AUDITORIA_EXAMEN_3

Informe de Auditoría: Corporate EPIS Pilot

Repositorio GitHub: [https://github.com/BCZLopezCatunta/AUDITORIA_EXAMEN_3.git] **Auditor:** [Brayar Christian Lopez Catunta] **Fecha:** 19/11/2024

---

1. Resumen Ejecutivo

El presente documento certifica la auditoría técnica realizada al sistema **"Corporate EPIS Pilot"**, un Asistente de IA conversacional para entornos corporativos. La auditoría se centró en verificar la migración exitosa del motor de inferencia al modelo ligero `smollm:360m`, la integridad del flujo de datos RAG (Retrieval-Augmented Generation) y la correcta persistencia de incidentes en la base de datos, operando bajo una arquitectura de microservicios orquestada.

2. Objetivos de la Auditoría

2.1 Objetivo General

Validar la fiabilidad operativa, la integridad de los datos y el cumplimiento de los requisitos funcionales del sistema de Mesa de Ayuda con IA, asegurando que la integración del modelo `smollm:360m` soporte adecuadamente el flujo de atención al usuario y la escalación a tickets de soporte.

2.2 Objetivos Específicos

1. **Verificar la Inferencia y Lógica del Modelo (smollm:360m):** Evaluar si el modelo seleccionado, ejecutado vía Ollama, es capaz de clasificar intenciones y proporcionar respuestas coherentes basadas en la documentación interna sin alucinaciones críticas.
2. **Validar el Flujo de Negocio "RAG a Ticket":** Comprobar que el sistema prioriza la solución automática mediante búsqueda vectorial (ChromaDB) y, ante fallos en la resolución, ejecuta correctamente la acción `ACTION_CREATE_TICKET`.
3. **Auditar la Persistencia e Integridad de Datos:** Certificar que los tickets generados desde la interfaz web se registran correctamente en la base de datos relacional (`tickets.db`), manteniendo la consistencia de los campos (descripción, estado).
4. **Evaluar la Arquitectura y Despliegue:** Verificar que el entorno contenerizado (Docker/Docker Compose) levanta correctamente los servicios de Frontend, Backend y Proxy, asegurando la disponibilidad del sistema en el puerto 5173.

3. Alcance y Metodología

3.1 Alcance

- **Backend:** API desarrollada en FastAPI (Python).
- **Frontend:** Interfaz de usuario en React + TypeScript.
- **Infraestructura:** Orquestación Docker, Proxy Nginx y servicio de IA Ollama.

- **Datos:** Base de datos SQLite y Vector Store ChromaDB.

3.2 Metodología

Se aplicó una metodología de pruebas híbrida:

- **Pruebas de Caja Negra:** Interacción con la UI (`localhost:5173`) simulando un usuario final para validar flujos de éxito y error.
- **Pruebas de Caja Blanca:** Inspección de código fuente (`main.py`), logs del servidor y consultas directas a la base de datos mediante scripts de verificación.

---

4. Evidencias y Hallazgos

A continuación, se documentan las evidencias que respaldan el cumplimiento de los objetivos planteados. Las capturas de pantalla originales se encuentran en la carpeta `/evidencias` de este repositorio.

📋 Hallazgo 1: Despliegue de Infraestructura

Descripción: Se verificó mediante CLI de Docker que todos los microservicios (Backend, Frontend, Proxy) se encuentran en estado `UP` y operando sin reinicios inesperados.

- **Estado:** ✅ CUMPLIDO
- **Referencia:** `evidencias/evidencia_1_docker.png`

🧠 Hallazgo 2: Capacidad RAG (Retrieval-Augmented Generation)

Descripción: El sistema fue capaz de responder preguntas específicas sobre la documentación corporativa (ej. "Política de Teletrabajo"), demostrando que la ingestión de documentos y la recuperación vectorial funcionan con el modelo `smollm:360m`.

- **Estado:** ✅ CUMPLIDO
- **Referencia:** `evidencias/evidencia_2_rag.png`

🎫 Hallazgo 3: Escalación y Creación de Tickets

Descripción: Se simuló un incidente técnico crítico no resuelto por la IA. El sistema detectó la negativa del usuario y habilitó la creación del ticket. El Frontend mostró la confirmación con el ID del ticket generado.

- **Estado:** ✅ CUMPLIDO
- **Referencia:** `evidencias/evidencia_3_ticket_creado.png`

💾 Hallazgo 4: Integridad de Base de Datos

Descripción: Se realizó una consulta SQL directa a la tabla `tickets` en el archivo `tickets.db`. Se constató que el ticket creado en el paso anterior fue registrado con la descripción correcta y estado "Abierto".

- **Estado:** ✅ CUMPLIDO
- **Referencia:** `evidencias/evidencia_4_bd.png`

---

5. Análisis Técnico

5.1 Modelo de Lenguaje

La implementación de `smollm:360m` ha demostrado ser eficiente para entornos locales de bajos recursos. Aunque su capacidad de razonamiento es menor que modelos más grandes (Llama 3), la adaptación de los *prompts* en el backend permitió mantener la funcionalidad del Router de intenciones.

5.2 Instrumentación

El sistema cuenta con métricas expuestas vía `prometheus-fastapi-instrumentator` en la ruta `/metrics`, y un sistema de logging estructurado con `Loguru`, facilitando la auditoría de errores en tiempo real.

---

6. Conclusiones

Tras la ejecución de las pruebas y el análisis de evidencias, se concluye lo siguiente:

1. **Funcionalidad Total:** El sistema "Corporate EPIS Pilot" cumple con el 100% de los requisitos funcionales estipulados para el examen.
2. **Adaptabilidad:** La arquitectura demostró ser flexible al permitir el cambio del modelo fundacional a `smollm:360m` sin romper la lógica de negocio.
3. **Robustez:** El mecanismo de persistencia en SQLite y la orquestación con Docker Compose/Nginx aseguran un despliegue estable y replicable.