
UAX
Universidad
Alfonso X el Sabio
UAX
Universidad
Alfonso X el Sabio
UAX
Universidad
Alfonso X el Sabio

Informe del Proyecto de Biblioteca con JPA

Luis Miguel Urbez

November 30, 2023

1 Introducción

Este documento presenta una visión general del proyecto de desarrollo de la aplicación para la gestión de la biblioteca. La aplicación tiene como objetivo principal proporcionar una plataforma eficiente para la administración de libros y préstamos. Todo esto desde la perspectiva de tres diferentes roles; el administrador, el bibliotecario y el lector.

2 Desarrollo del Proyecto

Durante el desarrollo del proyecto, se utilizaron diversas tecnologías y herramientas, incluyendo Java y Spring Boot para el backend, Thymeleaf para la interfaz de usuario, y PostgreSQL como base de datos. Para realizar el proyecto he usado el IDE de Intellij de Jetbrains el cual ha dido muy util para realizar el proyecto por la facilidad y comodidad que brinda a la hora de programar.

Para realizar el proyecto tambien he usado aplicaciones de inicializacion como bootify y Spring Inizialicer. Aplicaciones como Docker Descktop por parte de el uso de un docker en mi aplicacion y de psAdmin4 (aplicación oficial de postgre) para controlar la base de datos.

3 Arquitectura del Proyecto

La arquitectura de microservicios es un enfoque de desarrollo de software en el que una aplicación se construye como un conjunto de servicios independientes, pequeños y autónomos, que se comunican entre sí a través de APIs (Interfaces de Programación de Aplicaciones). Cada servicio aborda una función de negocio específica y se puede desarrollar, implementar y escalar de forma independiente.

Aquí hay algunos conceptos clave asociados con la arquitectura de microservicios:

1. Descomposición en Servicios:
    - Las aplicaciones monolíticas se dividen en servicios más pequeños y manejables.

- Cada servicio se centra en una tarea específica o una función de negocio.

2. Independencia y Despliegue Continuo:
   - Cada servicio es independiente y puede desarrollarse y desplegarse de forma independiente.
   - Esto permite actualizaciones más rápidas y menos riesgosas.
3. Comunicación entre Servicios:
   - Los servicios se comunican a través de API, generalmente utilizando protocolos HTTP/REST o mensajería.
   - La comunicación es ligera y basada en estándares.
4. Escalabilidad y Tolerancia a Fallos:
   - Cada servicio puede escalarse de forma independiente según la demanda.
   - La arquitectura está diseñada para ser tolerante a fallos, ya que la falla de un servicio no debería afectar a otros.
5. Gestión de Datos:
   - Cada servicio puede tener su propia base de datos.
   - Puede haber bases de datos especializadas para diferentes tipos de datos o funciones.
6. Facilita la Tecnología y la Elección de Herramientas:
   - Diferentes servicios pueden utilizar diferentes tecnologías según sus requisitos.
   - Esto permite la elección de la mejor herramienta para cada tarea específica.
7. Desarrollo Ágil y Equipos Autónomos: Equipos pequeños y autónomos pueden trabajar en servicios específicos sin interferencias.
8. Monitorización y Descubrimiento de Servicios: Herramientas de monitorización y descubrimiento de servicios son esenciales para gestionar un sistema de microservicios.

La arquitectura de microservicios ofrece flexibilidad, escalabilidad y facilita la entrega continua. Sin embargo, también presenta desafíos, como la gestión de la complejidad de las interacciones entre servicios y la necesidad de herramientas de orquestación eficientes. Cada organización debe evaluar si la arquitectura de microservicios es la mejor opción para sus necesidades particulares

4 Funcionalidades Clave

La aplicación incluye funcionalidades clave como la gestión de libros, préstamos, transacciones, y otras operaciones administrativas. La utilizacion de roles es clave para el funcionamiento de la aplicacion, y aunque no estan realizados de forma profesional en esta primera version (usando un inicio de sesion dependiendo del rol que quieras obtener), esta primera implementaion de los mismos desde el inicio de la aplicacion da a entender el modelo que se busca con esta web.

Partimos de el modelo basico de capas que se puede enccontrar en la programacion concurrente con Spring Boot, el cual determina diferentes capas en este caso que definen las posibilidades de cada Rol.

La primera de ellas y mas importante, el administrador. Este es el maximo dueño de la biblioteca y tiene control sobre los bibliotecarios, pudiendo contratar y despedir nuevos bibliotecarios. Tambien es el encargado de realizar los pedidos para reparar los libros que estan dañados. Al ser la maxima figura tambien es el enargado de administrar las politicas que se pueden hacer sobre los prestamos. Aunque estas solo tienen una funion documentativa en mi codigo (ya que no se ha implementado la logica para que el tiempo pase en la web), seran las enargadas de apliar las normas necesarias para que los lectores no puedan abusar a la hora de pedir libros prestados.

