Diagrama de sistema

Datos: Se procesaron datos en objetos JSON que se pueden leer desde archivos, posteriormente se puede reemplazar la lectura de los datos desde una base de datos o un webAPI que los exponga en formato JSON

Servicio de chat: Se requiere un servidor HTTP para procesar los mensajes e interactuar con los datos del sistema.

Cliente web: Se puede consumir desde un cliente REST (como Postman), se puede exponer para usar desde algún widget en alguna aplicación web o desde algún otro servidor como Twilio para consumir desde alguna aplicación como WhatsApp o por SMS.

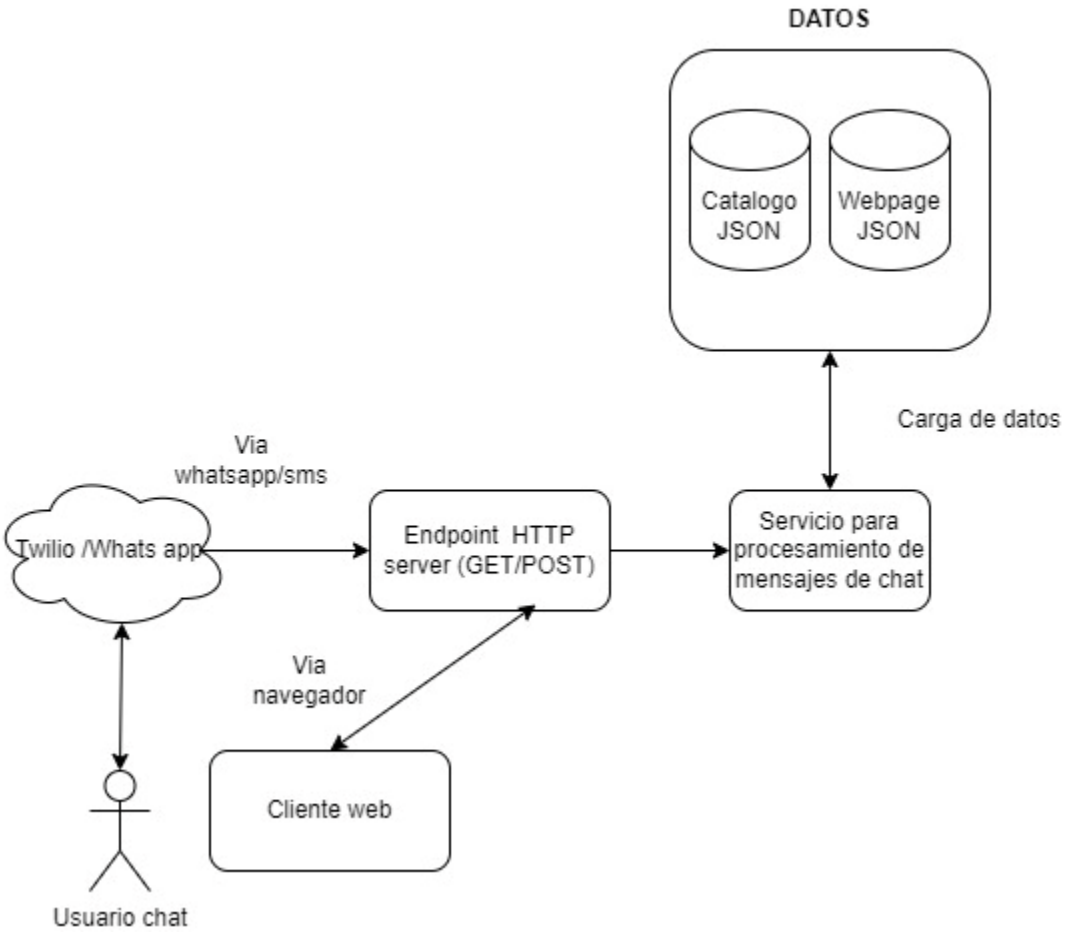


Diagrama de prompts

El servicio de chat consiste en dos pasos,

1. Clasificar el mensaje recibido por el usuario.
2. Dependiendo del resultado de la clasificación procesar el mensaje.

Los siguientes casos se pueden resolver directamente con prompts en donde se incluyen los datos disponibles en el sitio web: Plataforma, Sedes. Posteriormente se pueden agregar más casos al clasificador y es escalable para responder a más información.

