Bookstore Web App

Name: Zaharie Oana Denisa
Group: 30234

Table of Contents

Deliverable 1 3

Project Specification 3

Functional Requirements 3

Use Case Model 1 3

Use Cases Identification 3

UML Use Case Diagrams 4

Supplementary Specification 4

Non-functional Requirements 4

Design Constraints 4

Glossary 5

Deliverable 2 5

Domain Model 5

Architectural Design 6

Conceptual Architecture 6

Package Design 6

Component and Deployment Diagram 7

Deliverable 3 8

Design Model 8

Dynamic Behavior 8

Class Diagram 9

Data Model 10

System Testing 10

Future Improvements 10

Conclusion 11

Bibliography 11

Deliverable 1

Project Specification

The Bookstore web application aims to provide an online platform for users to search and purchase books. Users should be able to create accounts, add books to their cart, and complete purchases securely. Additionally, the application will allow administrators to manage the book inventory, user accounts and sales.

Functional Requirements

- User Registration: users can create accounts and log in with their credentials
- Book Searching: users can browse through available books
- Cart Management: users can add or remove books from their shopping cart and view it
- Place order: users can place an order
- Administrators can manage books by adding new ones or updating the existing books
- Administrators can add new book types
- Administrators can view all sales, books, and users

Use Case Model 1

Use Cases Identification

Use-Case: Search Books

Level: user

Primary Actor: client

Main success scenario:

- User selects filters to search for books
- System retrieves relevant books based on the search criteria
- User views the list of search results
- User selects a book from the list to view details

Extensions:

- If no matching books are found, the system displays a message

Use-Case: Add to Cart

Level: user

Primary Actor: client

Main success scenario:

- User clicks on the "Buy Book" button
- System adds the selected book to the user's shopping cart

Extensions:

- If the selected book is out of stock, the system displays a message

Use-Case: View Sales

Level: user

Primary Actor: administrator

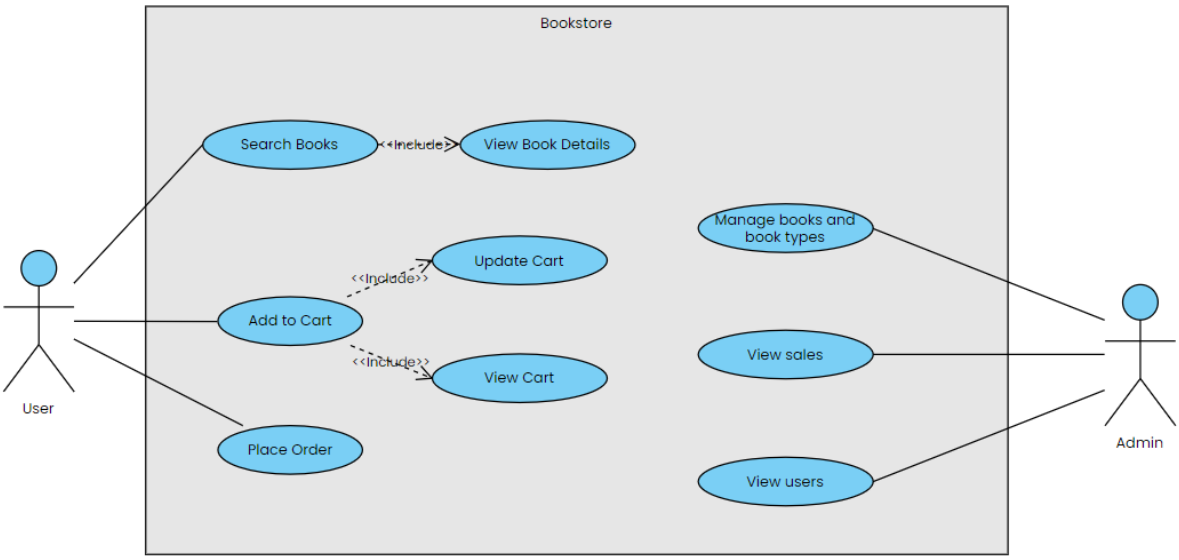
Main success scenario:

- Administrator clicks on the “Download sales report” button
- The sales report is downloaded as an XML
- Administrator views the sales report

Extensions:

