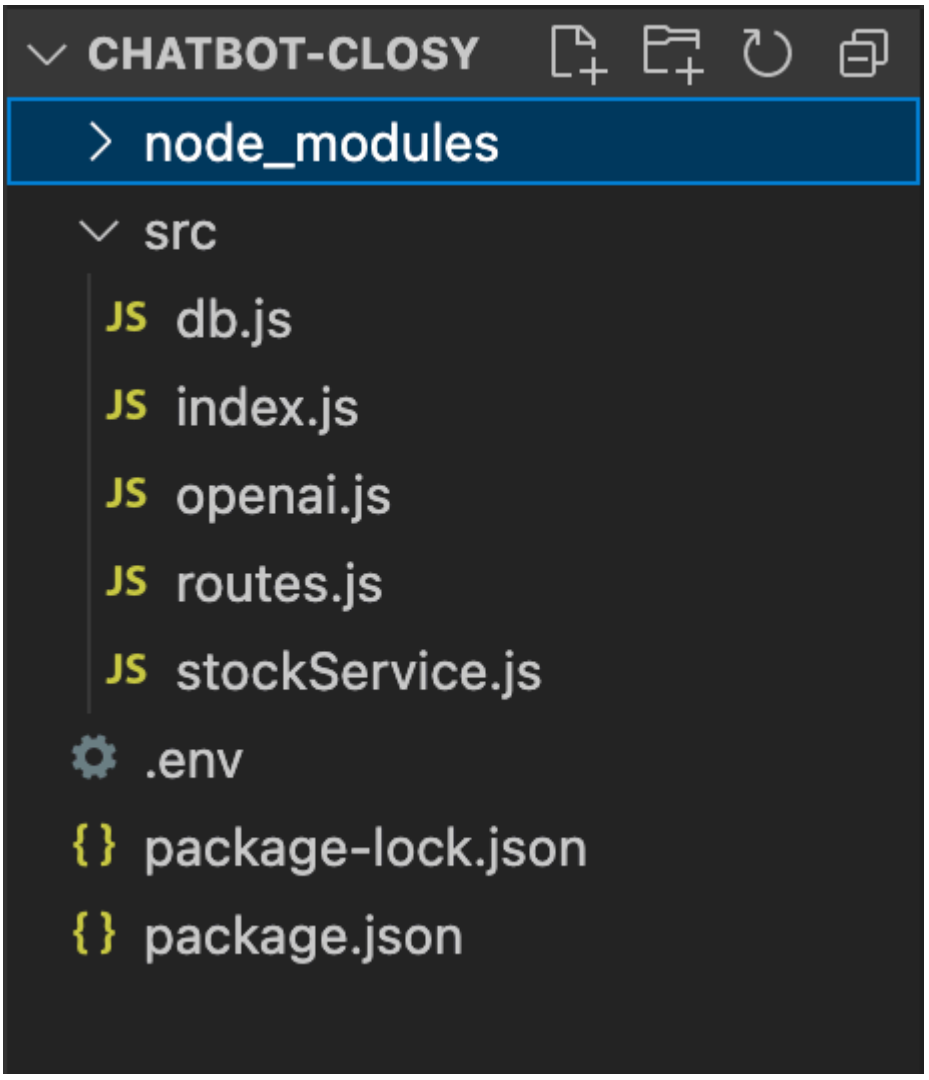
Backend

Para el desarrollo del backend he decidido utilizar las siguientes herramientas:

- PostgreSQL
- Node.js + Express
- OpenAI API

Esta es la estructura de mi proyecto Backend:

