DOCUMENTO TÉCNICO - BIBLIOTECA VIRTUAL

Grupo: 1
Fecha: Septiembre 2025
Versión: v1.0-final

---

1. INTRODUCCIÓN

Este proyecto implementa una API REST para la gestión de una biblioteca virtual, permitiendo administrar usuarios, libros y préstamos de forma eficiente.

---

2. DECISIONES DE DISEÑO

2.1 Arquitectura

- Arquitectura monolítica modular con Spring Boot
- Separación en capas: Controller, Service, Repository
- Uso de DTOs para separar modelo de negocio de la API

2.2 Base de datos

- MySQL 8.0 como motor de base de datos relacional
- Flyway para control de versiones de esquema
- Relaciones: User 1:N Loan, Book 1:N Loan

2.3 Tecnologías

- Java 21: última versión LTS
- Spring Boot 3.5.6: framework robusto
- JPA/Hibernate: persistencia de datos
- Docker: despliegue reproducible

---

3. FUNCIONALIDADES IMPLEMENTADAS

Gestión de Libros:

- CRUD completo
- Búsqueda por título, autor y categoría
- Control de copias disponibles

Gestión de Usuarios:

- CRUD completo
- Roles: USER y LIBRARIAN

- Búsqueda por username

Gestión de Préstamos:

- Crear préstamos
- Devolver libros
- Historial por usuario y libro
- Estados: ACTIVE, RETURNED

---

4. OBSERVABILIDAD

Se implementó Spring Boot Actuator con:

- Health check: `/actuator/health`
- Métricas: `/actuator/metrics`
- Monitoreo de JVM, base de datos y HTTP

---

5. LIMITACIONES ACTUALES

- No hay autenticación JWT implementada
- Falta validación de roles en endpoints
- No hay manejo de fechas de vencimiento automático

---

6. LECCIONES APRENDIDAS

- Importancia de las migraciones de base de datos
- Documentación automática con Swagger facilita pruebas
- Docker simplifica el despliegue
- Tests unitarios ayudan a detectar errores temprano

---

7. BACKLOG TÉCNICO

Prioridad Alta:

- Implementar Spring Security con JWT
- Validación de roles (LIBRARIAN/USER)
- Notificaciones de vencimiento

Prioridad Media:

- Tests de integración completos
- Manejo de excepciones personalizado

- Paginación en listados

Prioridad Baja:

- Caché con Redis
- Logs estructurados con ELK
- API de reportes

---

8. CONCLUSIÓN

El MVP cumple con todos los requisitos funcionales. La aplicación es escalable y está lista para agregar autenticación y nuevas funcionalidades.