UAX FP
Universidad
Alfonso X el Sabio

Plataforma de Análisis de Binarios Ejecutables

2º / DAM

Alumno: Mario Asenjo Pérez

Tutor: Antonio Reinoso Peinado

UAX FP
Universidad
Alfonso X el Sabio

UAX FP
Universidad
Alfonso X el Sabio

DEDICATORIA (OPCIONAL)

UAX FP
Universidad
Alfonso X el Sabio

ÍNDICES

De contenido, tablas e ilustraciones. Se recomienda realizarlos de manera automática.

UAX FP
Universidad
Alfonso X el Sabio

Abstract (Español)

La fase de pre-explotación constituye un paso esencial en una auditoría de seguridad ofensiva, ya que permite identificar y comprender la superficie de ataque de una aplicación antes de intentar cualquier explotación, en muchos entornos reales esta fase se apoya en herramientas aisladas y procesos manuales que generan información dispersa y difícil de correlacionar.

Este proyecto propone el desarrollo de una plataforma de pre-explotación y análisis de binarios, orientada a centralizar y estructurar el análisis inicial de ejecutables, con especial foco en binarios ELF sobre sistemas Linux. La solución combina análisis estático y análisis dinámico controlado, permitiendo obtener información relevante sobre la estructura del binario, sus mitigaciones de seguridad y su comportamiento durante la ejecución.

La plataforma se concibe como un producto extensible, basado en una arquitectura desacoplada que integra un backend desarrollado en Java mediante Spring Boot, una interfaz web y workers especializados para la ejecución de los análisis. Los resultados se almacenan de forma persistente utilizando PostgreSQL con soporte JSONB, facilitando la trazabilidad y el análisis posterior de las evidencias obtenidas.

Asimismo, el sistema incorpora mecanismos de autenticación, control de permisos y auditoría, permitiendo registrar de forma trazable quién realiza cada acción y cuándo. A partir de los datos recopilados, se aplica un sistema de puntuación de riesgos que facilita la priorización de hallazgos y apoya la toma de decisiones técnicas en fases posteriores de la auditoría. El proyecto adopta un enfoque ético, excluyendo funcionalidades de explotación automática y orientándose a entornos controlados de evaluación de seguridad.

UAX FP
Universidad
Alfonso X el Sabio

Abstract (Inglés)

The pre-exploitation phase is a fundamental step in offensive security audits, as it allows the identification and understanding of an application's attack surface before any exploitation attempts are made. However, in many real-world environments, this phase relies on isolated tools and manual processes that produce fragmented and hard-to-correlate information.

This project presents the development of a pre-exploitation platform for binary analysis, focused on ELF binaries running on Linux systems. The solution combines static analysis and controlled dynamic analysis to obtain relevant information about the binary structure, security mitigations, and runtime behavior.

The platform is conceived as an extensible product, based on a decoupled architecture that includes a backend implemented in Java using Spring Boot, a web-based interface, and specialized workers responsible for executing the analysis tasks. Results are persistently stored in PostgreSQL using JSONB, enabling traceability and further inspection of collected evidence.

In addition, the system incorporates authentication, authorization, and auditing mechanisms to record who performs each action and when. Based on the collected data, a risk scoring system is applied to prioritize findings and support technical decision-making in later stages of the audit. The project follows an ethical approach, explicitly excluding automatic exploitation features and targeting controlled security assessment environments.

UAX FP
Universidad
Alfonso X el Sabio

JUSTIFICACIÓN DEL PROYECTO

Motivación del proyecto

La motivación principal de este proyecto surge del interés por la seguridad ofensiva y por la necesidad de estructurar de forma adecuada la fase de pre-explotación dentro de una auditoría de seguridad. Esta fase resulta clave para comprender la superficie de ataque de un binario, evaluar sus mecanismos de protección y orientar correctamente los esfuerzos posteriores de análisis técnico.

En la práctica profesional, esta etapa inicial suele realizarse mediante el uso combinado de múltiples herramientas independientes y análisis manual, lo que genera información dispersa, dificulta su correlación y reduce la trazabilidad del trabajo realizado. Además, la ausencia de mecanismos de priorización obliga a depender en exceso del criterio individual del auditor, incrementando el riesgo de errores o análisis incompletos.

Este proyecto nace con el objetivo de ordenar y sistematizar esta fase, proporcionando una solución que centralice el análisis inicial de binarios y facilite una visión global, reproducible y priorizada de los resultados obtenidos.

Estado de la cuestión

Actualmente existen diversas herramientas ampliamente utilizadas en auditorías ofensivas para el análisis de binarios, como utilidades de análisis estático, herramientas de inspección de mitigaciones de seguridad o sistemas de trazado dinámico. Aunque estas herramientas son muy potentes de forma individual, presentan una serie de limitaciones comunes:

- Están diseñadas para resolver problemas concretos y no como soluciones integradas.

