
**FACULTAD DE INGENIERÍA, ARQUITECTURA Y DISEÑO**

**BASES DE DATOS II**

**ESTUDIANTE / CIP:**

**AARON NEWBALL / 8-1004-1287**

**ERIC MURILLO / 8-978-932**

**FECHA:**

**18/12/2025**

**TEMA:**

**PROYECTO VIVENCIAL**

**PROFESOR:**

**MARYON TORRES**

# Proyecto Final (Resumen Ejecutivo)

## 1) Presentación y datos generales

- **Nombre del proyecto:** Sistema de Expedientes Legales
- **Público objetivo:**
  - Asistentes legales y administradores de bufetes
  - Abogados, aseguradoras y personal de juzgados
  - Personal administrativo que requiere consulta y mantenimiento de expedientes
- **Fecha:** 18 de diciembre de 2025

**Plataforma de trabajo** - Lenguaje/Framework: Python + Flask - Base de datos: MySQL/MariaDB - Contenerización: Docker + Docker Compose - Control de versiones y colaboración: Git + GitHub - Herramientas de desarrollo: VS Code

**Artefactos clave (evidencia en repositorio):**

- **Scripts SQL (4):** data/sql/01_create_db.sql, 02_create_tables.sql, 03_initial_data.sql, 04_views.sql

- **API:** carpeta api/ (módulos para auth, expedientes y catálogos)

- **Tests:** tests/test_auth.py, tests/test_create_case.py

- **Docker:** Dockerfile, docker-compose.yml

## 2) Resumen ejecutivo

Sistema web (Flask) para centralizar expedientes legales y entidades relacionadas, con control de acceso por roles, API REST y UI con plantillas.

**Estado actual (según README)** - Core estable: v0.9.11 - Implementado: autenticación con roles, CRUD expedientes, catálogos, agenda diaria, caché, scripts de carga de BD, Docker, tests básicos - Pendiente: CRUD completo de conductores, dashboard UI, reportes/exportación, migraciones, adjuntos, notificaciones, auditoría

**Valor entregado** - Centralización y trazabilidad de información de expedientes - Operación reproducible (local o Docker) con inicialización de BD por scripts

**Próximos pasos (prioridad ejecutiva)**

1) Completar CRUD conductores

2) Implementar dashboard UI (las métricas backend ya existen)

3) Añadir reportes/exportación y ampliar pruebas

## 3) Objetivos

### 3.1 Objetivo general

Desarrollar un sistema web para administrar expedientes legales y sus entidades asociadas, con control de acceso por roles, API REST y una interfaz de usuario basada en plantillas, respaldado por una base de datos MySQL/MariaDB.

### 3.2 Objetivos específicos

- Implementar CRUD de expedientes con paginación y filtrado, y exponerlo también vía API REST.
- Implementar CRUD de catálogos (aseguradoras, abogados, juzgados, conductores) y pantallas de gestión.
- Proveer autenticación con roles y endpoints bajo /api con contrato JSON estable.
- Permitir inicialización reproducible de la BD mediante scripts SQL y/o cargador automático, incluyendo vistas/procedimientos para métricas.

## 4) Alcance del proyecto

### 4.1 Alcance (Incluye)

- Gestión de expedientes: crear/consultar/editar/eliminar (CRUD).
- Gestión de catálogos: aseguradoras, abogados, juzgados, conductores.
- Agenda diaria con filtrado por fecha.
- Autenticación con roles (administrador, asistente legal).
- API REST bajo /api.
- Carga de esquema/datos mediante scripts SQL.
- Ejecución local y con Docker Compose.
- Pruebas básicas automatizadas con pytest.

### 4.2 Fuera de alcance (Excluye)

- Migraciones de esquema automatizadas (por ahora el esquema se versiona con scripts en data/sql/).
- Landing Page
- Gestión de documentos adjuntos.
- Sistema de notificaciones.
- Auditoría de cambios.
- Reportes y exportación (PDF/Excel) y dashboard UI (pendientes).

## 5) Cronograma con hitos principales

| Hito | Descripción | Fecha planificada | Evidencia |
|---|---|---|---|
| H1 | Diseño e implementación de BD (tablas + relaciones) | Semana 1 (24–30 nov 2025) | data/sql/02_create_tables.sql |
| H2 | Carga inicial de datos | Semana 1 (24–30 nov 2025) | data/sql/03_initial_data.sql |
| H3 | API REST (expedientes + auth) | Semana 2 (01–07 dic 2025) | api/ (p. ej. api/auth.py, api/cases.py) |
| H4 | Presentación (vistas + templates + estáticos) | Semana 2–3 (01–14 dic 2025) | presentation/, templates/, static/ |
| H5 | Dockerización (app + db) | Semana 3 (08–14 dic 2025) | Dockerfile, dockercompose.yml |
| H6 | Pruebas unitarias básicas | Semana 3 (08–14 dic 2025) | tests/ |
| H7 | Landing page | Semana 4 (15–18 dic 2025) | Estructura base presente (por evidenciar con capturas/URL) |
| H8 | Prototipo plataforma final | Semana 4 (15–18 dic 2025) | Demo local/Docker + capturas/links (por definir) |

---

## 6) Solución propuesta

### 6.1 Problema identificado

La gestión de expedientes y entidades asociadas suele estar dispersa, dificultando trazabilidad, control de acceso y consulta rápida; además limita métricas/reportes.

