Asistente conversacional para restaurantes

Caso técnico de portfolio | IA aplicada a una carta digital, recomendaciones y datos reales

Resumen. MVP de asistente conversacional integrado en la web de un restaurante. Permite consultar la carta, resolver dudas sobre ingredientes, alérgenos, precios y preferencias dietéticas, y recibir recomendaciones personalizadas. La solución combina PostgreSQL, FastAPI, Docker, un frontend web y Groq. La IA no responde desde conocimiento genérico: trabaja con información recuperada desde el backend para reducir respuestas inventadas y mantener la carta como fuente de verdad.

Tipo	MVP académico	**Sector**	Restauración
Stack	FastAPI · PostgreSQL · Docker · Groq	**Frontend**	HTML · CSS · JavaScript
Rol	Desarrollo full stack dentro de un proyecto en equipo	**Enfoque**	Datos, API, IA y experiencia web

CONTEXTO Y PROBLEMA

Muchos restaurantes tienen cartas digitales estáticas. El cliente puede ver los platos, pero no siempre encuentra rápido información sobre ingredientes, alérgenos, opciones vegetarianas, precios o recomendaciones. Esta fricción aumenta cuando la carta es extensa, cuando hay restricciones alimentarias o cuando el usuario no domina el idioma del local.

El proyecto plantea una alternativa más directa: un asistente integrado en la web del restaurante al que el cliente puede preguntar de forma natural. En lugar de recorrer toda la carta, escribe lo que necesita y recibe una respuesta adaptada a la información real del negocio.

OBJETIVO DEL PROYECTO

El objetivo fue construir una demo funcional que uniera web, backend, base de datos e inteligencia artificial en un flujo completo. No se buscaba lanzar una plataforma comercial cerrada, sino validar una base técnica reutilizable para restaurantes que quieran mejorar su atención digital sin desarrollar una solución desde cero.

El sistema debía consultar platos, ingredientes, alérgenos, etiquetas dietéticas e información nutricional disponible. También debía generar recomendaciones y responder con prudencia cuando un dato no estuviera registrado.

FUNCIONALIDADES PRINCIPALES

- Carta conectada al backend, organizada por categorías y platos.
- Consulta de ingredientes, alérgenos, etiquetas dietéticas, precios y detalles de plato.
- Widget de chat integrado en la web del restaurante.
- Recomendaciones según gustos, categoría, precio, alergias o preferencias alimentarias.
- Cambio inicial de idioma entre castellano e inglés.
- Sección visual de reserva, preparada como punto de entrada para una integración futura.

FLUJO DE USUARIO

El usuario entra en la web, consulta la carta o abre el chat y escribe una pregunta. El frontend envía el mensaje al endpoint de chat del backend. FastAPI recupera la información necesaria desde PostgreSQL, prepara un contexto controlado y lo entrega a Groq para redactar la respuesta. La respuesta vuelve al frontend y se muestra en el widget.

Flujo resumido: Usuario -> Frontend -> FastAPI -> PostgreSQL -> Groq -> Respuesta en el chat.

ARQUITECTURA TÉCNICA

La arquitectura se organizó en capas para separar responsabilidades. El frontend muestra la experiencia visual y no almacena claves privadas ni lógica sensible. El backend centraliza las peticiones, los filtros, la comunicación con la base de datos y la integración con Groq. PostgreSQL actúa como fuente de verdad del menú. La capa de IA convierte los datos recuperados en una respuesta natural y fácil de entender.

TECNOLOGÍAS UTILIZADAS

PostgreSQL se eligió por la necesidad de modelar relaciones entre platos, ingredientes, alérgenos, etiquetas dietéticas y datos nutricionales. Docker permitió levantar la base de datos de forma reproducible. FastAPI facilitó la creación de endpoints REST, validación de datos y pruebas con Swagger. Groq aportó la generación de lenguaje natural, mientras que HTML, CSS y JavaScript fueron suficientes para construir una demo web ligera y conectada a la API.

