UAX Universidad Alfonso X el Sabio

Programación concurrente

SISTEMA GESTIÓN BIBLIOTECAS

By

Pablo Rodriguez

139852

Departamento de ingenieria Informática

Ingenieria Informática

RESUMEN

Este proyecto presenta un Sistema de Gestión de Bibliotecas (LMS) diseñado para una biblioteca pública de gran tamaño, utilizando Hibernate y JPA con una base de datos utilizando PostgreSQL para manejar eficientemente miles de libros y usuarios. El proyecto se ha estructurado como un proyecto Maven [1], facilitando la gestión de dependencias y la construcción del software. Se ha implementado un modelo de datos integral con funcionalidades para pedir prestado, devolver y renovar libros, así como para agregar y eliminar volúmenes. El sistema incluye control de concurrencia para evitar conflictos en préstamos simultáneos y características de auditoría para rastrear las actividades de los usuarios.

CONTENTS

1. INTRODUCCIÓN

En la era de la digitalización, la gestión eficiente de las bibliotecas públicas grandes se ha convertido en un desafío crucial, especialmente cuando se trata de manejar miles de libros y atender a cientos de usuarios. Este proyecto aborda la necesidad de un Sistema de Gestión de Bibliotecas (LMS) robusto y eficiente, diseñado específicamente para facilitar el préstamo, devolución y renovación de libros, así como la gestión de inventario y el seguimiento de préstamos.

El núcleo de este proyecto es el desarrollo de un LMS que no solo responda a las necesidades operativas de una biblioteca moderna sino que también garantice la seguridad y eficiencia en el manejo de datos. Para lograr esto, se ha empleado una combinación de tecnologías avanzadas, incluyendo Hibernate [2] y JPA [3] para la interacción con una base de datos SQL. Estas herramientas han sido seleccionadas por su capacidad para manejar solicitudes concurrentes de manera segura, un requisito clave para un entorno dinámico como el de una biblioteca pública.

Un aspecto destacado de este proyecto es el uso de Bootify [4] , una herramienta que ha facilitado la creación de la base de datos. Este enfoque ha permitido una implementación más rápida y eficiente, alineándose perfectamente con los requisitos dinámicos del sistema. Además, la fase de diseño del LMS se ha basado en un modelo UML detallado, lo que ha garantizado una estructura clara y una visión coherente durante todo el proceso de desarrollo.

2. METODOS

2.1. MODELADO DEL SISTEMA

Utilizando el diseño orientado a objetos, el sistema se modeló basándose en un diagrama UML, que define las entidades clave y sus relaciones:

- Usuario: Clase base que define las propiedades comunes de todos los usuarios del sistema, como el nombre y el identificador (ID).
- Lector y Bibliotecario: Subclases de Usuario que heredan atributos comunes y añaden relaciones y funcionalidades específicas. Por ejemplo, Lector está asociado con préstamos de libros, mientras que Bibliotecario puede gestionar libros y realizar préstamos a los lectores.
- Libro: Representa los recursos de la biblioteca con atributos como título, autor y estado (disponible, prestado, etc.).
- Prestamo: Encapsula la relación entre un Lector y un Libro, incluyendo fechas de inicio y finalización del préstamo.
- Biblioteca: (No se especifican atributos en el diagrama proporcionado) Podría usarse para representar la colección completa de libros o las operaciones de la biblioteca en su conjunto.

2.2. IMPLEMENTACIÓN DE LA BASE DE DATOS

Siguiendo el modelo UML, se diseñó un esquema de base de datos relacional. Cada clase del UML se tradujo en una tabla correspondiente, con Hibernate y JPA facilitando el mapeo entre las entidades orientadas a objetos y las tablas de la base de datos. Bootify se utilizó para generar el código inicial de la base de datos, acelerando el proceso de desarrollo.

2.3. DESARROLLO DE FUNCIONALIDADES

Cada entidad UML se convirtió en una entidad de dominio en el sistema. Se implementaron las siguientes funcionalidades principales:

- Gestión de Usuarios: Creación y mantenimiento de perfiles de usuario para Lectores y Bibliotecarios.
- Operaciones sobre Libros: Permitir operaciones como añadir, actualizar o eliminar registros de libros.
- Manejo de Préstamos: Facilitar el proceso de préstamo de libros, así como la renovación y devolución, con mecanismos para manejar solicitudes concurrentes y evitar condiciones de carrera.

2.4. DESARROLLO DE LA INTERFAZ DEL USUARIO

2.4.1. Implementacion

El desarrollo de la interfaz web del Sistema de Gestión de Bibliotecas se realizó utilizando tecnologías modernas de front-end como Bootstrap [5]. Para la construcción y gestión de las dependencias del lado del cliente, se empleó Node.js y npm [6] , lo cual permitió una integración eficiente de diversas bibliotecas y frameworks de JavaScript. Este enfoque aseguró que la interfaz de usuario fuera interactiva, receptiva y fácil de usar, mejorando significativamente la experiencia del usuario final.

2.4.2. Servidor

La lógica del servidor y la configuración de la aplicación se han implementado utilizando Spring Boot [7] , que ha simplificado significativamente el proceso de configuración y despliegue de la aplicación, permitiendo una integración eficiente de servicios REST y una gestión de dependencias automatizada.