Los siguientes casos utilizan datos de catalogo los cuales requieren un diseño que permita escalar el sistema considerando que el catálogo puede crecer de 100 a miles de autos y no se puede cargar toda esta información en un solo prompt.

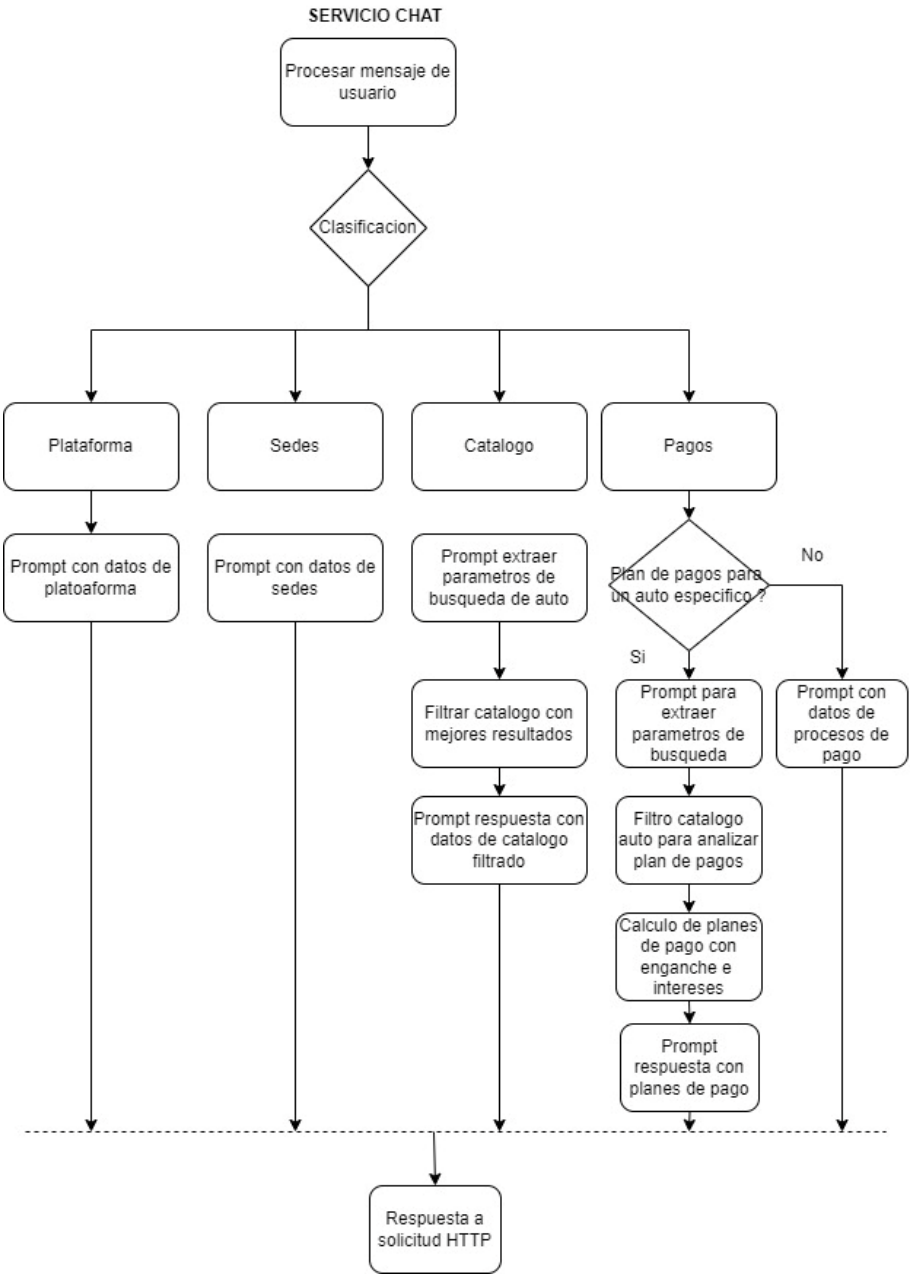
Para consultas de catalogo se realizan los siguientes pasos:

1. Extraer parámetros de búsqueda: Para simplificar se extrae la información para la búsqueda de un auto la cual puede ser modelo, marca, año, precio máximo, fabricante.
2. Filtrar los datos del catálogo con los parámetros de búsqueda reduciendo los datos para utilizar en el siguiente prompt.
3. Utilizar un prompt para encontrar la mejor coincidencia para la solicitud del usuario y redactar una respuesta.

El flujo de pagos tiene dos casos, cuando se hace una pregunta genérica sobre procesos de pagos y cuando se pregunta algo mas especifico sobre el plan de financiamiento de un auto. El caso genérico se resuelve con un prompt que incluye datos de pagos del sitio web.

El caso de calculo de planes de financiamiento sigue los mismos pasos del catalogo, pero una vez obtenido el auto se generan los planes de pago y se redacta una respuesta con esta información.

Los planes de pago se calculan en rango de 3 a 6 años, todos los cálculos de interés se hacen proceduralmente para evitar cambios de precios o errores de aritmética.


SERVICIO CHAT
Procesar mensaje de usuario
Clasificacion
Plataforma
Sedes
Catalogo
Pagos
Prompt con datos de platoaforma
Prompt con datos de sedes
Prompt extraer parametros de busqueda de auto
Plan de pagos para un auto especifico ?
No
Si
Filtrar catalogo con mejores resultados
Prompt para extraer parametros de busqueda
Prompt con datos de procesos de pago
Prompt respuesta con datos de catalogo filtrado
Filtro catalogo auto para analizar plan de pagos
Calculo de planes de pago con enganche e intereses
Prompt respuesta con planes de pago
Respuesta a solicitud HTTP

Propuesta de roadmap

1. Objetivos del Proyecto

Desarrollar un bot comercial basado en LLMs que simule el comportamiento de un agente comercial de Kavak.

Ofrecer capacidades clave:

- Responder preguntas sobre la propuesta de valor de Kavak.
- Minimizar alucinaciones dentro del chatbot.
- Manejar interacciones en lenguaje natural, incluso con errores en la redacción.
- Brindar recomendaciones de autos disponibles.
- Otorgar planes de financiamiento basados en datos proporcionados por el cliente.

2. Roadmap (12 Semanas)

2.1 Fase de Planificación (Semanas 1-2)

- Definición de Requerimientos:
- Detallar todos los casos de uso del chatbot.
- Especificar las capacidades y limitaciones de los LLMs en relación con los objetivos del proyecto.

Análisis de riesgos y limitaciones:

- Identificar riesgos técnicos como la falta de datos, posibles errores derivados de parseo de datos, y problemas en la integración con otros sistemas.
- Proponer estrategias de mitigación, como el uso de datos mock o pruebas con datos limitados inicialmente.
- Considerar escenarios para uso de historial de mensajes por sesion, por ejemplo reglas para poder reusar contexto de mensajes anteriores durante la sesión del chat.
- Considerar preguntas que no tienen soporte y como gradualmente se podrían incluir en el sistema (Por ejemplo nuevas secciones, preguntas frecuentes, etc)
- Considerar datos que no se incluyen en el POC para dar claridad sobre alcance a cubrir en la version inicial considerando viabilidad técnica y capas de construcción del sistema.

Planificación del Entorno de Desarrollo:

- Definir el entorno de desarrollo, incluyendo herramientas necesarias, frameworks, y configuraciones de CI/CD.
- Planificar la configuración inicial de servidor con API de primera versión para iniciar un POC.

Preparación de Datos:

- Estructurar los datos del catálogo en JSON schemas para iniciar el desarrollo, definir uso de bases de datos o de otros servicios web existentes.
- Configurar el entorno necesario para leer y procesar los datos del catálogo y otras entidades que se puedan necesitar.