- Generan salidas heterogéneas que requieren interpretación manual.
- No ofrecen mecanismos nativos de correlación ni de priorización de riesgos.
- Carecen de trazabilidad sobre quién ha realizado el análisis y en qué contexto.
- No están pensadas para flujos de trabajo multiusuario.

Por otro lado, existen plataformas comerciales de análisis de malware o sandboxing que sí proporcionan soluciones más integradas, pero suelen ser cerradas, complejas de desplegar o excesivas para escenarios de auditoría técnica, laboratorio o formación avanzada.

Ante este contexto, se identifica un espacio claro para una solución intermedia: una plataforma abierta, extensible y orientada específicamente a la pre-explotación, que integre análisis estático y dinámico sin llegar a la explotación automática.

Público objetivo

El proyecto está dirigido principalmente a:

- Profesionales y estudiantes avanzados de ciberseguridad ofensiva.
- Auditores de seguridad que necesitan estructurar la fase inicial de análisis.
- Equipos técnicos que trabajen en entornos de laboratorio o evaluación controlada.
- Contextos académicos donde se requiera trazabilidad y reproducibilidad del análisis.

La plataforma no está orientada a usuarios sin conocimientos técnicos previos, sino a perfiles que ya comprenden los fundamentos del análisis de binarios y desean mejorar su eficiencia y organización.

Comparativa razonada (hacer tabla comparativa mejor)

Frente al uso de herramientas aisladas, la solución propuesta aporta:

- Centralización de resultados en un único sistema.

- Correlación entre análisis estático y dinámico.
- Priorización de hallazgos mediante un sistema de puntuación.
- Persistencia y trazabilidad de análisis y acciones.
- Soporte para flujos de trabajo multiusuario.

En comparación con plataformas comerciales cerradas, el proyecto destaca por:

- Ser una solución abierta y extensible.
- Ofrecer mayor control técnico sobre el proceso de análisis.
- Adaptarse a entornos de laboratorio y auditoría técnica.
- Evitar dependencias obligatorias de servicios externos.

De este modo, el proyecto se posiciona como una solución equilibrada entre simplicidad, control técnico y visión de producto.

Valor añadido del proyecto

El principal valor añadido de la propuesta reside en su enfoque estructurado de la fase de pre-explotación y en su diseño orientado a evolucionar hacia un producto real. La integración de mecanismos de autenticación, auditoría y persistencia de resultados permite que la plataforma sea adecuada para entornos profesionales donde la trazabilidad y la reproducibilidad son requisitos fundamentales.

Además, el proyecto sienta una base sólida para futuras ampliaciones, como la incorporación de nuevos módulos de análisis, mejoras en los mecanismos de contención o la integración con proveedores de identidad externos, manteniendo siempre un enfoque ético y controlado.

UAX FP
Universidad
Alfonso X el Sabio

INTRODUCCIÓN

Este proyecto aborda el desarrollo de una plataforma orientada a mejorar la fase de pre-explotación dentro de una auditoría de seguridad ofensiva. El objetivo principal es proporcionar una visión clara y estructurada sobre la superficie de ataque de un binario ejecutable antes de entrar en fases posteriores del análisis, facilitando la identificación de elementos relevantes desde un punto de vista técnico.

En muchos entornos, esta fase se realiza mediante el uso de herramientas independientes y análisis manual, lo que dificulta la correlación de resultados y reduce la trazabilidad del trabajo realizado. La solución propuesta pretende **centralizar este proceso**, integrando análisis estático y análisis dinámico controlado, y almacenando los resultados de forma persistente para su posterior consulta mediante una interfaz web.

Asimismo, al tratarse de un sistema diseñado con una visión de producto, la plataforma incorpora mecanismos de control de acceso y auditoría, permitiendo registrar quién realiza cada acción y cuándo. Este enfoque no solo mejora la organización del análisis, sino que también facilita la evolución futura del sistema hacia escenarios de uso más complejos.

Principales requisitos del proyecto:

- Control de acceso mediante usuarios autenticados y roles.
- Subida y gestión de binarios para su análisis.
- Análisis estático de binarios ELF y evaluación de mitigaciones de seguridad.
- Análisis dinámico controlado con registro de eventos relevantes.
- Almacenamiento persistente de resultados y consulta posterior.
- Generación de informes descargables.
- Registro de auditoría de acciones realizadas en el sistema.

UAX FP
Universidad
Alfonso X el Sabio

OBJETIVOS

Listado de objetivos que se plantean resolver. Requisitos.

Se debe presentar un **RFTP** inicial para acompañar a la propuesta.

R – Requisitos: Lo que debe hacer el programa expresado en lenguaje coloquial.

F – Funciones: Desglose de las características asociadas o subrequisitos de cada requisito. Expresado en leguaje técnico.

T – Tareas asociadas a cada funcionalidad. Deben describir completamente su alcance.

P – Pruebas. Demostración o prueba planificada para cumplir cada tarea.

