Chatbot de Pedidos con IA para "La Pizza Feliz"

Este proyecto es una aplicación web completa que simula un chatbot para tomar pedidos en una pizzería. Utiliza un modelo de lenguaje de OpenAI para mantener una conversación natural con el usuario, gestionar un carrito de la compra en tiempo real y guardar el pedido final en una base de datos.

✨ Características Principales

- **Interfaz de Chat Interactiva:** Un frontend limpio y sencillo construido con HTML y JavaScript puro.
- **Resumen de Pedido en Tiempo Real:** Un panel lateral que se actualiza instantáneamente a medida que el usuario interactúa con el bot.
- **Backend Inteligente con IA:** Un servidor en Python con FastAPI que se conecta a OpenAI para procesar el lenguaje natural.
- **Memoria de Conversación:** El chatbot recuerda el historial de la conversación y el estado del carrito para ofrecer una experiencia fluida.
- **Menú Dinámico:** El menú de la pizzería se carga desde una base de datos, permitiendo añadir o modificar productos sin cambiar el código.
- **Persistencia de Datos:** Los pedidos completados se guardan en una base de datos SQLite.

🚀 Guía de Instalación y Ejecución en Windows

Sigue estos pasos para poner en marcha el proyecto en tu máquina local.

1. Prerrequisitos

- **Python 3.7+** instalado en tu sistema.
- Una clave de API de OpenAI.

2. Configuración del Backend

1. **Abre una terminal:**

   Puedes usar PowerShell o el Símbolo del sistema (cmd). Para abrirlo, pulsa la tecla de Windows y escribe "PowerShell".

2. **Navega a la carpeta `backend`:**

Usa el comando `cd` para moverte al directorio donde guardaste el proyecto.

```
Plain Text
cd ruta\a\tu\proyecto\chatbot_pedidos\backend
```

3. **Crea un entorno virtual:**

   Este comando crea una carpeta llamada `venv` que contendrá todas las librerías específicas para este proyecto.

```
Plain Text
python -m venv venv
```

4. **Activa el entorno virtual:**

   Es crucial activar el entorno antes de instalar nada.

```
Plain Text
.\venv\Scripts\activate
```

   Verás que `(venv)` aparece al principio de la línea de tu terminal.

5. **Instala las dependencias:**

   Este comando instalará FastAPI, Uvicorn, OpenAI y las demás librerías necesarias.

```
Plain Text
pip install "fastapi[all]" uvicorn openai python-dotenv
```

6. **Configura tu clave de API:**

   Dentro de la carpeta `backend`, crea un archivo de texto simple, llámalo `.env` y escribe dentro tu clave de API de OpenAI.

```
Plain Text
OPENAI_API_KEY="tu_clave_secreta_aqui"
```

3. Ejecución del Proyecto

1. **Inicia el servidor Backend:**

   Desde la carpeta `backend` y con el entorno virtual activado, ejecuta:

```
Plain Text
```

```
uvicorn main:app --reload
```

El servidor estará disponible en `http://127.0.0.1:8000`.

2. **Inicia el Frontend:**

   Navega a la carpeta `frontend` y simplemente haz doble clic en el archivo `index.html` en tu navegador web.

📂 Análisis de la Estructura de Archivos

backend/main.py

Es el cerebro de la aplicación. Gestiona la API, la lógica de la IA y la base de datos.

- **Funciones Principales:**
  - `inicializar_db()`: Crea las tablas de la base de datos (`productos`, `pedidos`, `items_pedido`) si no existen al arrancar el servidor. También puebla la tabla de productos con un menú inicial.
  - `guardar_pedido_en_db(pedido_data)`: Se activa cuando un pedido se completa. Recibe el JSON del pedido y lo guarda de forma segura en la base de datos.
  - `chat_endpoint(request)`: Es la ruta principal de la API. Recibe el historial de la conversación, construye un prompt detallado para la IA, envía la petición a OpenAI y devuelve la respuesta conversacional junto con el estado actualizado del carrito.

frontend/js/script.js

Controla toda la interactividad del lado del cliente.

- **Variables Clave:**
  - `conversationHistory`: Un array que almacena cada mensaje del usuario y del bot para mantener el contexto de la conversación.
- **Funciones Principales:**
  - `addMessage(sender, text)`: Dibuja un nuevo mensaje en la ventana del chat, con estilos diferentes para el usuario y el bot.
  - `updateOrderSummary(cart)`: Recibe el estado del carrito desde el backend y redibuja el panel del resumen del pedido, mostrando los productos, cantidades y el total.
  - `handleSendMessage()`: Se ejecuta al enviar un mensaje. Envía el historial al backend, recibe la respuesta, y llama a las funciones para actualizar la UI.

- `window.onload()` : Se ejecuta al cargar la página para mostrar el mensaje de bienvenida inicial.

`frontend/index.html`

Define la estructura de la página web. Contiene los `div` para el contenedor del chat y el panel del resumen del pedido.

🛠️ Tecnologías Utilizadas

- **Backend:** Python, FastAPI, Uvicorn, OpenAI, python-dotenv, SQLite.
- **Frontend:** HTML5, JavaScript (Vanilla).
- **IA:** GPT-3.5-Turbo de OpenAI.