Explicación del Backend: Una Guía Didáctica

Introducción

En esta guía, te voy a explicar cómo funciona el backend de nuestra aplicación. Vamos a ir desglosando cada uno de los componentes importantes, de manera que puedas entender cómo se relacionan y cómo contribuyen al funcionamiento general. Siéntete libre de seguir este recorrido paso a paso, como si estuviéramos charlando cara a cara sobre código.

Archivo Principal: app.py

El archivo app.py es el punto de entrada de nuestra aplicación. Imagínate que este archivo es como la puerta principal de una casa: todo comienza aquí. Vamos a ver qué elementos lo componen y cuál es su rol.

Importaciones de Módulos

Lo primero que hacemos es importar varias bibliotecas. Esto es como si estuviéramos preparando nuestras herramientas antes de empezar a construir:

- dotenv: Utilizamos dotenv para cargar variables de entorno desde un archivo llamado .env. Este archivo contiene configuraciones importantes, como las credenciales para conectar con otros servicios.
- aiohttp.web: Esta biblioteca nos ayuda a crear un servidor web, que es la pieza fundamental para que nuestra aplicación pueda recibir peticiones y responder a los usuarios.
- Módulos internos: También importamos attach_rag_tools de ragtools y RTMiddleTier de rtmt. Estos son nuestros componentes personalizados que nos ayudarán a conectar con las APIs y gestionar respuestas.
- Azure: Utilizamos azure.identity y azure.core.credentials para manejar la autenticación con los servicios de Azure.

Explicación del Backend: Una Guía Didáctica

Cargando Variables de Entorno

Ahora que tenemos las herramientas listas, necesitamos configurarlas. Esto lo hacemos cargando nuestras variables de entorno, que son como las instrucciones que le damos a nuestra aplicación para que sepa cómo conectarse con Azure. Utilizamos load_dotenv() para esto, y luego cargamos variables como AZURE_OPENAI_ENDPOINT, AZURE_OPENAI_API_KEY, entre otras.

Autenticación con Azure

Para trabajar con Azure, primero debemos autenticarnos. Dependiendo de la configuración, usamos DefaultAzureCredential() o AzureKeyCredential(). Es como si estuviéramos mostrando nuestra identificación para entrar a un lugar seguro.

Creando la Aplicación Web

Con todo configurado, ahora podemos crear la aplicación web. Utilizamos web.Application() de aiohttp para crear una instancia de nuestra app, algo así como darle vida a nuestra estructura.
Luego, inicializamos una instancia de RTMiddleTier, que es como un asistente dentro de nuestra aplicación. Este asistente se encarga de manejar las solicitudes y consultas hacia Azure, y tiene una configuración especial (un "mensaje del sistema") que le dice cómo debe comportarse. En este caso, le indicamos que sea breve y consulte siempre la base de conocimientos antes de responder.

Herramientas RAG (Retrieval-Augmented Generation)

Utilizamos la función attach_rag_tools() para conectar herramientas adicionales al RTMiddleTier. Estas herramientas nos permiten hacer búsquedas en una base de conocimientos antes de generar una respuesta, lo que ayuda a hacer respuestas más precisas y útiles.

Explicación del Backend: Una Guía Didáctica

Definiendo Rutas de la Aplicación

Las rutas definen qué sucede cuando alguien visita una URL específica de nuestra aplicación:

- Ruta Raíz (/): Cuando el usuario visita la página principal, le mostramos un archivo HTML llamado index.html.
- Archivos Estáticos: También configuramos una ruta para servir archivos estáticos, como JavaScript, CSS y otros recursos que la interfaz necesita.

Iniciando la Aplicación Web

Finalmente, nuestra aplicación se inicia en localhost en el puerto 8765. Esto significa que podemos acceder a ella escribiendo esa dirección en nuestro navegador.

Explorando ragtools.py

Este archivo contiene las herramientas que ayudan a nuestra aplicación a buscar información y fundamentar las respuestas. Vamos a verlo en detalle.

Búsqueda en la Base de Conocimientos

La función _search_tool() utiliza SearchClient de Azure para realizar búsquedas en nuestra base de conocimientos. Es como preguntarle a una enciclopedia en línea para obtener respuestas más precisas.

Reporte de Fuentes

La función _report_grounding_tool() se encarga de citar las fuentes de la base de conocimientos que se utilizaron en la respuesta. Esto es útil para asegurarnos de que nuestras respuestas estén respaldadas por información confiable.

Explicación del Backend: Una Guía Didáctica

Conectando las Herramientas

La función attach_rag_tools() conecta estas herramientas al RTMiddleTier, para que estén disponibles cuando el asistente las necesite.

Explorando rtmt.py

Finalmente, tenemos el archivo rtmt.py, donde se implementa RTMiddleTier. Este archivo es fundamental, ya que se encarga de manejar la comunicación en tiempo real entre nuestra aplicación y los usuarios.

Clase Principal: RTMiddleTier

La clase RTMiddleTier gestiona las conexiones mediante WebSockets. Esto significa que podemos tener una comunicación fluida y en tiempo real, como una conversación constante entre el servidor y el cliente.

Manejo de Herramientas y Mensajes

RTMiddleTier también tiene la capacidad de gestionar herramientas, es decir, ejecutar funciones que están definidas de antemano para realizar tareas específicas. Además, tiene varios métodos para procesar mensajes que van del cliente al servidor y viceversa, dándonos flexibilidad para personalizar el flujo de información.