R01 – El programa debe permitir acceder solo a usuarios autorizados

R01F01 – El usuario debe poder registrarse en el sistema

- R01F01T01 – Crear la tabla `users` en la base de datos (id, email, password_hash, enabled, created_at).
  - R01F01T01P01 – Insertar usuario de prueba y verificar persistencia.
- R01F01T02 – Implementar servicio de registro con validaciones (email único, formato, password mínimo).
  - R01F01T02P01 – Probar registro con email duplicado y password débil (debe fallar).
- R01F01T03 – Implementar endpoint REST `/auth/register`.
  - R01F01T03P01 – Petición HTTP válida devuelve 201 y usuario creado.
- R01F01T04 – Crear pantalla de registro en el frontend con validación de formularios.
  - R01F01T04P01 – Visualizar formulario y validar campos requeridos.

R01F02 – El usuario debe iniciar sesión

- R01F02T01 – Implementar verificación de credenciales usando hash seguro (BCrypt/Argon2).
  - R01F02T01P01 – Login correcto/incorrecto con usuario real.
- R01F02T02 – Implementar emisión de token (JWT) con expiración.
  - R01F02T02P01 – Probar acceso a endpoint protegido con y sin token.
- R01F02T03 – Implementar pantalla de login y almacenamiento seguro del token en frontend.
  - R01F02T03P01 – Login desde UI y navegación a zona privada.

R01F03 – El sistema debe controlar permisos por roles

- R01F03T01 – Crear tablas `roles` y `user_roles` y roles base (ADMIN/ANALYST/VIEWER).

    - R01F03T01P01 – Asignar rol a usuario y verificar en DB.
- R01F03T02 – Proteger endpoints por rol (Spring Security).
    - R01F03T02P01 – Un VIEWER no puede lanzar análisis (403).

R01F04 – (Opcional) El sistema debe poder integrarse con Auth0

- R01F04T01 – Documentar integración OIDC (redirect URIs, scopes, issuer, audience).
    - R01F04T01P01 – Checklist de configuración completado.

R02 – El programa debe poder guardar datos y mantener historiales de análisis

R02F01 – El sistema debe disponer de base de datos implantada

- R02F01T01 – Preparar PostgreSQL (docker dev) y parámetros de conexión.
    - R02F01T01P01 – Arrancar DB y conectar desde backend.
- R02F01T02 – Configurar migraciones (Flyway/Liquibase).
    - R02F01T02P01 – Ejecutar migraciones en entorno limpio sin errores.

R02F02 – El sistema debe guardar análisis y resultados

- R02F02T01 – Crear tablas `analyses` (metadatos) y `analysis_results` (JSON/JSONB).
    - R02F02T01P01 – Crear un análisis y guardar un results de prueba.
- R02F02T02 – Crear índices mínimos (analysis_id, created_at, hash).
    - R02F02T02P01 – Consultar listados por fecha sin degradación evidente.

R03 – El programa debe permitir subir binarios para su análisis

R03F01 – El usuario debe poder subir un ejecutable

- R03F01T01 – Implementar endpoint `/binaries/upload` (multipart) con límite de tamaño.
    - R03F01T01P01 – Subir binario válido y uno demasiado grande (debe rechazar).
- R03F01T02 – Almacenar archivo con ruta segura y nombre interno (no el original).
    - R03F01T02P01 – Verificar que no hay path traversal y que el archivo existe.
- R03F01T03 – Calcular SHA-256 y asociarlo al analysis.
    - R03F01T03P01 – Hash coincide con herramienta externa.

R03F02 – El sistema debe identificar el tipo de binario

- R03F02T01 – Detectar formato por magic bytes (ELF/PE/unknown).
    - R03F02T01P01 – Probar con fixtures ELF, PE y fichero aleatorio.

UAX FP
Universidad
Alfonso X el Sabio

R04 – El programa debe analizar el binario de forma estática (ELF MVP)

R04F01 – El sistema debe extraer información estructural

- R04F01T01 – Parsear cabecera ELF (arquitectura, entrypoint, tipo).
  - R04F01T01P01 – Comparar con `readelf -h` en binario de prueba.
- R04F01T02 – Listar secciones/segmentos con tamaños.
  - R04F01T02P01 – Comparar con `readelf -S` en muestra.

R04F02 – El sistema debe evaluar mitigaciones

- R04F02T01 – Determinar NX (segmentos/flags ejecutables).
  - R04F02T01P01 – Validar contra herramienta de referencia (p.ej. checksec).
- R04F02T02 – Determinar PIE (tipo de ejecutable).
  - R04F02T02P01 – Validar con binarios PIE/no PIE conocidos.
- R04F02T03 – Determinar RELRO (parcial/completo).
  - R04F02T03P01 – Validar contra checksec.

R04F03 – El sistema debe extraer strings relevantes

- R04F03T01 – Extraer strings con filtros (minlen, charset) y resumen.
  - R04F03T01P01 – Verificar conteo y ejemplos esperados.
- R04F03T02 – Marcar “strings interesantes” (URLs, paths, tokens).
  - R04F03T02P01 – Fixture con patrones → deben detectarse.