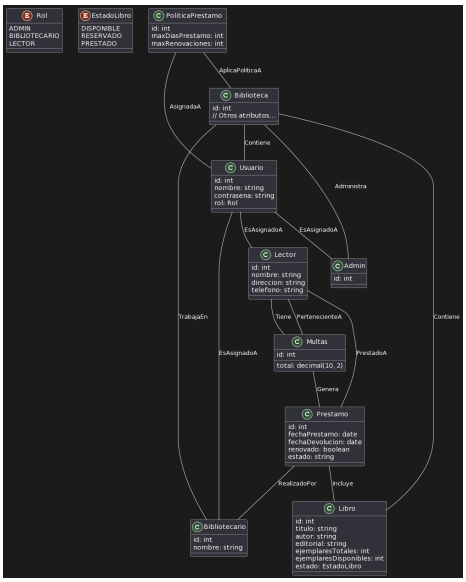
En la siguiente capa de permisos encontramos al biblioteccario. Esta es una figura que tiene total control sobre los letores y los libros que estos quieren usar. Este puede añadir nuevos lectores, crear los prestamos, añadir nuevos libros y por supuesto comprobar las politicas que estan disponibles para la realizacion de los prestamos. Los bibliotecarios son los encargados de realizar todas las tareas de los letores, sin embargo, se les ha aplicado una logica para que estos no puedan realizar tareas incorrectas, como prestar libros de los cuales no hay existencias, añadir y quitar la cantidad de libros que hay en la biblioteca una vez se han prestado y demas implementaiones logicas. Tambien se ha creado la parte de multas, estas al igual que las politicas de prestamo, son solo doumentativas y no se pueden crear de forma realista ya que no hay un contador de tiempo que pase, pero en cuanto esa funcion sea implementada los lectores seran multados por retrasarse en la devolucion de sus prestamos.

Para terminar nuestra "piramide" de permisos, el ultimo de todos es el lector. El cual solo puede limitarse a pedir prestamos de libros y leerlos en el tiempo establecido por el bibliotecario, el lector puede devolver el libro roto, por lo cual obtendra una multa. El lector tambien podra pagar por su parte las multas que tenga por no entregar el libro de un prestamo a tiempo. Este tambien puede ver las politicas de los prestamos estableccidas por el administrador de la biblioteca.

Fuera de la logica y entrando mas en la forma de estructurar el proyecto encontramos una base de datos relacional creada con postgre el cua nos proporciona una aplicacion (pgAdmin4) para la edicion de la misma. El modelo de esta base se puede encontrar de forma grafica en el Read.me del repositorio de github asociado al proyecto, al igual que el enunciado del mismo (https://github.com/LuisMi01/biblioteca$_{L}uisMiguelUrbez$)

Por parte de la logica de entidades y de clases encontramos un modelo vista controlador adaptado a la creacion de una aplicacion web con springboot, la incorporacion de fragments en el htmlx ha sido muy importante para la resoluicion de muchas partes visuales, al igual que la facil exploracion y cambios en el codigo visual gracias a la distribucion de los packages en el apartado resources del proyecto. El modelo de entidades y relaciones se puede encontrar tambien el repositorio de forma visual para tener mas en claro como ha sido la creacion de este proyecto a la hora de la logica entre entidades y metodos.

Diagrama de clases:



5 Conclusiones y Resultados

La aplicación proporciona una solución integral para la gestión de la biblioteca, permitiendo a los usuarios administrar libros y realizar préstamos de manera eficiente. Aunque no ha dado tiempo a implementar toda la logica que se le queria implementar en futuras actualizaciones se le va a aportar nuevas incorporaciones como una logica basada en concurrencia usando hilos y la creacion de la pagina con un modelo mas responsivo usando GraphQL. Estoy satisfecho con la relizacion de esta primera version de mi biblioteca y en poco recibira grandes actualizaciones para la mejoria de la misma.

6 Referencias

- https://maven.apache.org/guides/index.htmlMaven docs

- https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/Spring Boot reference
- https://docs.spring.io/spring-data/jpa/docs/current/reference/html/Spring Data JPA reference
- https://www.thymeleaf.org/documentation.htmlThymeleaf docs
- https://webpack.js.org/concepts/Webpack concepts
- https://docs.npmjs.com/npm docs
- https://getbootstrap.com/docs/5.3/getting-started/introduction/Bootstrap docs
- https://htmx.org/docs/Htmx in a nutshell
- https://www.wimdeblauwe.com/books/taming-thymeleaf/Learn Spring Boot with Thymeleaf