- If there are no sales data available, the system displays a message

UML Use Case Diagrams



Supplementary Specification

Non-functional Requirements

1. Performance: The system should be able to handle a large number of concurrent users without a loss in performance and the response time for user actions should be under 2 seconds. This is important so that users don’t give up on using the app due to a long waiting time.
2. Security: Users data should be securely stored, to avoid personal data theft
3. Usability: the user interface should be intuitive and user-friendly, for an enjoyable experience
4. Portability: The application should be accessible across different devices and screen sizes. It should not be dependent on the operating system or browser used, so that no one is restricted from using the app

Design Constraints

1. Technology Stack:
    - The application will be developed using Java Spring Boot framework
    - Front-end components will be implemented using React
    - Spring Security will be used for authentication and authorization purposes.
    - Data will be employed for seamless integration with the MySQL database.

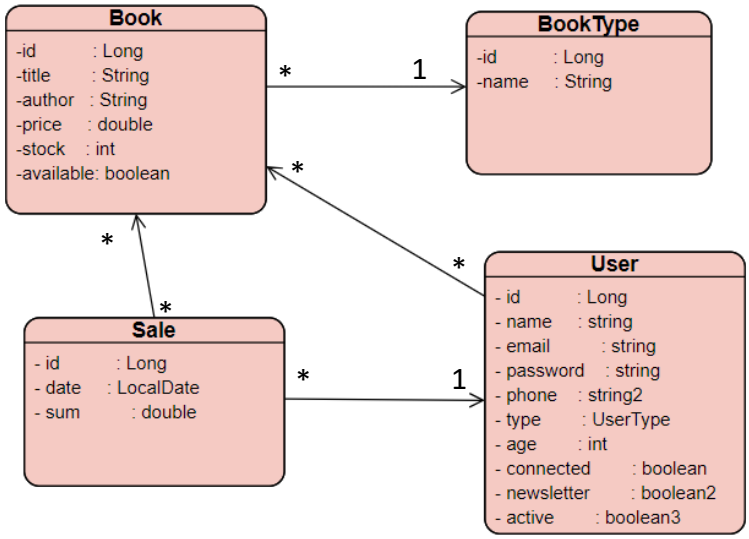
2. Architectural Constraints:
    - The application will adhere to a Layered Architecture for clear separation of concerns
    - The application will utilize Hibernate ORM for database interaction, ensuring efficient data access and management

Glossary

- Layered Architecture: one of the most common architectural styles, where the modules or components with similar functionalities are organized into horizontal layers
- Authentication and Authorization: both are important for the systems and information security. Authentication verifies the identity of a user and authorization determines their access rights
- Hashing: the process of transforming any given key or a string of characters into another corresponding value. For this project I used the SHA-256 algorithm