R05 – El programa debe ejecutar un análisis dinámico controlado

R05F01 – El sistema debe ejecutar con control de tiempo

- R05F01T01 – Ejecutar el binario con timeout y finalización segura.
  - R05F01T01P01 – Binario “sleep infinito” debe ser terminado.

R05F02 – El sistema debe registrar llamadas al sistema

- R05F02T01 – Implementar agente C (ptrace) que capture syscalls básicas.
  - R05F02T01P01 – Binario de prueba genera syscalls y se registran.
- R05F02T02 – Emitir eventos en formato estructurado (JSON lines).
  - R05F02T02P01 – Validar que el output es JSON válido y parseable.

R05F03 – El sistema debe registrar accesos a ficheros

- R05F03T01 – Detectar syscalls relacionadas con FS y registrar paths cuando sea posible.
  - R05F03T01P01 – Binario que abre un fichero debe reflejar evento.

UAX FP
Universidad
Alfonso X el Sabio

R06 – El programa debe guardar y mostrar resultados del análisis

R06F01 – El sistema debe almacenar resultados

- R06F01T01 – Persistir `results` (estático+dinámico) en `analysis_results` como JSON/JSONB.
    - R06F01T01P01 – Guardar y recuperar sin pérdida/alteración.

R06F02 – El usuario debe visualizar resultados

- R06F02T01 – Implementar endpoint `/analyses/{id}` con resultados.
    - R06F02T01P01 – Respuesta contiene secciones estático/dinámico.
- R06F02T02 – Crear vista UI con pestañas (Resumen/Estático/Dinámico).
    - R06F02T02P01 – Navegación por tabs correcta.

R07 – El programa debe generar un informe del análisis

R07F01 – El sistema debe generar informe HTML

- R07F01T01 – Generar HTML con secciones y hallazgos principales.
    - R07F01T01P01 – Abrir HTML y verificar contenido mínimo.
- R07F01T02 – Permitir descarga del informe.
    - R07F01T02P01 – Descargar y verificar integridad.

R08 – El programa debe registrar quién realiza cada acción (auditoría)

R08F01 – El sistema debe registrar eventos de auditoría

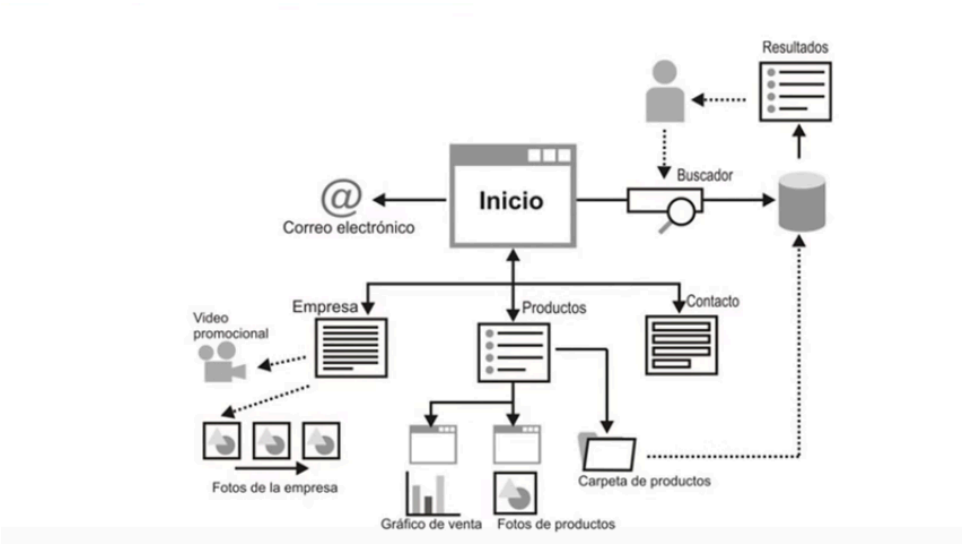
- R08F01T01 – Crear tabla `audit_events` (user_id, action, timestamp, analysis_id, result).
    - R08F01T01P01 – Insertar evento manual y consultarlo.
- R08F01T02 – Registrar automáticamente eventos en acciones clave (login, upload, analysis start).
    - R08F01T02P01 – Realizar flujo y comprobar eventos generados.
- R08F01T03 – Endpoint de consulta para ADMIN con filtros/paginación.
    - R08F01T03P01 – Consultar por fecha/usuario y validar paginado.

DESCRIPCIÓN

Se deben incluir todos los diagramas y explicaciones necesarias para entender el tipo de solución que propones en tu proyecto. Enumeramos algunos de los más comunes.
Todos deben ser perfectamente legibles.

Son ejemplos.