2.5. UML

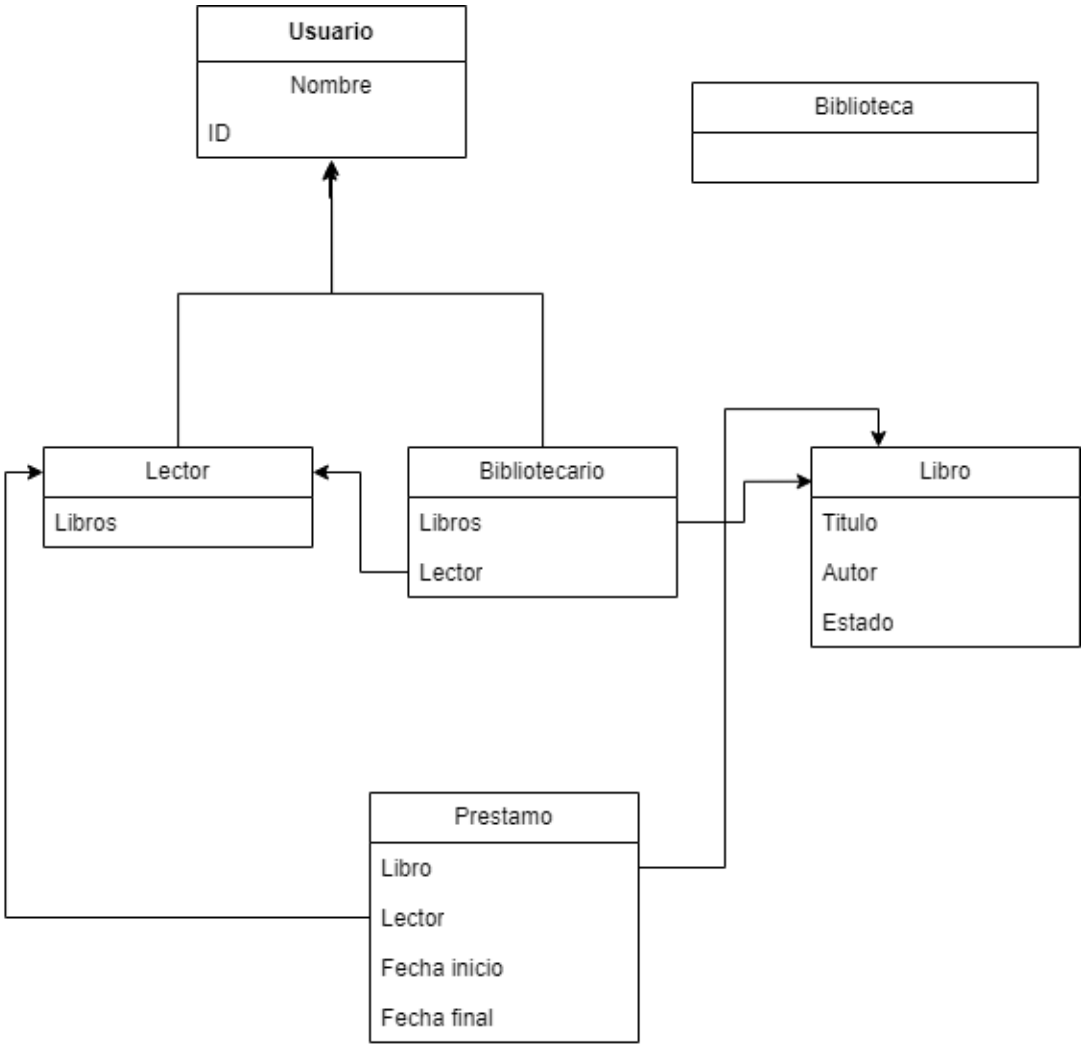

Figure 2.1: Example figure.

3. RESULTADOS

1. Refinamiento de Funcionalidades

- Gestión de Préstamos: Se ha implementado con éxito una función clave que impide la creación de múltiples préstamos del mismo libro mientras haya uno activo. Esto asegura la coherencia y previene la sobre asignación de recursos.
- Validaciones de Proceso: La introducción de una validación para confirmar la presencia de al menos un bibliotecario antes de crear un préstamo mejora la responsabilidad y el control en el proceso de préstamo, garantizando que las operaciones no puedan proceder sin la supervisión adecuada.

2. Impacto Operacional

- Optimización del Flujo de Trabajo: Estas funciones no solo optimizan el flujo de trabajo de la biblioteca, sino que también mejoran la experiencia del usuario al asegurar la disponibilidad de los recursos y el cumplimiento de las políticas de la biblioteca.
- Fiabilidad del Sistema: La inclusión de estas salvaguardas operativas contribuye a la fiabilidad y robustez del LMS, demostrando la capacidad del sistema para manejar reglas de negocio complejas.

3. Conclusión Final

- Satisfacción de Requisitos: El LMS desarrollado satisface los requisitos establecidos al inicio del proyecto, ofreciendo un sistema sólido y bien estructurado que aborda los desafíos de gestión de una biblioteca pública grande.

BIBLIOGRAPHY

[1] Apache Software Foundation, https://maven.apache.org/guides/index.html, Maven Documentation, accessed: 2023-11-29.

[2] Red Hat, Inc., https://hibernate.org/orm/documentation/5.4/, Hibernate ORM Documentation, accessed: 2023-11-29.

[3] Pivotal Software, https://docs.spring.io/spring-data/jpa/docs/current/reference/html/, Spring Data JPA Reference Documentation, accessed: 2023-11-29.

[4] W. Deblauwe, Learn Spring Boot with Thymeleaf, Self-Published, https://www.wimdeblauwe.com/books/taming-thymeleaf/, 2023, accessed: 2023-11-29.

[5] Bootstrap, https://getbootstrap.com/docs/5.3/getting-started/introduction/, Bootstrap Documentation, accessed: 2023-11-29.

[6] npm, Inc., https://docs.npmjs.com/, npm Documentation, accessed: 2023-11-29.

[7] Pivotal Software, https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/, Spring Boot Reference Documentation, accessed: 2023-11-29.