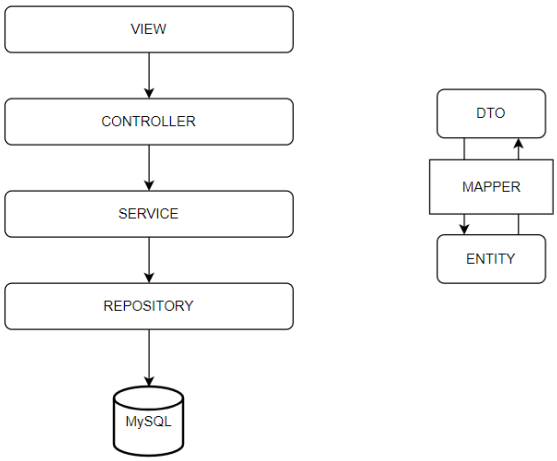
Deliverable 2

Domain Model



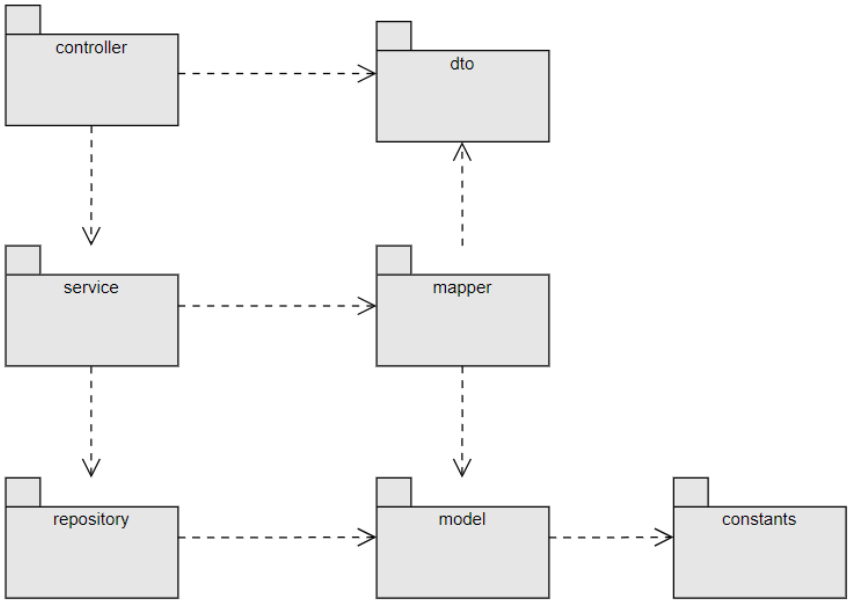
Architectural Design

Conceptual Architecture

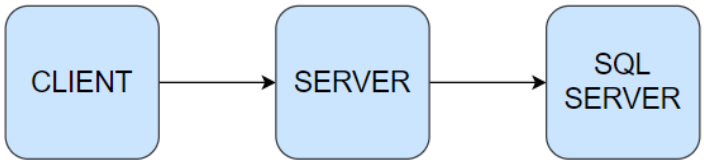


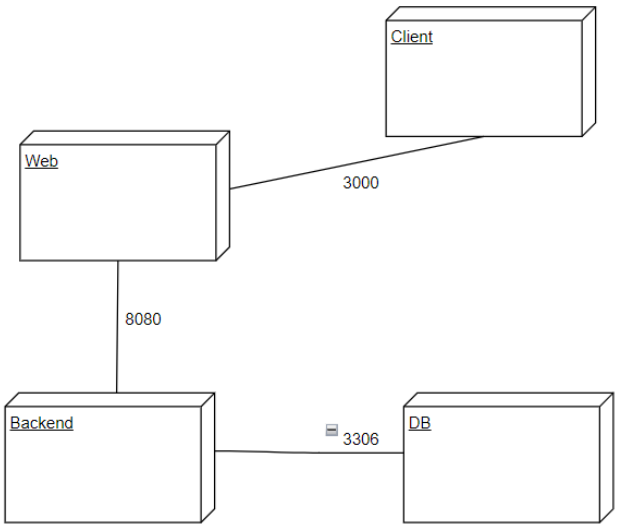
For this project, I used a layered architecture. It helps improve the maintainability, scalability and testability of the code. It also facilitates the reuse of existing components and reduces the coupling of the classes.