Arquitectura de la solución. Es un diagrama en el que se vea cómo funcionara el desarrollo planificado. Por ejemplo:



Casos de uso. Incluye diagrama y tabla con:

- Descripción.
- Precondiciones
- Postcondiciones
- Datos de entrada
- Datos de salida
- Tablas
- Clases
- Interfaces

Ejemplo:

Caso de uso: Pedir ayuda

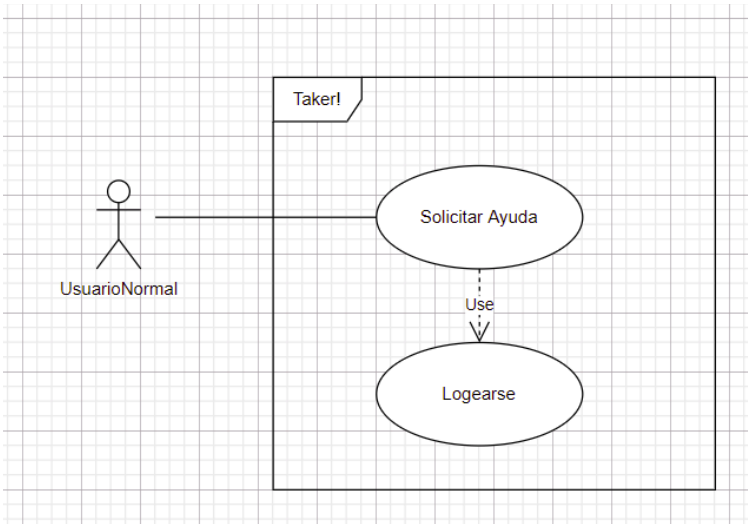


Ilustración 1: caso de uso Pedir Ayuda

DESCRIPCIÓN: Solicitar ayuda al especialista	
PRECONDICIONES: Usuario logado	**POSTCONDICIONES**: Solicitud en espera Se inicia el chat
DATOS ENTRADA Nombre especialista Id usuario Id especialista	**DATOS SALIDA** Nombre especialista Id usuario Id especialista Idchat Valoración fecha/hora

TABLAS: USUARIOS CHAT	**CLASES**: ESPECIALISTA.PHP USUARIO NORMAL.PHP CHAT.PHP
INTERFACES: PERFILUSUARIO.HTML CHAT.HTML	

Tabla 1: caso de uso Pedir Ayuda

DISEÑOS (Los que procedan según el tipo de proyecto)

Diagrama de clases.

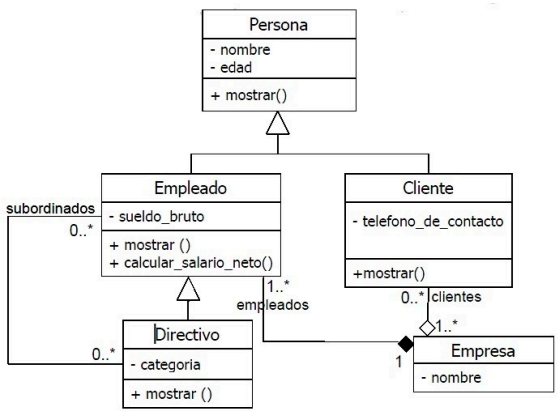


Diagrama E/R (Entidad - Relación)

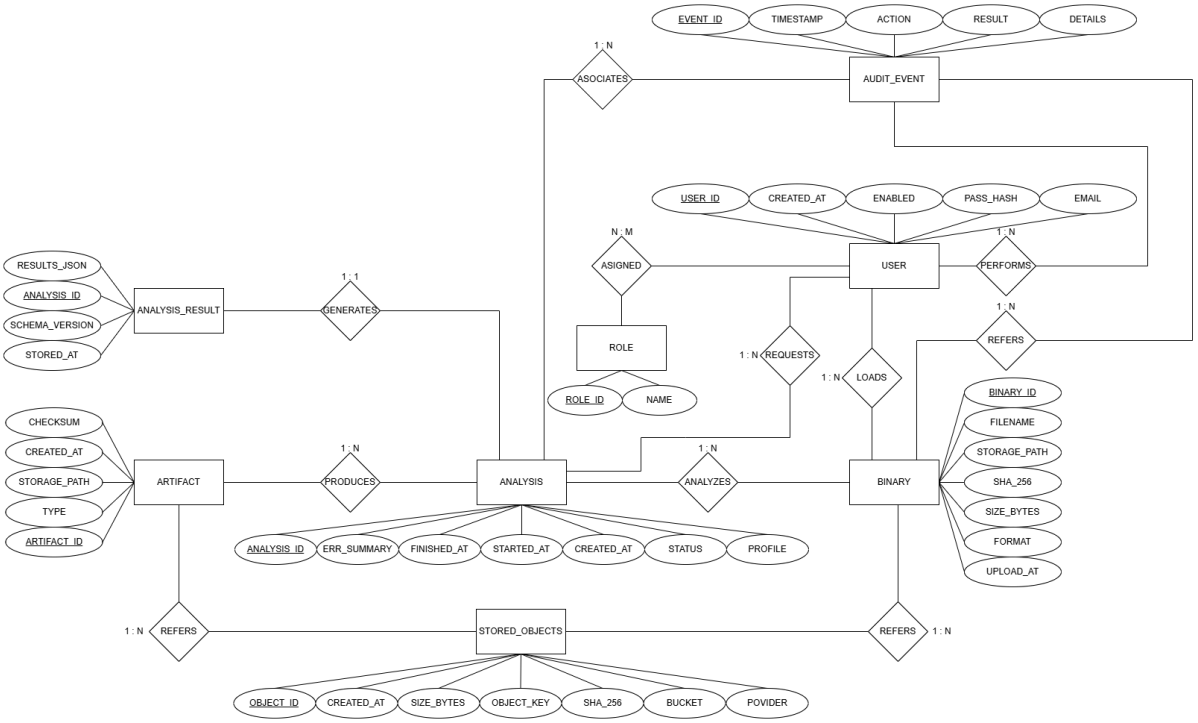


Diagrama de la base de datos. Con detalle de campos.

