Documentación de Desarrollo del Asistente Virtual Úbatron para la UBA

1. Introducción

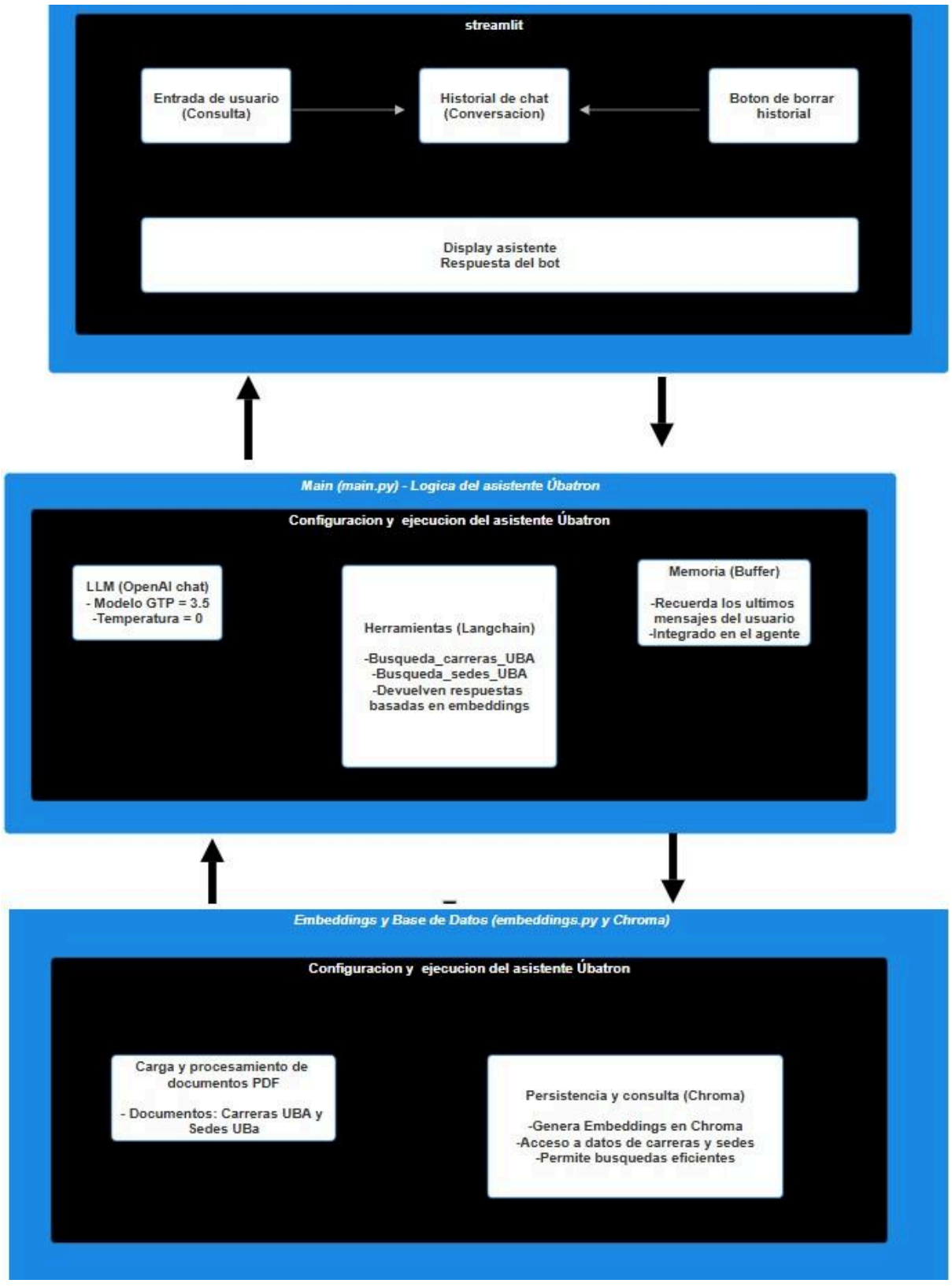
Este documento detalla el desarrollo de un asistente virtual llamado **Úbatron**, diseñado para la Universidad de Buenos Aires (UBA). Úbatron ayuda a los estudiantes a encontrar carreras y sedes de la universidad, utilizando técnicas avanzadas de inteligencia artificial, memoria conversacional y prompting. El asistente ha sido implementado localmente con **Streamlit** y **LangChain**.

