**Frontend (React) & API Communication & Visualización SQL**

* Establecer ambientes y modelos de creación de chat Webview.

**Día 1 (29/04/2025):** Configurar entorno React, instalar dependencias iniciales (create-react-app/Vite, axios o similar, quizás una lib UI). Acceso a GIT pruebas.

**Preparar el ambiente para desarrollo**

* Definir conexión de front wv con tablas y manejo de información. Secuencias iniciales de mensajes de prueba.

**Día 2 (30/04/2025):** Lectura de base de datos de clientes, lectura de información y acceso especialmente la tabla messages para entender la estructura del historial que necesitará mostrar.

Crear la estructura de carpetas/componentes inicial para la app React.

* Componentes de gestión para que Natalia responda. Y contenedores de mensajes. Instalación y lectura de gestión.

**Día 3 (02/05/2025):** Implementar el componente principal ChatContainer. Implementar el componente MessageDisplay que recibe un objeto mensaje (ej. { sender: 'user', text: 'Hola' } o { sender: 'bot', text: '...', options: [...] }) y lo muestra (texto simple inicialmente).

En esta parte de ejecutan prueba de lectura y gestión básica para desarrollar las conexiónes de comunicación.

* Visual de chat Webview (acá se puede replicar el chat interno que se tiene como base. y lectura de base de clientes. Identificar respuesta de Natalia

**Día 4: (03/05/2025)** Crear la estructura básica de la página/vista que contendrá el ChatContainer. Implementar una lista simple para mostrar varios mensajes (usando datos mock iniciales).

* Creación la lógica de respuesta de mensajes mediante Post a endpoint que se entrega para comunicación. Recepción de endpoint y envio por chat.

**Día 5: (05/05/2025)** Implementar la lógica para hacer llamadas HTTP POST al endpoint /chat/message del backend. Usar axios o Workspace. Enviar un user\_id fijo o generado simple.

* Comportamiento de respuesta de mensajes y envio a backend

**Día 6 (06/05/2025):** Recibir la respuesta del backend y actualizar el estado de la aplicación React para mostrar el nuevo mensaje del bot en el feed. Conectar el área de input de texto básico para enviar mensajes.

* Botones en front y visual de conocimiento (apoyo en lecturas bases de conocimiento) para que construyamos la lógica de nodos de conocimiento.

**Día 7: (07/05/2025)** Implementar el componente ButtonOptions que recibe una lista de opciones de botones y renderiza botones clickeables.

* Adopción e interpretar mesnajes del back para transformar en botones con textos específicos. Acá miremos si es mejor dejarlo de manera que lo identifique y procese por conocimiento de machine o establecidos (quemados)

**Día 8: (08/05/2025)** Implementar la lógica para que la aplicación React alterne entre mostrar el área de input de texto o el componente ButtonOptions basándose en la respuesta recibida del backend (si la respuesta incluye options, mostrar botones; si no, mostrar input de texto o nada).

* Lógica de captación de mensaje y log… variación de pruebas diversas por tipología.

**Día 9: (09/05/2025)** Implementar la lógica para que al hacer clic en un botón, se capture el value de ese botón.

* Envio de mensaje de botón a backend para lógica de logs y procesos.

**Día 10: (12/05/2025)**  Implementar la lógica para que al hacer clic en un botón, se envíe el value capturado al backend a través de la API /chat/message. Después de enviar, limpiar el área de botones y esperar la respuesta.

* Solicitud de histórico de conversación por cedula identificada.

**Día 11: (13/05/2025)** Implementar la lógica para llamar al endpoint GET /chat/history/{userId} al cargar la página de chat, recibir la lista de mensajes históricos y popular el feed de mensajes inicial.

* Perfeccionar la visualización del chat y pruebas unitarias y estrés de sistema.

**Día 12:** **(14/05/2025)** Refinar la visualización de los mensajes (MessageDisplay) para distinguir visualmente entre mensajes del usuario y del bot. Refinar la apariencia de los botones.

* Mirar si es necesario que se guarde cache de chat de acuerdo a un id de identificación interno.

**Día 13: (15/05/2025)** (Manejar el user\_id en el frontend: generarlo si no existe (ej. un UUID guardado en localStorage) y enviarlo en cada petición al backend pueda mantener el estado de la conversación correcto.

* Union de códigos de back y front completos.

**Día 14** **(16-18/05/2025)** **Integración con Backend** Probar el flujo completo React <-> Python API. Debugging intensivo: verificar que los datos enviados y recibidos tienen el formato correcto esperado por ambos lados. Asegurarse de que la UI reacciona correctamente a diferentes tipos de respuestas (texto solo, texto con botones).

* Pruebas finales y presentacion.

**Día 15: (19/05/2025)** Refinamiento final de la UI. Asegurarse de que la aplicación React levanta correctamente desde cero usando el README. Asegurarse de que la carga de historial funciona. Documentar cómo configurar el frontend y conectarlo al backend.