Package Design



Component and Deployment Diagram

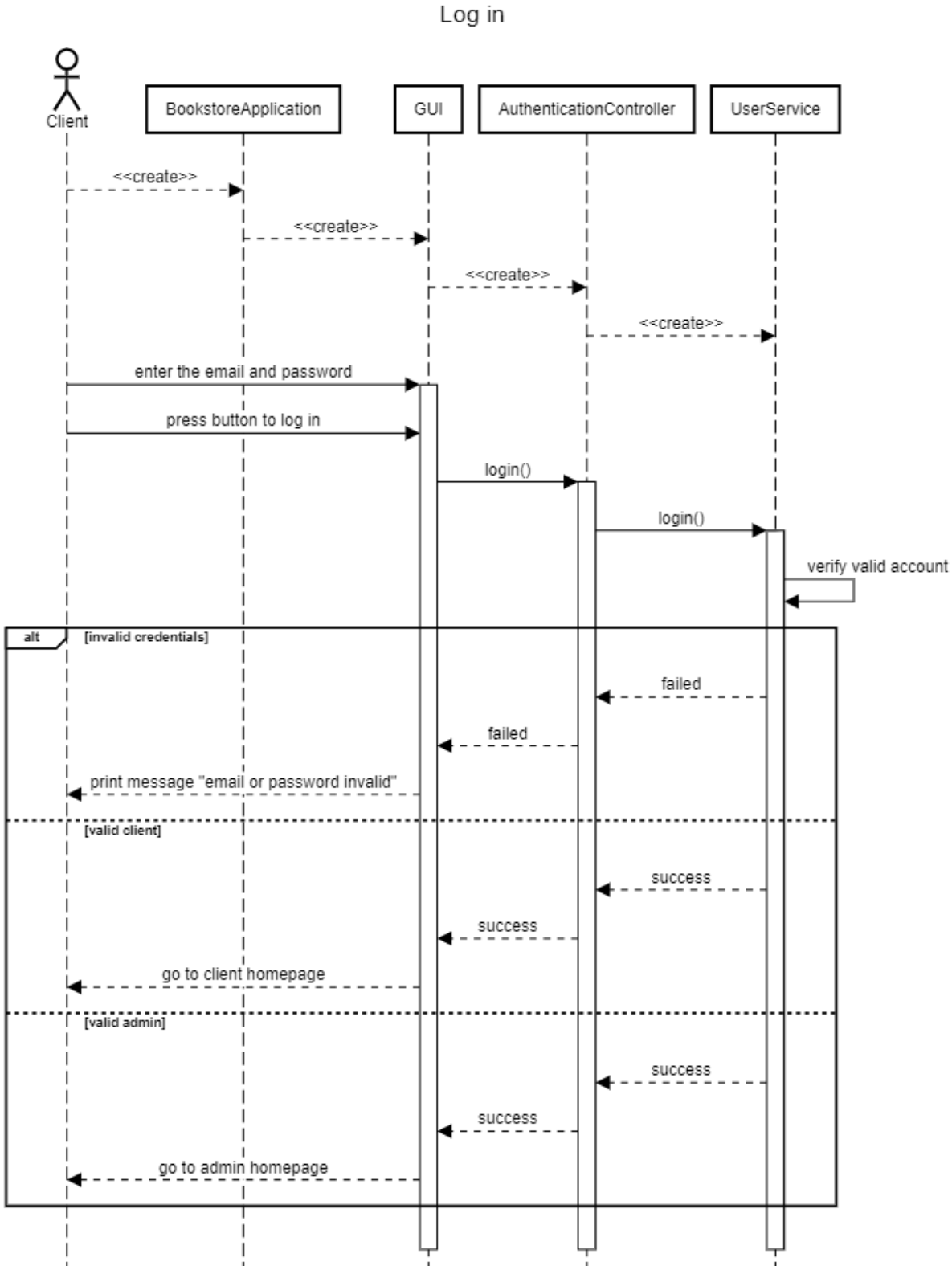


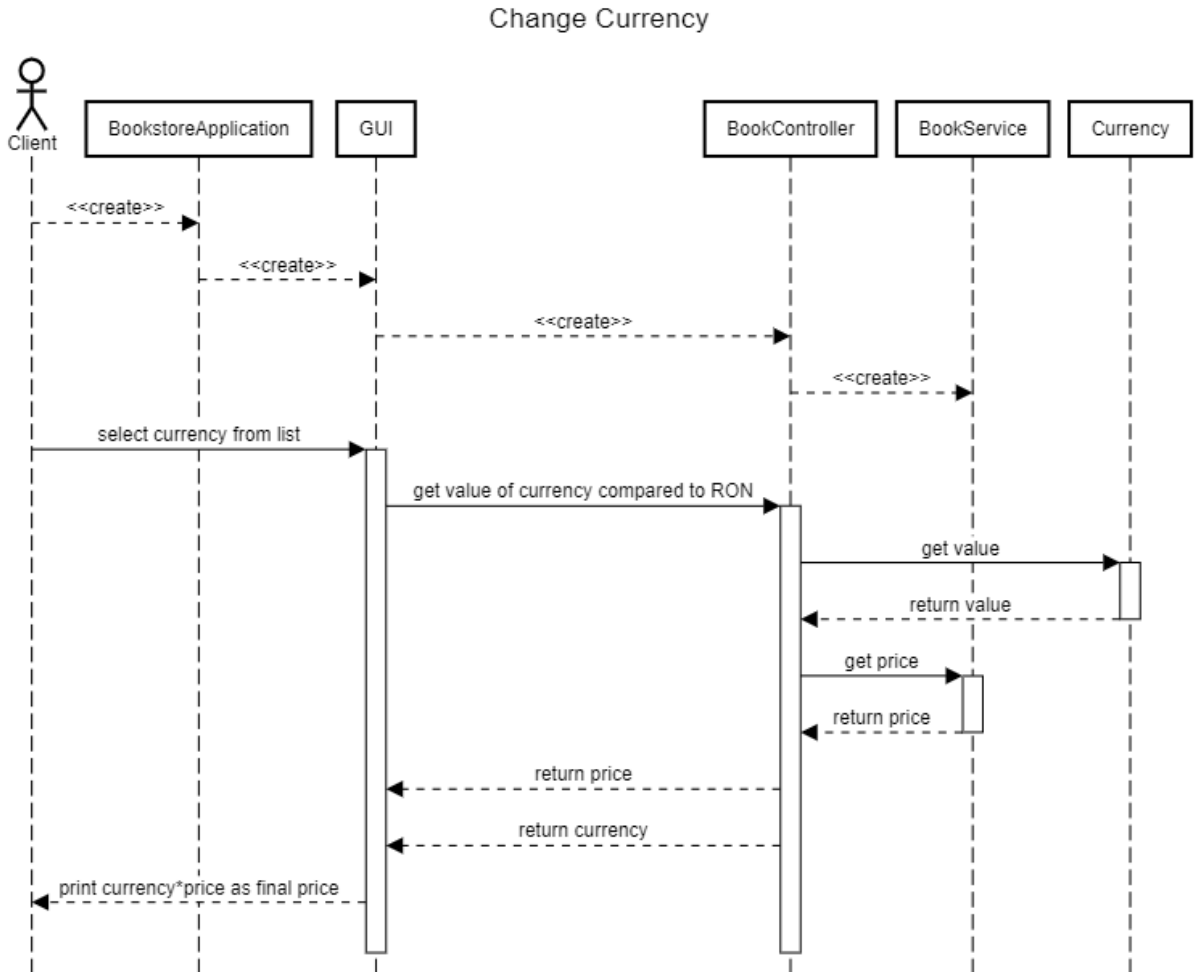


Deliverable 3

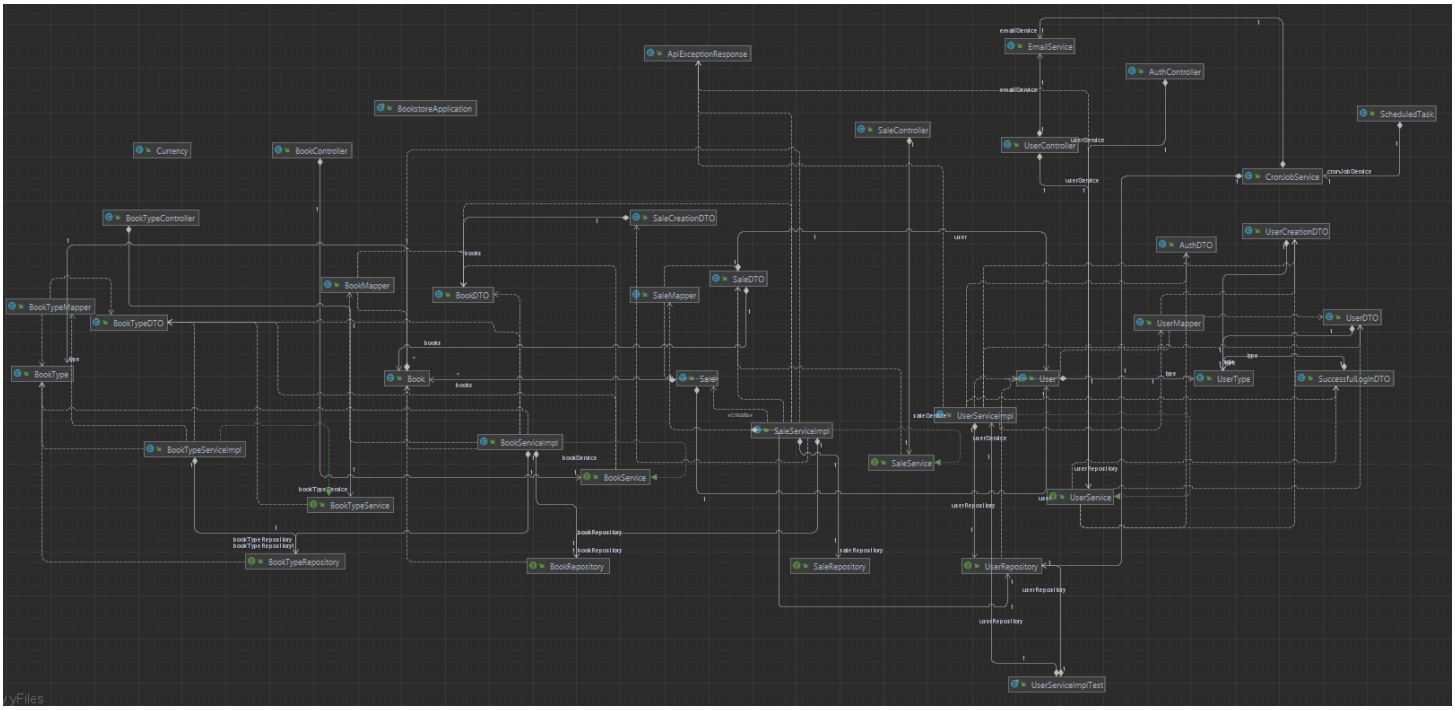
Design Model

Dynamic Behavior





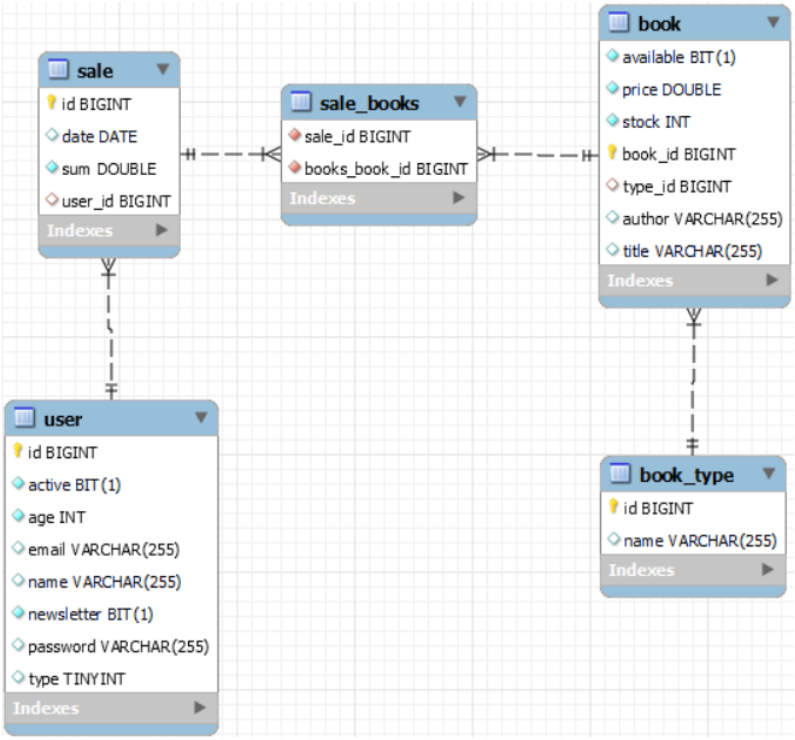
Class Diagram



Design patterns:

- Builder: for models and dtos, as they have a lot of features and every combination of constructor parameters would be hard to write
- Singleton: for the currency class, in order to create only one instance of the class, as it contains constants

Data Model



System Testing

Firstly, for testing the services, I used JUnit tests for the user service for:

- Creating a new user
- Finding a user by email
- Deleting a user by email

Secondly, I tested the functionalities through the user interface such as: create an account, log in, view books / users, etc.

Future Improvements

- Add pictures for every books
- Add the possibility to visit the website without using an account
- Saving multiple addresses and select which one to use at every order, or add a new one
- Add multiple ways of paying

Conclusion

To sum up, this project is a well-structured bookstore web application, open for any new improvements. It comes with many use cases and incorporates all functional and non-functional requirements. Considering the technologies, it uses Java Spring Boot for Backend and ReactJS for frontend.

Bibliography

- https://mailtrap.io/blog/spring-send-email
- https://mailtrap.io/blog/sending-email-using-java
- https://medium.com/@bectorhimanshu/how-to-create-a-scheduled-task-using-a-cron-job-in-spring-boot-a1987e679d60
- https://www.baeldung.com/spring-scheduled-tasks