Memoria de Dificultades Atopadas, Solucións Propostas e Referencias

Funcionalidades Pendentes:

Carga de CSV ou tickets dixitais: Está pendente que funcione cargar arquivos CSV ou tickets dixitais (sin modelos) directamente dende a interface web.

Restauración de conta e verificación en dous pasos: A funcionalidade está prevista para reforzar a seguridade do sistema, mais aínda non se implementou (Redirixida a formularios de contacto temporalmente).

Contedorización con Docker: Non se logrou integrar todo o sistema en contedores Docker por problemas de configuración e dependencias cruzadas.

Dificultades Atopadas e Solucións Propostas:

- Backend e Infraestrutura:

  Problemas coas datas entre SQL e Spring Boot: Houbo desaxustes ao interpretar formatos de data/hora entre a base de datos e o backend. Solucionouse engadindo conversores específicos e controlando o uso de @Temporal e @JsonFormat.

  Eliminación dunha segunda base de datos en PostgreSQL: Intentouse usar dúas bases, pero a configuración foi complexa e a instalación de PostgreSQL resultaba innecesaria. Decidiuse manter unha única base.

  Inicialización da base de datos: O inicializador lanzaba erros por falta de datos e estrutura. Engadíronse parámetros adicionais e validacións para estabilizalo.

  Implementación de JWT e OAuth: Foi unha das partes máis complicadas. Supuxo revisar moitos repositorios e titoriais para comprender como combinar Spring Security con JWT e OAuth.

  Uso de key compartida en API Flask: No canto dun sistema de login completo, optouse por empregar unha API key compartida para autenticar os usuarios e limitar o acceso aos endpoints, simplificando a implementación.

  Swagger en Python: Intentouse usar Flasgger para a documentación automática da API Flask, pero o resultado visual non era axeitado. Aplicouse Sphinx.

- *Frontend e Experiencia de Usuario*

  Responsive design con Bootstrap: A interface tiña que ser responsiva, pero non se adaptaba ben a móbiles. Empregáronse clases Bootstrap con trucos manuais para conseguir un comportamento máis fluído.

  Problemas co dashboard que non se actualizaba: O dashboard non mostraba datos en tempo real. Engadiuse un temporizador para refrescar a información, pero queda o problema de que realiza chamadas constantes mentres o usuario admin está conectado.

  Gráficas en Angular: Ao principio non se comprendía ben como pasar os parámetros dinámicos aos gráficos. Unha vez entendido, foi sinxelo integrar Chart.js e outras librerías para visualización.

- *Outros*

  Problemas coa IA local (OCR): O recoñecemento de texto con Tesseract non foi moi preciso en moitos casos. A solución parcial foi combinar o OCR con interpretación mediante IA.

  Errores tipográficos e solucións manuais: A IA Copilot foi útil para detectar erros tipográficos ou de sintaxe.

  Moitas cousas que me ensinaro a facer automaticas , fixenas a man. Pendiente de mellorar..

- *Estratexias e Axustes Técnicos*

  Uso de almacenamento local para ficheiros e datos temporais (Paso de AWS).

  Asistencia da IA para solución de erros, comentar de código e mellora de este.

  Simplificación do backend de Flask mediante API key no canto de autenticación complexa.

  Aplicación de boas prácticas de commits e ramas en Git tras problemas iniciais.

  Probas manuais exhaustivas e revisións de logs para mellorar a estabilidade do sistema.

- *Referencias Técnicas e Bibliográficas*

  Spring Boot Docs: https://spring.io/projects/spring-boot

  Angular Docs: https://angular.io/docs

  Bootstrap 5: https://getbootstrap.com/docs/5.3/getting-started/introduction/

  JWT y OAuth en Spring: https://www.baeldung.com/spring-security-oauth-jwt

  Tesseract OCR: https://github.com/tesseract-ocr/tesseract

  Flasgger (Swagger para Flask): https://github.com/flasgger/flasgger

  Chart.js con Angular: https://www.chartjs.org/docs/latest/

  StackOverflow, YouTube (Amigoscode, Java Brains, etc.)

  Asistencia continua con OpenAI ChatGPT