DOCUMENTO TÉCNICO

[RetroVault]
Cristian Suárez Melian

[RV_2025]. [RetroVault]
[App_v1.0]. [Aplicación Web de Coleccionismo]

Nombre del fichero:	DAW_PRW_CSM_UT01.4. Documento Técnico.odt
Fecha de esta versión:	18/02/2026

Historial de revisiones

Fecha	Descripción	Autor
[18/01/2025]	[Creación del documento técnico]	[Cristian Suárez Melian]
[25/01/2025]	[Modificación del documento técnico]	[Cristian Suárez Melian]
[18/02/2025]	[Modificación del documento técnico]	[Cristian Suárez Melian]

ÍNDICE

1 INTRODUCCIÓN

El presente documento detalla la arquitectura de software y el stack tecnológico seleccionado para el desarrollo del sistema RetroVault.

La solución se plantea como una aplicación web basada en el patrón de diseño Modelo-Vista-Controlador (MVC), complementada con una arquitectura de microservicios para la delegación de tareas de computación intensiva (Inteligencia Artificial). Esta estructura híbrida garantiza la robustez en la gestión de datos transaccionales y la flexibilidad necesaria para integrar funcionalidades avanzadas de procesamiento de imagen.

2 ARQUITECTURA DEL SISTEMA

El sistema se divide en tres capas lógicas:

1. Capa de Presentación (Frontend): Renderizado en servidor para optimizar la carga.
2. Capa de Negocio (Backend Core): Lógica principal, seguridad y orquestación de servicios.
3. Capa de Servicios Externos (Microservicio IA): Módulo desacoplado para visión por computador.

3 STACK TECNOLÓGICO: FRONTEND

Se ha priorizado un stack que permita una integración nativa con el motor de plantillas del backend, asegurando una validación de datos robusta desde el servidor.

Técnologia	**Versión**	**Función**	**Justificación**
HTML	5.0	Estructura semántica de las vistas.	Estándar web obligatorio. Uso de etiquetas semánticas para accesibilidad.
CSS	3.0	Estilos, diseño responsivo y animaciones.	Permite el diseño adaptativo requerido por el usuario.
Thymeleaf	3.1	Motor de plantillas Java.	Permite inyectar datos de Java directamente en el HTML. Facilita la gestión de seguridad y visualización de datos dinámicos.
JavaScript	ES6+	Interactividad del lado del cliente.	Manejo de eventos puntuales y validación de formularios.

Bootstrap	5.3	Framework CSS / UI	Proporciona un sistema de rejilla (grid) responsive y componentes pre-estilizados para garantizar una interfaz adaptable a dispositivos móviles.

4 STACK TECNOLÓGICO: BACKEND

El núcleo del sistema reside en un entorno Java robusto, aprovechando el ecosistema Spring para la inyección de dependencias y la gestión de transacciones.

Tecnología	**Versión**	**Función**	**Justificación**
Java JDK	17	Lenguaje principal.	Versión de soporte principal. Ofrece tipado fuerte, alto rendimiento y estabilidad para la lógica de negocio.
Spring Boot	3.2	Framework de desarrollo.	Provee auto-configuración, servidor embebido (Tomcat) y facilita la creación de aplicaciones listas para producción.
Spring Security	6.0	Seguridad y Control de acceso.	Implementación de autenticación, autorización basada en roles y protección contra ataques comunes.

5 MÓDULO DE IA

Para cumplir con el requisito de innovación, se ha desacoplado la lógica de IA en un servicio independiente.

Tecnología	**Versión**	**Función**	**Justificación**
Google Gemini API	v1beta (Modelo 2.5 Flash)	Motor de IA Generativa.	Integración nativa vía REST en Java para análisis multimodal.

6 SISTEMA DE GESTIÓN DE DATOS

Tecnología	Versión	Función	Justificación
MySQL	8.0.40	Sistema de gestión de base de datos relacional.	Se utiliza la última versión menor estable (8.0.40) para garantizar parches de seguridad extendidos hasta 2026, asegurando estabilidad en producción antes del salto a la rama 8.4
Cloudinary	SDK 1.36.0	Gestión de Medios en la Nube	Almacenamiento persistente de imágenes y optimización automática, solucionando la limitación de sistema de archivos efímero en entornos PaaS

7 HERRAMIENTAS DE DESARROLLO

El ciclo de vida del desarrollo software (SDLC) se apoya en las siguientes herramientas para garantizar la calidad y el control.

Categoría	Herramienta	Versión	Uso en el proyecto
IDE	Eclipse / VSCode	1.86	Entorno de desarrollo con soporte para las tecnologías elegidas.
Control de versiones	Git	2.43	Gestión local del historial de cambios.
Repositorio remoto	GitHub	Web	Alojamiento del código fuente y documentación.
Gestión de dependencias	Maven	3.9.6	Gestión del ciclo de vida de construcción y dependencias Java.
Diseños UML/ER	Draw.io	24.0	Modelado de diagramas.

8 RESUMEN DE INTEGRACIÓN

El flujo de datos principal para la funcionalidad de IA es un ejemplo de Arquitectura Orientada a Servicios:

1. El usuario sube una imagen a Spring Boot.
2. Java actúa como cliente HTTP y reenvía la imagen al microservicio Python.
3. Python procesa la imagen y devuelve un objeto JSON que contiene: título, género, año...
4. Java deserializa el JSON y lo presenta en la vista Thymeleaf.

Esta separación de responsabilidades asegura que un fallo en el módulo de IA no detenga la ejecución principal de la aplicación web.

9 ANÁLISIS DE RIESGO Y PLAN DE CONTINGENCIA

Dada la complejidad técnica de integrar un entorno híbrido (Java + Python) en el tiempo establecido, se ha definido una matriz de riesgos y sus correspondientes acciones mitigadoras para garantizar la viabilidad del proyecto.

Riesgo detectado	**Nivel de impacto**	**Plan de contingencia**
Fallo en la integración del micro-servicio IA	Alto	Si la comunicación HTTP entre Spring Boot y Python presenta errores de configuración en el despliegue, se deshabilitará el módulo de "autocompletado con IA". El sistema pasará a un modo de carga manual estándar, donde el usuario deberá cargar la imagen manualmente.
Curva de aprendizaje de Python	Medio	Se priorizará el desarrollo del CRUD funcional en Java. El módulo de IA se tratará como una funcionalidad “Deseable”. Si no se alcanza un nivel de precisión adecuado en la detección de texto, se mantendrá la funcionalidad de subida de imágenes.