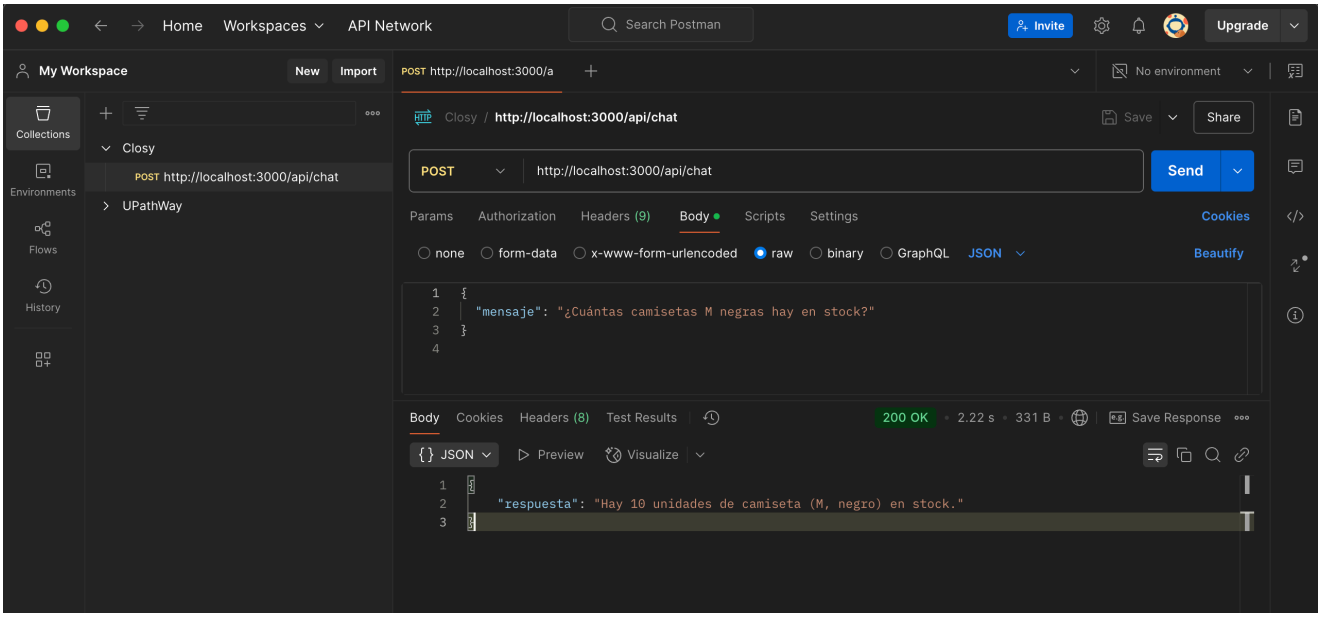
- **db.js**: Configura la conexión a **PostgreSQL** usando el módulo pg.
- **index.js**: Servidor **Express**. Su función es inicializar el servidor, manejar la conexión con la base de datos y exponer las rutas de la API.
- **routes.js**: Define las **rutas del backend** para manejar las solicitudes del **chatbot**, mezclando la lógica de OpenAI con la consulta de stock.
- **openai.js**: Este archivo define la lógica para interactuar con OpenAI y hace que el modelo pueda consultar **stock de productos** usando la función `buscarStock` de `stockService.js`. Es un **intermediario** que decide si OpenAI responde directamente o si necesita consultar el stock antes de responder.
- **stockService.js**: Define la **función** `buscarStock`, encargada de consultar la base de datos PostgreSQL para verificar la disponibilidad de productos en el stock.



- He creado un archivo **.env** que me guardará las variables de entorno que me hagan falta a lo largo del desarrollo, ahí se pueden ver las que se han utilizado.

- Para levantar el proyecto se puede ver en la terminal el comando utilizado.

En un principio hice pruebas con **Postman** antes de tener creado el Frontend, aquí también se puede ver una captura de una petición POST:

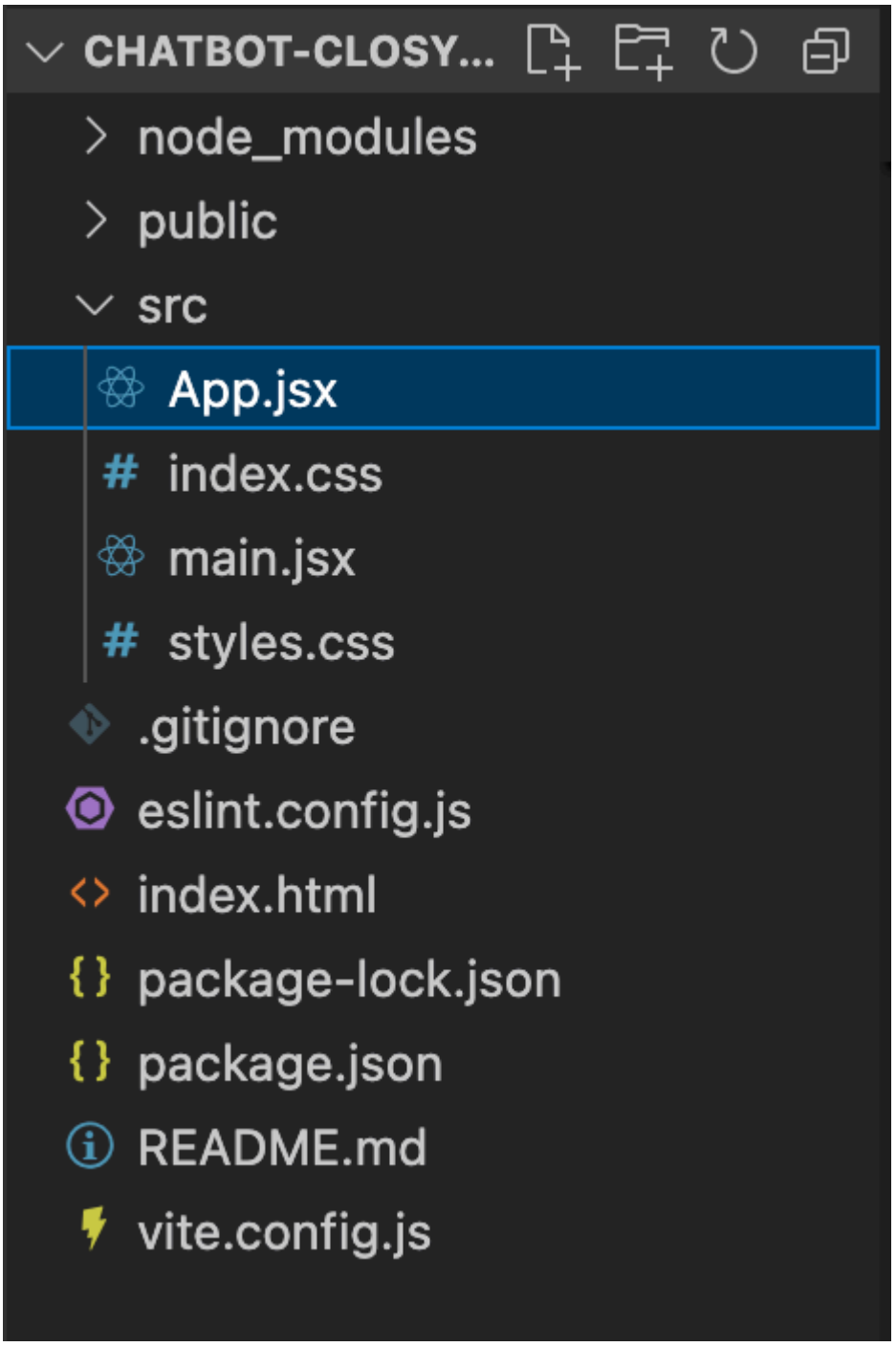


Frontend

Para el desarrollo del front he utilizado **React/Vue** ya que es más personalizable y profesional.

Estructura del proyecto:

- **App.jsx**: Define la interfaz de usuario de un **chatbot** en **React**, que permite enviar mensajes y recibir respuestas desde un servidor backend (`http://localhost:3000/api/chat`). El chatbot muestra los mensajes en pantalla y permite interactuar con la API usando **Axios**.
- **styles.css**: Define los estilos del chat.



