INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

“ESCOM”

UNIDAD DE APRENDIZAJE

INGENIERÍA DE SOFTWARE

Informe de Implementación

6CV3

García García Aram Jesua

Profesor: Gabriel Hurtado Avilés

Fecha de entrega: 07 de abril de 2025

1. Decisiones Técnicas Tomadas

1.1. Arquitectura del Sistema

1. **Backend con Spring Boot:**
    - Se eligió por su robustez, integración con Spring Security (JWT) y facilidad para crear APIs RESTful.
    - Ventajas: Escalabilidad, soporte para JPA/Hibernate y configuración modular.
2. **Frontend con HTML/CSS/JS Vanilla:**
    - Se optó por no usar frameworks frontend (React/Vue) para mantener simplicidad y carga rápida.
    - Se implementó un diseño responsive con CSS Grid/Flexbox y variables CSS para temas claro/oscuro.

1.2. Autenticación y Seguridad

3. **JWT (JSON Web Tokens):**
    - Decisión: Reemplazar sesiones tradicionales por JWT para un sistema stateless y escalable.
    - Implementación: Tokens firmados con clave secreta, almacenados en el cliente (localStorage).
    - Spring Security para gestión de roles (USER/ADMIN) y protección de endpoints.
4. **BCrypt para hashing de contraseñas:**
    - Garantiza almacenamiento seguro en la base de datos.

1.3. Integración con OpenLibrary

API RESTful:

- Conexión asíncrona mediante fetch en el frontend para obtener datos de libros.
- Se cachearon respuestas para reducir llamadas redundantes a la API.

1.4. Base de Datos

MySQL:

- Elección basada en soporte para relaciones complejas (usuarios, roles, preferencias).
- Dockerizado para entornos de desarrollo (docker-compose.yml).

1.5. Despliegue y Desarrollo

- **Docker:**
  - Contenerización de la app y la base de datos para garantizar consistencia entre entornos.
- **LiveReload y Spring DevTools:**
  - Agilizan el desarrollo con recarga automática de cambios.

2. Problemas Encontrados y Soluciones Implementadas

2.1. Problema: Conexión Inestable con OpenLibrary

- **Síntoma**: Tiempos de respuesta largos o fallos en la carga de libros.
- **Solución**:
  - Implementar un sistema de reintentos automáticos en el frontend.
  - Mostrar un mensaje amigable al usuario ("Reintentar" o "Volver más tarde").

2.2. Problema: Gestión de Roles en JWT

- **Síntoma**: Dificultad para actualizar permisos de usuario sin cerrar sesión.
- **Solución**:
  - Invalidar tokens antiguos al cambiar roles y forzar reautenticación.

2.3. Problema: Rendimiento en Búsquedas

- **Síntoma**: Lentitud al filtrar libros por categorías o términos.
- **Solución**:
    - Paginación en el frontend y backend.
    - Uso de índices en la base de datos para consultas frecuentes.

2.4. Problema: Compatibilidad con Navegadores

- **Síntoma**: Estilos no consistentes en navegadores antiguos.
- **Solución**:
    - Uso de prefijos CSS (-webkit-, -moz-) y polyfills para JavaScript.

3. Tecnologías y Patrones Utilizados

3.1. Tecnologías Clave

Área	Tecnologías
Backend	Spring Boot, Spring Security, JPA/Hibernate, MySQL, JWT.
Frontend	HTML5, CSS3, JavaScript Vanilla, FontAwesome (iconos).
DevOps	Docker, Docker Compose.
APIs	OpenLibrary (REST), Fetch API (frontend).

3.2. Patrones de Diseño

- **MVC (Modelo-Vista-Controlador):**
    - Separación clara entre lógica de negocio (Spring), presentación (HTML/JS) y datos (MySQL).
- **Singleton:**
    - Para gestionar la configuración de JWT y conexiones a la base de datos.
- **Factory:**

    - Creación dinámica de objetos de libros basados en respuestas de OpenLibrary.

4. Posibles Mejoras y Trabajos Futuros

4.1. Mejoras Inmediatas

- **Sistema de Recomendaciones:**
    - Implementar algoritmos basados en historial de búsqueda (ej: filtrado colaborativo).
- **Reseñas y Valoraciones:**
    - Permitir a los usuarios calificar libros y dejar comentarios.
- **Exportación de Listas:**
    - Opción para guardar listas de libros en PDF o CSV.

4.2. Escalabilidad

- **Microservicios:**
    - Dividir el sistema en módulos independientes (auth, catálogo, recomendaciones).
- **API GraphQL:**
    - Reemplazar REST para consultas flexibles y reducir over-fetching de datos.

Conclusión:

El sistema cumple con su objetivo principal (exploración de libros) gracias a decisiones técnicas como JWT, Spring Boot e integración con OpenLibrary. Los problemas se resolvieron con soluciones escalables, y las mejoras futuras apuntan a personalización y escalabilidad.