# Informe Técnico - Mini Bot Conversacional

## 1. Contexto de la Solución

El chatbot desarrollado cumple estrictamente con los requerimientos funcionales y no funcionales solicitados en la evaluación, implementando un sistema conversacional basado en Flask y SQLite para persistencia de datos, con comunicación REST segura mediante HTTPS. Se priorizó la simplicidad escalable, evitando frameworks preconstruidos para demostrar capacidad de desarrollo desde cero.

## 2. Cumplimiento de Requisitos

|  |  |
| --- | --- |
| Requerimiento | Implementación |
| 10+ consultas | Base de datos SQLite con 10 pares pregunta-respuesta iniciales |
| Interfaz web embebida | Ventana de chat responsive en esquina inferior derecha (index.html + CSS/JS) |
| Respuesta en <5 segundos | Lógica optimizada con normalización de texto (responses.py) |
| Persistencia de datos | Tabla en SQLite + servicio REST (/chat POST) con Flask |
| Seguridad HTTP | HTTPS con certificados autofirmados (OpenSSL) y validación básica |

## 3. Decisiones Técnicas Clave

- Arquitectura modular: separación en backend (Flask), frontend (HTML/CSS/JS) y base de datos (SQLite).

- Normalización de texto: elimina tildes y convierte a minúsculas para mejorar coincidencias (función normalizar() en responses.py).

- Escalabilidad: diseño preparado para migrar a MySQL/PostgreSQL y agregar autenticación JWT.

## 4. Limitaciones y Mejoras

Actuales:

* - No maneja conversaciones multi-turno.
* - Certificado SSL autofirmado (en producción se recomienda Let's Encrypt).

Futuras:

* - Integrar NLP con spaCy para mejor comprensión semántica.
* - Cachear respuestas frecuentes en Redis para reducir latencia.

## 5. Proceso de Demo

Ejecución local:

python app.py # Inicia servidor en https://localhost:5000

Flujo de interacción:

1. Usuario escribe “¿Cuál es su horario?” → Bot responde con datos de la base de datos.

2. Usuario pregunta “¿Hacen envíos?” → Bot indica que no entendió y sugiere temas relacionados.

## 6. Puntos a Destacar en la Entrevista

* - Mantenibilidad: código documentado y separación clara de capas (frontend/backend/DB).
* - Portabilidad: incluye Dockerfile para despliegue en cualquier entorno.
* - Seguridad: headers HTTP configurados (CORS, no-sniff) y sanitización de inputs.