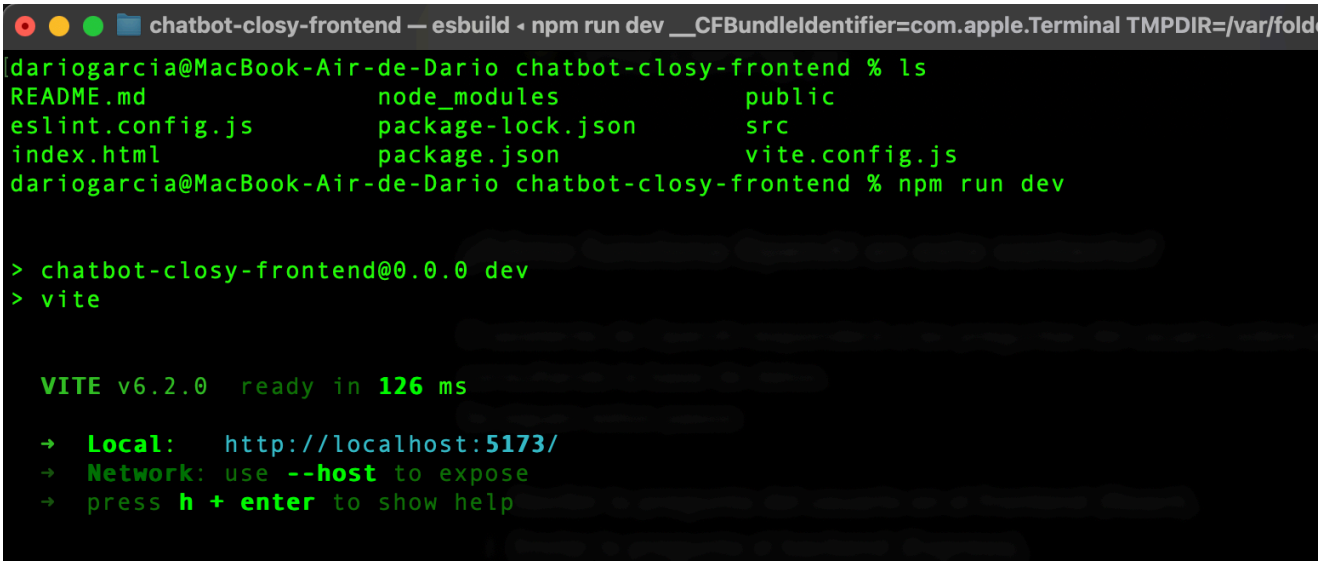
¿Cómo funciona OpenAI en este contexto?

El asistente de OpenAI responderá a las preguntas del usuario sobre el stock de productos consultando la base de datos.
Se siguen estos pasos:

1. **Recibir la pregunta del usuario en el frontend (React).**
2. **Enviar la pregunta al backend (Express).**
3. **El backend envía la consulta a OpenAI usando su API.**
4. **OpenAI devuelve una respuesta basada en nuestro asistente.**
5. **El backend devuelve la respuesta al frontend.**
6. **El frontend muestra la respuesta en el chat.**