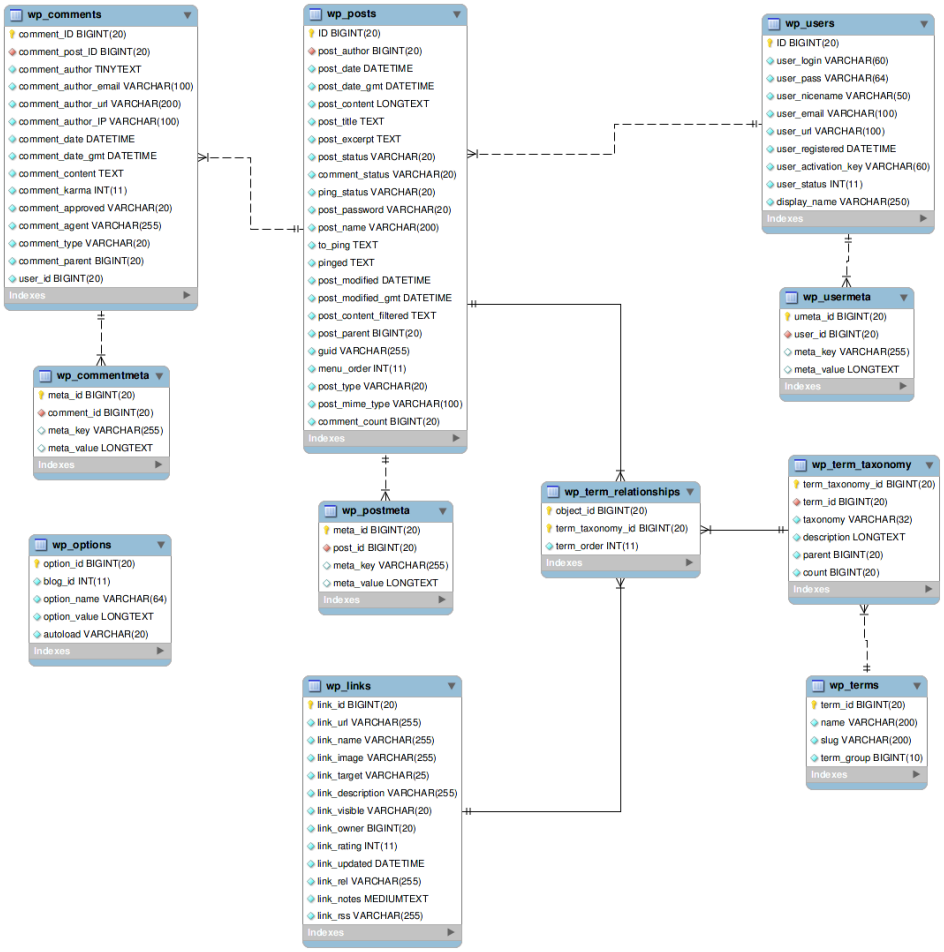
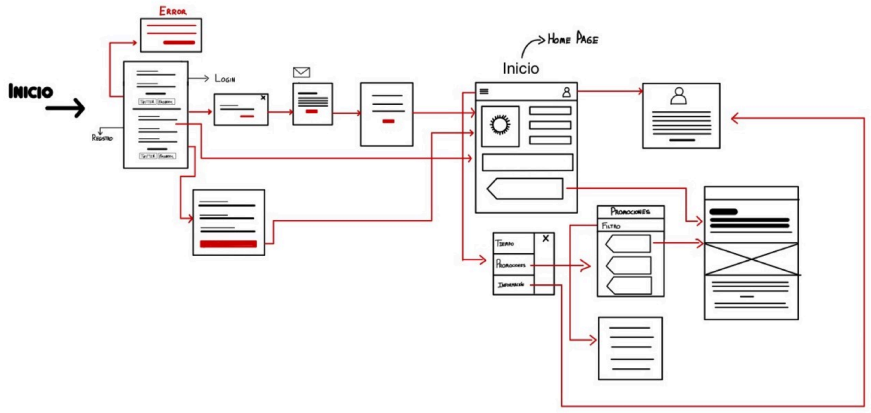
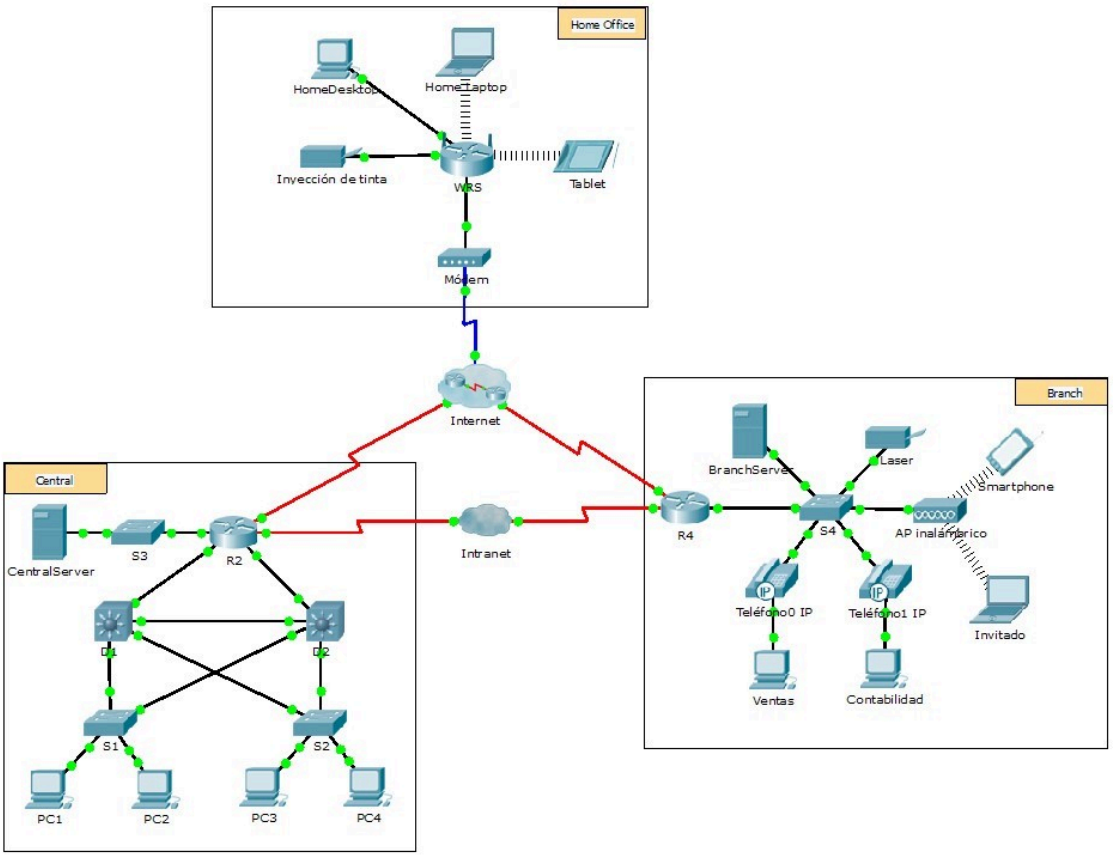


