Arquitectura del Sistema del Agente de IA Conversacional

El Agente de IA Conversacional está diseñado con una arquitectura modular y escalable, dividida en varias capas funcionales que permiten una clara separación de responsabilidades y facilitan el mantenimiento y la expansión. La arquitectura se puede visualizar como un sistema de tres capas principales: Interfaz de Usuario (Frontend), Lógica de Negocio (Backend) y Servicios de IA/Datos.

1. Capa de Interfaz de Usuario (Frontend)

Esta capa es la cara visible del agente, interactuando directamente con el usuario final. Está diseñada para ser intuitiva y responsiva, proporcionando una experiencia de usuario fluida.

- **Tecnologías**: HTML5, CSS3 (con Bootstrap para diseño responsivo), JavaScript (para interactividad).
- **Componentes Clave**:
  - **Chat UI**: Interfaz principal donde los usuarios escriben sus consultas y reciben respuestas del agente.
  - **Dashboard de Gestión**: Secciones para la administración de la base de conocimiento, visualización de análisis y configuración del agente.
  - **Elementos Interactivos**: Botones, campos de entrada, indicadores de estado (conectado/desconectado).

2. Capa de Lógica de Negocio (Backend)

El backend actúa como el cerebro del agente, orquestando las interacciones entre la interfaz de usuario y los servicios de IA/datos. Está construido sobre un framework web moderno que garantiza alto rendimiento y escalabilidad.

- **Tecnologías**: Python 3.11, FastAPI (framework web asíncrono).

- **Componentes Clave**:
  - **API RESTful**: Conjunto de endpoints que exponen las funcionalidades del agente (ej. `/api/ask`, `/api/knowledge`, `/api/analytics`).
  - **Manejadores de Solicitudes**: Procesan las peticiones del frontend, validan los datos y coordinan la llamada a los servicios de IA y la base de conocimiento.
  - **Servidor Uvicorn**: Servidor ASGI de alto rendimiento para ejecutar la aplicación FastAPI.

3. Capa de Servicios de IA y Datos

Esta capa contiene los módulos inteligentes y la persistencia de datos que potencian las capacidades del agente. Es el corazón de la inteligencia del sistema.

- **Tecnologías**: Python, SQLite, librerías de ML (sentence-transformers, scikit-learn), OpenAI API.
- **Componentes Clave**:
  - **Servicio GenAI (`genai_service.py`)**: Encargado de interactuar con modelos de lenguaje grandes (LLMs) para generar respuestas coherentes y contextualizadas. Utiliza la API de OpenAI (o un modelo simulado en modo demo).
  - **Servicio de Embeddings (`embedding_service.py`)**: Transforma el texto en representaciones numéricas (embeddings) que permiten la búsqueda semántica y la comparación de similitudes. Utiliza modelos pre-entrenados como `all-MiniLM-L6-v2`.
  - **Base de Conocimiento (`knowledge_base.py`)**: Un módulo que gestiona el almacenamiento y la recuperación de información estructurada. Implementado con SQLite para almacenar documentos y sus embeddings, permitiendo una recuperación eficiente de información relevante.
  - **Clasificador de ML (`ml_classifier.py`)**: Un modelo de Machine Learning (ej. `RandomForestClassifier`) entrenado para clasificar las consultas de los usuarios en categorías predefinidas (ej. RRHH, Tecnología, Procesos). Esto ayuda a dirigir la consulta al módulo o información correcta.

- **Templates de Prompts (`prompt_templates.py`)**: Colección de plantillas de texto que guían al LLM para generar respuestas específicas y con el formato deseado, mejorando la calidad y relevancia de las interacciones.

Flujo de Interacción Típico

1. El usuario ingresa una consulta en la **Interfaz de Usuario (Frontend)**.
2. La consulta se envía a la **API RESTful** en la capa de Lógica de Negocio (Backend).
3. El Backend utiliza el **Clasificador de ML** para determinar la intención de la consulta.
4. Si la consulta requiere información específica, el Backend envía la consulta al **Servicio de Embeddings** para generar su representación numérica.
5. Con el embedding, la **Base de Conocimiento** busca documentos o fragmentos de texto relevantes.
6. La información recuperada, junto con la consulta original y un **Template de Prompt** adecuado, se envía al **Servicio GenAI**.
7. El Servicio GenAI genera una respuesta utilizando el LLM.
8. La respuesta se envía de vuelta al Backend y luego a la **Interfaz de Usuario (Frontend)** para ser mostrada al usuario.

Esta arquitectura garantiza un sistema robusto, eficiente y fácil de mantener, capaz de manejar diversas consultas y proporcionar respuestas precisas y contextualizadas.

Renzo patricio valencia oyarce
Renzo Valencia Oyarce
21-06-2025