Informe Técnico: Sistema de Búsqueda y Recomendación de Libros

Este informe explica la correspondencia entre los modelos de análisis, diseño e implementación, las decisiones técnicas tomadas, los desafíos encontrados y las métricas de calidad aplicadas al sistema de gestión de libros.

1. Correspondencia entre Modelos de Análisis, Diseño e Implementación

El sistema presenta una correspondencia clara entre las fases de análisis, diseño e implementación:

- **Modelo de Dominio**: Define entidades como `Usuario`, `LibroFavorito`, `Recomendación` y `Rol`.
- **Diseño**: A través de diagramas de clases y arquitectura en capas, se organiza la estructura del sistema y la interacción entre los componentes.
- **Implementación**: Se lleva a cabo en Java usando Spring Boot, respetando las relaciones y responsabilidades diseñadas.

Correspondencias específicas:

- `Usuario` se implementa como `UserModel.java`, incluyendo atributos y relaciones a roles y libros favoritos.
- `LibroFavorito` es representado como una entidad persistente en `LibroFavorito.java`.
- `Recomendación` es gestionada en `RecomendacionesService.java`, conectando los libros favoritos con recomendaciones personalizadas.
- La arquitectura sigue el patrón de capas: presentación, aplicación (servicios), dominio (entidades) y datos (repositorios).

Además, se utiliza el patrón **Factory Method** para permitir flexibilidad en la generación de recomendaciones, anticipando cambios futuros.

2. Decisiones Técnicas Tomadas y su Justificación

- **Stack Tecnológico**:
  - **Java 17** y **Spring Boot 3.1.x**: Para aprovechar mejoras de lenguaje y framework.
  - **Spring Security**: Gestión robusta de autenticación y autorización.
  - **Thymeleaf + Bootstrap 5**: Desarrollo de interfaces web responsivas.
  - **PostgreSQL** y **Hibernate**: Persistencia eficiente con JPA.
- **Seguridad**:

    - Uso de **BCrypt** para cifrado de contraseñas.
    - Definición de roles y restricciones de acceso a rutas.
- **Comunicación con API Externa**:
    - Uso de `RestTemplate` para obtener datos de OpenLibrary.
    - Implementación de manejo de errores para mayor resiliencia.
- **Gestión de Caché**:
    - Cacheo de resultados de la API y uso de caché de segundo nivel en Hibernate para mejorar el rendimiento.

Estas decisiones fueron tomadas para maximizar la productividad, seguridad, rendimiento y escalabilidad del sistema.

3. Desafíos Encontrados y Soluciones Implementadas

- **Aprendizaje de Spring Boot y Docker**:
    - Solución: Adopción progresiva de buenas prácticas de Spring (anotaciones, estructura por capas) y uso de Docker para despliegue y contenedorización.
- **Implementación del Sistema de Recomendaciones**:
    - Solución: Desarrollo de `RecomendacionesService`, que genera recomendaciones basadas en autores favoritos, evita duplicados y maneja fallos de conexión a la API externa.
- **Seguridad por Niveles de Rol**:
    - Solución: Configuración de `SecurityConfig.java`, creación de servicios personalizados de autenticación, y restricciones de acceso basadas en roles.

Cada desafío fue enfrentado priorizando la robustez y la mantenibilidad del sistema.

4. Métricas de Calidad Aplicadas al Diseño

- **Principio de Responsabilidad Única (SRP)**:
    - Cada clase (servicio, entidad, repositorio) tiene una única responsabilidad bien definida.
- **Principio Abierto/Cerrado (OCP)**:
    - Implementación del patrón Factory Method para recomendaciones, facilitando extensiones futuras sin modificar código existente.
- **Cohesión y Acoplamiento**:
    - Alta cohesión interna en cada clase.
    - Bajo acoplamiento gracias a la inyección de dependencias y uso de interfaces.
- **Mantenibilidad**:
    - Código modular y estructurado.
    - Métodos pequeños y específicos que favorecen la facilidad de prueba y depuración.
- **Escalabilidad**:
    - Diseño de servicios stateless que facilita el escalado horizontal.

- Implementación de paginación en consultas para manejar grandes volúmenes de datos.

Estas métricas garantizan un sistema fácil de mantener, seguro y preparado para el crecimiento futuro.

Conclusiones

El sistema de búsqueda y recomendación de libros presenta una correspondencia sólida entre el análisis, el diseño y la implementación. Las decisiones técnicas priorizaron la seguridad, el rendimiento y la facilidad de mantenimiento. Los desafíos encontrados durante el desarrollo se resolvieron con estrategias que fortalecieron la calidad final del producto.

El uso de principios SOLID, el patrón Factory Method, y una arquitectura por capas consolidan un sistema bien organizado, escalable y preparado para futuras extensiones.