Diagrama de flujo de navegación. Esquemático. Debe incluirse en la propuesta.



Interfaces. Interesa ver la solución en diferentes tamaños o dispositivos.

Diagrama de red.



UAX FP
Universidad
Alfonso X el Sabio

TECNOLOGÍA

Las tecnologías y herramientas utilizadas para este proyecto. Por ejemplo:

Java.
Descripción de la herramienta.
Descripción del uso de la herramienta en el proyecto.

UAX FP
Universidad
Alfonso X el Sabio

METODOLOGÍA

Metodología usada y justificación de la misma.

Se presentarán dos planificaciones, una valoración inicial y previa a la implementación del proyecto y otra final con el tiempo real dedicado a cada parte del RFTP. Se analizarán las desviaciones.
El tiempo se expresará en horas. Debe existir una totalización final.

UAX FP
Universidad
Alfonso X el Sabio

Diagrama de Gantt realizado con Google spreadsheets de código abierto:

WBS	ID	Tarea	Inicio	Fin	Duracion_dias	Dependencias	Responsable	Entregable/Salida
1	PH1-000	Fase 1 - Diseño y Documentación Funcional	2026-02-11	2026-03-01	19		Mario	Baseline funcional y técnica v1
1.1	PH1-100	R01 - Autenticación y autorización	2026-02-11	2026-02-16	6		Mario	R01 definido y documentado
1.1.2001	PH1-110	R01F01 - Diseño gestión de usuarios	2026-02-11	2026-02-13	3		Mario	Flujos de usuarios definidos
1.1.1.1	PH1-111	R01F01T01 - Modelo usuarios y roles	2026-02-11	2026-02-12	2		Mario	Esquema users roles user_roles
1.1.1.2	PH1-112	R01F01T02 - Flujo creación usuarios por admin	2026-02-12	2026-02-13	2	PH1-111	Mario	Flujo Auth documentado
1.1.2002	PH1-120	R01F02 - Login y control de sesión	2026-02-13	2026-02-15	3	PH1-110	Mario	Login y JWT definidos
1.1.2.1	PH1-121	R01F02T01 - Validación credenciales	2026-02-1	2026-02-1	2	PH1-110	Mario	Proceso login documentado

