Book Application

Student: Gliga-Hambet Bogdan Andrei

**Group:30431**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

The designed application is designed for the front desk employees of a book. It is designed to allow different types of users manage their accounts, perform actions on books.

# Functional Requirements

The regular user can perform the following operations:

* Search books by genre, title, author.
* Sell books.

The administrator can perform the following operations:

* CRUD on books (book information: title, author, genre, quantity, and price).
* CRUD on regular users’ information.
* Generate two types of reports files, one in pdf format and one in csv format, with the books out of stock.

# Non-functional Requirements

The application is a web-app which can be accessed via the browser, as long as the user has an active interned connection. It must have persistent storage, in this case XML files, in which the data is stored such that it is not lost.

2. Use-Case Model

C:\Users\GligaBogdan\Downloads\Untitled Diagram.png

Use case: Generate Report File in PDF and CSV

Level: User-goal level

Primary actor: Application Administrator

Main success scenario: Login with user having permission rights => View all books => Click “Generate report file” button

Extensions: Failure in case of corrupted XML files

3. System Architectural Design

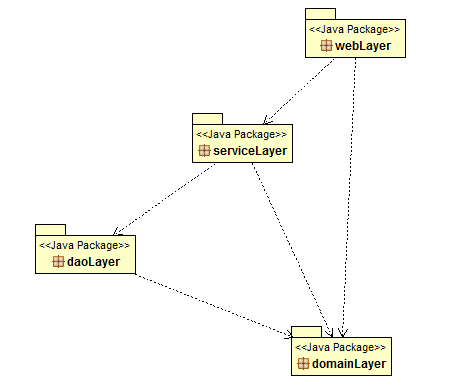
**3.1 Architectural Pattern Description**

The architectural pattern used in this application is the “MVC Pattern”. MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

* **Model** - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.
* **View** - View represents the visualization of the data that model contains.
* **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

**3.2 Diagrams**

More specific, the layers used in this application were the: presentation-layer ( consisting of JSPs), web-layer (containing all the Spring Controllers), service-layer (containing the Spring Services handling the business logic), the dao-layer (containing DAOs for XML interaction) and the domain-layer ( being consisted of simple POJOs)



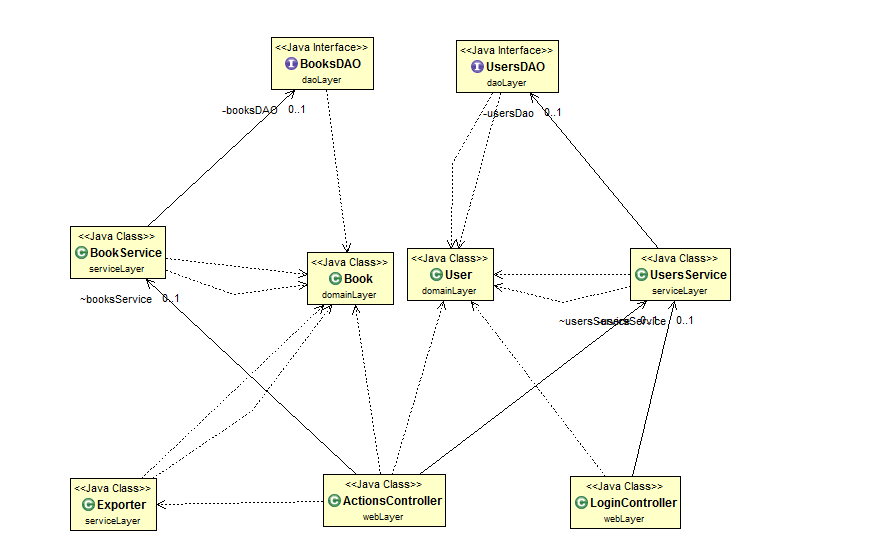
4. UML Sequence Diagrams

This is the sequence diagram for generating a report by the administrator for creating a new book as administrator or the aplication

C:\Users\GligaBogdan\Downloads\Untitled Diagram (1).png

5. Class Design

**5.1 UML Class Diagram**



6. Data Model

**6.1 User**

This is used to model a real time user of the application. It has different fields on it such as the name, username, password, etc. It is the main model associated with each application user and it is by its attributes that the level of operations he can perform is decided.

**6.2 Book**

The account class represents a real life Book. It has internal data such as title, genre, auithor, quantity, price

**6.3 SearchQuery**

This object models and encapsulates the query a user makes when searching for a particular book. U user can now test after title, genre or author.

7. System Testing

The application has been tested using JUnit tests, as they provide an easy, robust and reliable method to test your application. The main features that have been testes are the classes in the DAO Layer, as they handle the database operations and they are the most prone to fail.

A Test Suite has been designed for each of the 2 DAO classes, namely: BookDAOTests, and UsersDAOTests. Each one of these has methods which independently test CRUD operations which are performed by the application. Before each test, the database is cleaned and populated with validation data and after each test the database is cleaned again in order to leave it in a stable state and not affect other Junit tests

8. Bibliography

<http://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/htmlsingle/spring-framework-reference.html>

<http://hibernate.org/orm/documentation/5.1/>

<http://docs.oracle.com/javase/7/docs/technotes/guides/xml/jaxb/>

<http://developers.itextpdf.com/frequently-asked-developer-questions>

<http://javacsv.sourceforge.net/>