### 6.2 Propuesta de solución (basada en el análisis)

- Implementar un sistema web para centralizar expedientes jurídicos y entidades relacionadas.
- Incluir autenticación y roles para control de acceso.
- Exponer API REST para operaciones CRUD y facilitar integración futura.

- Implementar agenda para gestión de actividades asociadas.

### 6.3 Arquitectura (resumen)

- **Repositorios (repositories/)**: acceso a datos y consultas SQL (pool de conexiones).
- **Servicios (services/)**: lógica de negocio, validaciones y caché.
- **API REST (api/)**: endpoints JSON bajo /api usando Flask blueprints.
- **Presentación (presentation/)**: rutas de vistas + plantillas Jinja y archivos estáticos.

---

## 7) Metodología

Iterativa e incremental por módulos (BD → API → UI → Docker → tests), con versionamiento semántico y validación continua.

- **Planificación:** Definición de alcance por módulos (expedientes, catálogos, auth, agenda) y artefactos clave (scripts SQL, API, UI).
- **Ejecución:** Implementación por capas (repositorios/servicios/endpoints/vistas), integrando Docker y scripts de inicialización.
- **Control:** Pruebas automatizadas con pytest (auth y CRUD "happy path"), más verificación manual local/Docker.
- **Cierre:** Funcionalidades core operativas, guía de instalación reproducible, y tests básicos ejecutables en local y contenedor.

---

## 8) Listado de entregables

- PDF Parte I (Informe).
- PDF Parte II (Proyecto Final).
- Scripts SQL: creación de BD/tablas/datos iniciales (y vistas/procedimientos si aplica).
- Código fuente: backend (Flask), API, servicios, repositorios.
- Frontend: templates + estáticos.
- Dockerfile + docker-compose para entorno reproducible.
- Tests básicos con pytest.
- Evidencias: capturas (UI/API/BD), links de commits, y video demo.

**Entrega de presentación (pendiente de adjuntar en este documento)** - Video demo + capturas clave (login, expedientes, catálogos, agenda, API JSON, BD)

## 9) Inversión (parte económica)

### 9.1 Supuestos de costo

- Estimación simulada (referencial) en USD, tomando como base rangos "convencionales" del mercado panameño para desarrollo de software.
- No es cotización formal: los valores pueden variar por seniority, urgencia, alcance final y modalidad (freelance/empresa).
- Supuesto de proyecto tipo MVP académico ejecutado en 4 semanas.
- Herramientas open-source (sin costos de licencias).

**Tarifas referenciales (Panamá, USD/h) — usadas para el cálculo**

- Desarrollo (Full-Stack / Backend): $25–35

- QA / Pruebas: $15–25

- DevOps/Soporte de despliegue: $20–35

- Gestión/Análisis (PM/BA): $20–35

Para este cálculo se usa un punto medio conservador.

### 9.2 Estimación de costos (tabla)

| Rubro | Descripción | Cantidad | Costo unitario | Total |
|---|---|---|---|---|
| Gestión/Análisis | Levantamiento + planificación + definición de alcance | 15 h | 25 | $375 |
| Desarrollo | Implementación backend + UI + integración BD | 120 h | 30 | $3,600 |
| Pruebas (QA) | Ejecución de tests + casos "happy path" + verificación Docker | 25 h | 18 | $450 |
| DevOps/Soporte | Ajustes Docker/variables/ejecución en entorno reproducible | 10 h | 25 | $250 |
| Infraestructura | Desarrollo local/Docker (sin hosting) | 1 | 0 | 0 |
| Dominio | No impementado | 1 | 0 | 0 |

| | | | | |
|---|---|---|---|---|
| Otros | Licencias (open-source)<br><br>**Subtotal (mano de obra):** $4,675<br><br>**Contingencia (10%):** $468<br><br>**Total estimado (simulado):** $5,143<br><br>**Fuera de Rango:**<br><br>- VPS básico $10–25<br><br>- Dominio anual: $12–20 | 1 | 0 | 0 |

## 10) Lecciones aprendidas y otra información relevante

### 10.1 Lecciones aprendidas

- La arquitectura en capas facilita pruebas y mantenimiento, pero requiere disciplina en contratos entre capas (servicio/repositorio/API).
- Docker Compose acelera la reproducibilidad de la demo, siempre que se documenten variables de entorno y pasos de inicialización.
- Versionar el esquema con scripts SQL es simple para un curso, pero a futuro conviene migraciones (pendiente) para evoluciones controladas.

**Riesgos / dependencias (ejecutivo)**

- Dependencia de variables de entorno correctas para conexión a BD (local/Docker).

- Scripts SQL requieren mantenerse alineados con el esquema cuando se agreguen cambios.

### 10.2 Link del repositorio

- https://github.com/anxwball/Sistema-Expedientes-Legales

### 10.3 Contribuciones de cada miembro

Fuente: historial de git del repositorio.

| Integrante | Rol | Tareas / contribuciones principales | Evidencia | Estado |
|---|---|---|---|---|
| Aaron Newball | Desarrollo principal | Backend Flask, scripts SQL/cargador, Docker, tests, UI (Jinja/CSS/JS) | Commits (63) + estructura del repo | Completado (core) |

| | | | | |
|---|---|---|---|---|
| Eric Murillo | Investigacion de Informacion | Buscar informacion relacionada a costos y panorama de mercado | estructura de inversión para un proyecto tipo MVP académico | En investigación con personas |