UAX FP
Universidad Alfonso X el Sabio

			3	4				
1.1.2.2	PH1-122	R01F02T02 - Ciclo de vida JWT	2026-02-14	2026-02-15	2	PH1-121	Mario	JWT lifecycle definido
1.1.2003	PH1-130	R01F03 - Control de acceso por roles	2026-02-15	2026-02-16	2	PH1-120	Mario	Reglas RBAC definidas
1.2	PH1-200	R02 - Persistencia y almacenamiento	2026-02-16	2026-02-21	6	PH1-100	Mario	R02 definido y documentado
1.2.2001	PH1-210	R02F01 - Infraestructura base de datos	2026-02-16	2026-02-18	3	PH1-100	Mario	Postgres y migraciones definidos
1.2.1.1	PH1-211	R02F01T01 - Diseño esquema relacional	2026-02-16	2026-02-17	2		Mario	Modelo relacional v1
1.2.1.2	PH1-212	R02F01T02 - Estrategia migraciones	2026-02-17	2026-02-18	2	PH1-211	Mario	Estrategia Flyway definida
1.2.2002	PH1-220	R02F02 - Almacenamiento resultados análisis	2026-02-18	2026-02-21	4	PH1-210	Mario	Modelo persistencia análisis
1.2.2.1	PH1-221	R02F02T01 - Diseño analysis y results JSONB	2026-02-18	2026-02-20	3	PH1-210	Mario	Tablas analyses y analysis_results
1.2.2.2	PH1-222	R02F02T02 - Índices y restricciones	2026-02-20	2026-02-21	2	PH1-221	Mario	Estrategia índices definida

UAX FP
Universidad Alfonso X el Sabio

1.3	PH1-300	R03 - Gestión de binarios	2026-02-21	2026-02-26	6	PH1-200	Mario	R03 definido y documentado
1.3.2001	PH1-310	R03F01 - Subida y deduplicación binarios	2026-02-21	2026-02-24	4	PH1-200	Mario	Flujo upload definido
1.3.1.1	PH1-311	R03F01T01 - Contrato subida binarios	2026-02-21	2026-02-22	2		Mario	Endpoint upload documentado
1.3.1.2	PH1-312	R03F01T02 - Estrategia almacenamiento objetos	2026-02-22	2026-02-23	2	PH1-311	Mario	Uso MinIO definido
1.3.1.3	PH1-313	R03F01T03 - Deduplicación por hash	2026-02-23	2026-02-24	2	PH1-312	Mario	Flujo dedupe documentado
1.3.2002	PH1-320	R03F02 - Identificación tipo binario	2026-02-24	2026-02-26	3	PH1-310	Mario	Detección ELF definida
1.3.2.1	PH1-321	R03F02T01 - Reglas detección ELF	2026-02-24	2026-02-25	2		Mario	Lógica detección documentada
1.4	PH1-400	R04 - Análisis estático	2026-02-26	2026-03-01	4	PH1-300	Mario	R04 definido y documentado
1.4.2001	PH1-410	R04F01 - Extracción estructura ELF	2026-02-26	2026-02-28	3	PH1-300	Mario	Estructura ELF documentada
1.4.1.1	PH1-411	R04F01T01 - Cabeceras y secciones ELF	2026-02-26	2026-02-27	2		Mario	Parse ELF definido

1.4.1.2	PH1-412	R04F01T02 - Dependencias y segmentos	2026-02-27	2026-02-28	2	PH1-411	Mario	Deps y segments documentados
1.4.2002	PH1-420	R04F02 - Evaluación mitigaciones	2026-02-28	2026-03-01	2	PH1-410	Mario	NX PIE RELRO definidos

Actualmente faltan los R05, R06, R07 y R08 por incluir en la planificación provisional.

Presupuesto. Con detalle de horas, indispensable si se realiza en grupo, y coste total del desarrollo por cada requisito.

README y GIT.

Repositorio remoto licenciado bajo MIT: mario-asenjo/DAM-TFG-PPAOB: Repositorio de mi trabajo final de grado superior de Desarrollo de Aplicaciones Multiplataforma.

UAX FP
Universidad
Alfonso X el Sabio

TRABAJOS FUTUROS

Trabajos de ampliación y mejora proyectados.

UAX FP
Universidad
Alfonso X el Sabio

CONCLUSIONES

Conclusión profesional del proyecto.

UAX FP
Universidad
Alfonso X el Sabio

REFERENCIAS

Glosario:

Para el documento Gantt las columnas utilizadas son:

`WBS` → identificador jerárquico

`RFTP_ID` → referencia directa al requisito

`Task_Name` → nombre legible

`Phase` → Documentation o Implementation

`Start_Week` → semana relativa

`End_Week` → semana relativa

`Owner` → You

`Difficulty_1_5` → dificultad estimada

`Deliverable` → qué se entrega