IA, RAG Y CONTROL DE RESPUESTAS

La integración de IA se diseñó con un enfoque tipo RAG: primero se recupera información real del backend y después se utiliza como contexto para que el modelo genere la respuesta. El modelo no decide la carta ni inventa datos; redacta a partir de la información que recibe.

Este punto fue esencial para evitar respuestas inventadas. Si el usuario pregunta por un plato, un ingrediente o un alérgeno, el sistema consulta la base de datos antes de responder. Si la información no está disponible, el asistente debe indicarlo de forma prudente en lugar de confirmar algo que no puede verificar.

GESTIÓN DE PLATOS, INGREDIENTES Y ALÉRGENOS

El modelo de datos permite reutilizar información y mantener consistencia. Un plato pertenece a una categoría, puede tener varios ingredientes y cada ingrediente puede estar asociado a uno o varios alérgenos. Los platos también pueden incluir etiquetas dietéticas, como vegetariano o vegano, además de información nutricional cuando esté disponible.

Esta estructura permite filtrar, buscar y recomendar con más precisión. El asistente puede responder preguntas como qué opciones son vegetarianas, qué platos contienen ciertos ingredientes o qué alternativas encajan con una restricción alimentaria concreta.

RETOS TÉCNICOS

El reto principal fue hacer que la IA resultara útil sin dejarla responder de forma libre. Para conseguirlo, la base de datos se trató como fuente de verdad y el backend controló qué contexto llegaba al modelo. Otro reto fue conectar correctamente las capas del sistema: una misma API debía servir tanto a la carta visual como al bot conversacional.

También fue importante mantener el frontend organizado a medida que crecían la carta, el cambio de idioma, la sección de reservas y el widget de chat.

DECISIONES TÉCNICAS RELEVANTES

- Construir primero la base de datos y la API antes de integrar la IA.
- No exponer claves privadas en el frontend.
- Usar PostgreSQL en lugar de depender de prompts o texto plano.
- Separar la generación de lenguaje natural de la consulta real de datos.
- Diseñar el asistente para responder con cautela ante alérgenos e información nutricional.

RESULTADO Y APRENDIZAJE

El resultado fue una demo funcional de restaurante con carta conectada, asistente conversacional, backend en FastAPI, base de datos PostgreSQL y respuestas generadas con Groq a partir de datos reales. El proyecto demuestra un flujo completo de producto: el usuario entra en una web, consulta la carta, pregunta al asistente y recibe una respuesta coherente con el menú registrado.

El aprendizaje más importante fue entender que un chatbot útil no depende solo del modelo de IA. Necesita datos bien estructurados, una API fiable, reglas claras y una arquitectura que limite el contexto que puede usar. El proyecto reforzó conocimientos de backend, bases de datos, APIs REST, Docker, frontend e integración de servicios externos.

LIMITACIONES Y MEJORAS FUTURAS

Actualmente el proyecto funciona como MVP local. La reserva no está automatizada de principio a fin, el soporte multilingüe se limita inicialmente a castellano e inglés y todavía no existe un panel de administración para que el restaurante gestione su carta de forma autónoma.

Como evolución, se podrían añadir despliegue online, base de datos en la nube, panel de administración, reservas automatizadas, más idiomas, analítica de preguntas frecuentes, memoria conversacional corta y control del consumo del modelo.

VALOR PARA UNA EMPRESA

Este caso demuestra capacidad para conectar varias áreas de desarrollo en una solución coherente: diseño de datos, API, frontend, integración de IA y enfoque de producto. No es solo un chatbot: es una arquitectura pensada para que la IA trabaje con información real de un negocio.

- Integración de IA con datos estructurados.
- Arquitectura con FastAPI, PostgreSQL, Docker y Groq.
- Aplicación práctica a un problema real del sector restauración.
- Control de respuestas inventadas mediante recuperación de contexto.
- Capacidad para convertir una idea en un MVP funcional y demostrable.