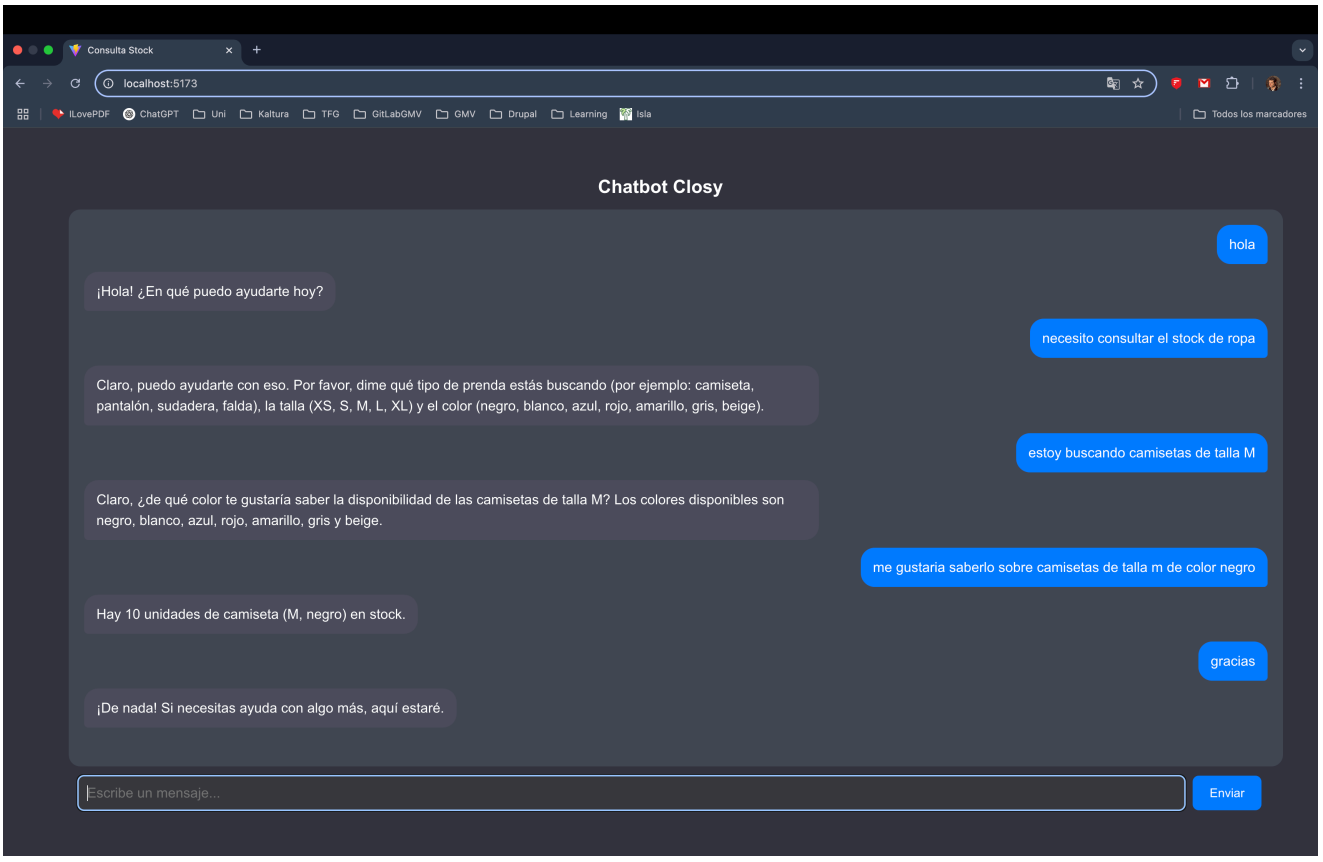
Instalo Dependencias en el Proyecto

- **cors** → Permite que el frontend se comunique con el backend.
- **dotenv** → Carga las claves de OpenAI desde un archivo `.env`.
- **openai** → Cliente oficial para interactuar con OpenAI.
- Para levantar el proyecto:



Interfaz del ChatBot:

- Aquí podemos ver un fragmento de una conversadción con el ChatBot consultando camisetas en la base de datos creada previamente.


Consulta Stock
localhost:5173
ILovePDF
ChatGPT
Uni
Kaltura
TFG
GitLabGMV
GMV
Drupal
Learning
Isla
Todos los marcadores
Chatbot Closy
hola
¡Hola! ¿En qué puedo ayudarte hoy?
necesito consultar el stock de ropa
Claro, puedo ayudarte con eso. Por favor, dime qué tipo de prenda estás buscando (por ejemplo: camiseta, pantalón, sudadera, falda), la talla (XS, S, M, L, XL) y el color (negro, blanco, azul, rojo, amarillo, gris, beige).
estoy buscando camisetas de talla M
Claro, ¿de qué color te gustaría saber la disponibilidad de las camisetas de talla M? Los colores disponibles son negro, blanco, azul, rojo, amarillo, gris y beige.
me gustaria saberlo sobre camisetas de talla m de color negro
Hay 10 unidades de camiseta (M, negro) en stock.
gracias
¡De nada! Si necesitas ayuda con algo más, aquí estaré.
Escribe un mensaje...
Enviar