2. Objetivos

El objetivo es crear un asistente que:

- Proporcione recomendaciones de carreras y sedes de la UBA.
- Implemente memoria conversacional para ofrecer respuestas contextuales.
- Despliegue una interfaz amigable y atractiva con personalidad definida en idioma español argentino.
- Funcione en un entorno local para garantizar la privacidad y control de los datos.

3. Diagrama de Arquitectura



Explicación Detallada del Diagrama

1. **Interfaz de Usuario (app.py)**:

    - La aplicación utiliza **Streamlit** para crear una interfaz donde el usuario puede ingresar preguntas y ver respuestas del asistente.
    - Los elementos principales incluyen un campo para la entrada del usuario, un botón para borrar el historial y un historial de conversación donde se muestran las interacciones pasadas.
    - **Roles Clave**:
        - **Entrada de Usuario**: Captura la consulta.
        - **Historial de Chat**: Guarda y muestra las conversaciones previas.
        - **Botón de Borrar Historial**: Permite al usuario limpiar el historial de chat.
2. **Asistente Virtual Úbatron (main.py)**:
    - **Modelo LLM (OpenAI Chat)**: Se utiliza GPT-3.5 como motor de procesamiento para responder consultas, configurado con una temperatura baja para asegurar respuestas más coherentes.
    - **Memoria**: Usa `ConversationBufferWindowMemory` para mantener un historial de interacción con el usuario.
    - **Herramientas (Tools)**:
        - `Busqueda_carreras_UBA` y `Busqueda_sedes_UBA` se configuran en este archivo usando LangChain y permiten al asistente recuperar información relevante de los embeddings.
3. **Embeddings y Base de Datos (embeddings.py y Chroma)**:
    - En `embeddings.py`, se cargan y procesan los documentos PDF (carreras y sedes de la UBA), extrayendo y estructurando información.
    - **Chroma** actúa como la base de datos para almacenar estos embeddings y facilitar la consulta posterior mediante herramientas.
    - **Roles Clave**:
        - **Persistencia de Datos**: Almacena los embeddings de manera permanente.
        - **Función de Consulta**: Facilita la búsqueda eficiente en respuesta a consultas del usuario.

4. Estructura de Archivos

- **main.py**: Archivo principal que define el agente de LangChain y la lógica de consulta.
- **Front.py**: Archivo para la interfaz de usuario de Streamlit.
- **embeddings.py**: Archivo de preprocesamiento de documentos y configuración de embeddings.
- **.env**: Archivo de variables de entorno con configuraciones confidenciales, como claves de API.

5. Configuración y Funcionamiento

5.1 Embeddings y Base de Datos

En **embeddings.py**, se define la función de embeddings de OpenAI y se utiliza **Chroma** como base de datos. Los documentos PDF (carreras y sedes de la UBA) se preprocesan y

dividen en fragmentos basados en títulos y contenidos para facilitar el acceso semántico a la información. La estructura de los documentos asegura que la consulta sea rápida y precisa.

5.2 Prompting y Agentes

En **main.py**, se configura el `prompt` para que el asistente tenga un tono amigable y proporcione respuestas claras en español argentino. Úbatron utiliza dos herramientas (`Busqueda_carreras_UBA` y `Busqueda_sedes_UBA`) para responder preguntas sobre carreras y ubicaciones. Se configura el modelo `gpt-3.5-turbo` con una baja temperatura y top_p para mejorar la coherencia y precisión en las respuestas.

5.3 Memoria Conversacional

La memoria se implementa mediante `ConversationBufferMemory` en LangChain, permitiendo al asistente recordar las interacciones del usuario. Esto le permite ofrecer respuestas personalizadas y basadas en el contexto de la conversación.

5.4 Interfaz de Usuario

En **front.py**, la interfaz de Streamlit despliega la interacción entre el usuario y el asistente en un formato de chat. El usuario puede escribir consultas y ver las respuestas de Úbatron en tiempo real. La interfaz también cuenta con un botón para limpiar el historial de conversación y reiniciar la memoria.