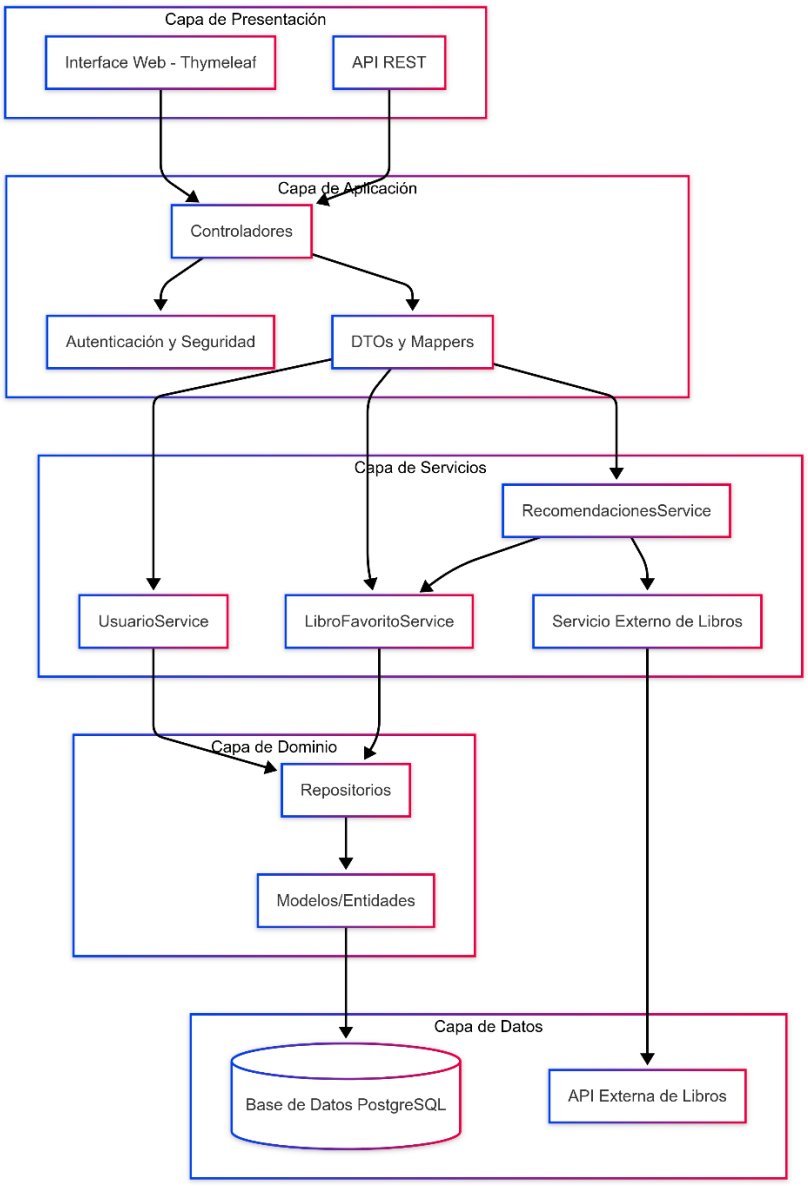
Modelo de Implementación

Arquitectura Técnica del Sistema



Tecnologías y Frameworks Utilizados

Backend

- **Java 17** como lenguaje de programación principal.
- **Spring Boot 3.1.x** para la construcción del backend.
- **Spring Security** para la autenticación y autorización de usuarios.

- **Spring Data JPA** para la persistencia de datos.
- **Hibernate** como implementación de JPA.
- **Lombok** para la reducción de código repetitivo (boilerplate).

Frontend

- **Thymeleaf** como motor de plantillas para la generación dinámica de vistas.
- **Bootstrap 5** para diseño responsivo y componentes visuales.
- **JavaScript/jQuery** para la interactividad del cliente.

Base de Datos

- **PostgreSQL** como sistema de gestión de bases de datos relacional.

API y Comunicación

- **REST API** para la comunicación entre el frontend y el backend.
- **RestTemplate** o **WebClient** como cliente para el consumo de APIs externas de libros.
- **JSON** como formato de intercambio de datos.

Herramientas de Desarrollo

- **Maven** para la gestión de dependencias y el ciclo de vida del proyecto.
- **Git** como sistema de control de versiones.
- **JUnit** y **Mockito** para la implementación de pruebas unitarias y de integración.

Estructura de Componentes y Organización

La aplicación está estructurada en capas, siguiendo una arquitectura limpia que garantiza una separación clara de responsabilidades:

Capa de Presentación

- Controladores MVC para la gestión de vistas Thymeleaf.
- Controladores REST para la exposición de endpoints de API.
- Plantillas Thymeleaf organizadas por funcionalidades específicas.

Capa de Aplicación

- **DTOs (Data Transfer Objects)** para la transferencia eficiente de datos entre capas.
- **Mappers** para la conversión entre entidades y DTOs.
- **Filtros de seguridad** y configuración avanzada de autenticación.

Capa de Servicios

- Implementación de la lógica de negocio central.
- Integración con servicios externos.
- Aplicación de patrones de diseño como Factory Method.

Capa de Dominio

- **Entidades JPA** que representan conceptos del negocio (Usuario, LibroFavorito, Rol).
- **Repositorios** para el acceso y manipulación de datos.
- **Especificaciones** para la creación de consultas complejas.

Capa de Datos

- Configuración de conexiones a la base de datos.
- Migraciones de esquemas mediante herramientas de gestión de cambios.
- Configuración de clientes para el consumo de APIs externas.

Consideraciones sobre Seguridad, Rendimiento y Escalabilidad

Seguridad

Autenticación y Autorización

- Implementación de Spring Security con autenticación basada en sesiones.
- Definición de roles diferenciados (ADMIN, USER) para el control de acceso.
- Cifrado de contraseñas utilizando BCrypt.
- Protección contra ataques comunes como CSRF, XSS y SQL Injection.

Validación de Datos

- Validaciones tanto en el frontend como en el backend.
- Sanitización de entradas de usuario.
- Control estricto de acceso a nivel de servicios y controladores.

Rendimiento

Caché

- Implementación de caché para resultados de consumo de API externa.
- Uso de caché de segundo nivel de Hibernate para entidades de alta demanda.

Optimización de Consultas

- Utilización de consultas específicas y optimizadas en los repositorios.

- Aplicación de paginación para el manejo de grandes volúmenes de resultados.
- Búsquedas eficientemente indexadas.

Escalabilidad

Diseño Modular

- Componentes desacoplados que permiten un escalado horizontal sencillo.
- Servicios diseñados como stateless, favoreciendo la ejecución en múltiples instancias.

Base de Datos

- Índices optimizados para consultas rápidas.
- Uso de conexiones en pool para mejorar la gestión de recursos.
- Planificación para futura migración a clusters de base de datos en caso de necesidad.