2.2 Backlog de Desarrollo y Pruebas (Semanas 3-8)

Desarrollo de la API y Clasificación de Mensajes:

- Implementación de la API en Python para manejar solicitudes y respuestas.
- Desarrollo del módulo de clasificación de mensajes utilizando técnicas de procesamiento de lenguaje natural (NLP).
- Crear pruebas unitarias para asegurar la correcta funcionalidad de la API y la clasificación de mensajes.

Desarrollo de Consultas de Catálogo y Planes de Financiamiento:

- Implementación de la funcionalidad para consultas de catálogo.
- Desarrollar el módulo de cálculo de planes de financiamiento y su integración con las consultas del catálogo.

Pruebas de Integración:

- Probar la interacción entre los módulos (API, clasificación, consultas de catálogo, generación de planes de financiamiento).

Optimización y Refinamiento:

- Refinar el manejo de prompts para reducir alucinaciones y cubrir casos con diferentes tipos de mensajes del usuario.
- Optimizar la carga de datos del catálogo para manejar grandes volúmenes de autos.

Pruebas de Escalabilidad y Desempeño:

- Simular consultas al catálogo con un número creciente de autos para asegurar que el sistema puede manejar grandes volúmenes de datos.
- Realizar pruebas de carga para evaluar el rendimiento del sistema bajo condiciones de uso intensivo. Considerar limitaciones en el rate del API de openAo (tokens por minuto)
- Para validar que una version no tiene retroceso es necesario evaluar nuevamente todo el batch de solicitudes con la version candidata a usar. Se puede utilizar un prompt para evaluar automáticamente el resultado y generar una métrica de calidad de la version.

2.3 Fase de Despliegue (Semanas 9-10)

Preparación del Entorno de Producción:

- Configurar servidores para el despliegue de la API, considerando costos y eficiencia (uso de VMs o serverless computing) definir proveedor de servicios (Ej Microsoft, Google, AWS, etc).
- Configurar el sistema de monitoreo y alertas para el entorno de producción.

Despliegue en Staging:

- Desplegar el bot en un entorno de staging para realizar pruebas finales antes de la implementación en producción.
- Ejecutar pruebas integrales y pruebas de usuarios clave en el entorno de staging.

Despliegue en Producción:

- Realizar un despliegue controlado para verificar el comportamiento en producción antes de escalar (elegir grupo de control para version beta que pueda dar retroalimentación útil en etapa temprana)
- Monitorear de cerca las métricas de desempeño y resolver cualquier problema que surja inmediatamente después del despliegue.

2.4 Fase de Mejora Continua y Mantenimiento (Semanas 11-12)

Monitoreo Post-Despliegue:

- Revisar las métricas de uso y desempeño del bot en producción.
- Ajustar y optimizar el sistema basado en los datos recolectados.
- Recolectar feedback de usuarios finales y stakeholders para identificar áreas de mejora.
- Preparar el roadmap para futuras iteraciones que incluyan nuevas funcionalidades y mejoras basadas en el feedback y análisis post-lanzamiento.

Ciclo de Iteración y Mejoras:

- Implementar mejoras basadas en el feedback recibido.
- Preparar el backlog para la siguiente iteración de desarrollo.
- Completar la documentación del proyecto, incluyendo manuales de usuario, documentación técnica, y reportes de desempeño.
-

Guia de instalación

1. El código se desarrolló con la versión 3.11 de Python
2. Las librerías adicionales se incluyen en requirements.txt, las cuales se instalan usando pip3.11 install -r requirements.txt
3. Agrega en la misma carpeta donde esta el archivo README un archivo config.json con un diccionario que tenta una llave { “openaikey”:”TOKEN”} que tenga como valor el token a usar con el API de openAI
4. Para correr la aplicación se corre python3.11 autoreloader.py
5. Se puede correr usando el navegador http://localhost:8000 o con POSTMAN, se aceptan solicitudes GET con parámetro “text” enviado en el URL o POST con parámetro ”text” enviado en un JSON body