<Assignment A2>
Analysis and Design Document

Student:Budusan Ciprian
Group:30434

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 4
3. System Architectural Design 4
4. UML Sequence Diagrams 6
5. Class Design 6
6. Data Model 7
7. System Testing 8
8. Bibliography 8

1. Requirements Analysis

1.1 Assignment Specification

Use Swing/C# API to design and implement an application for the employees of a book store. The application should have two types of users (a regular user represented by the book store employee and an administrator user) which have to provide a username and a password in order to use the application.

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, price).
- CRUD on regular users' information.
- Generate a report into a *txt* or *xml* file with the books out of stock.

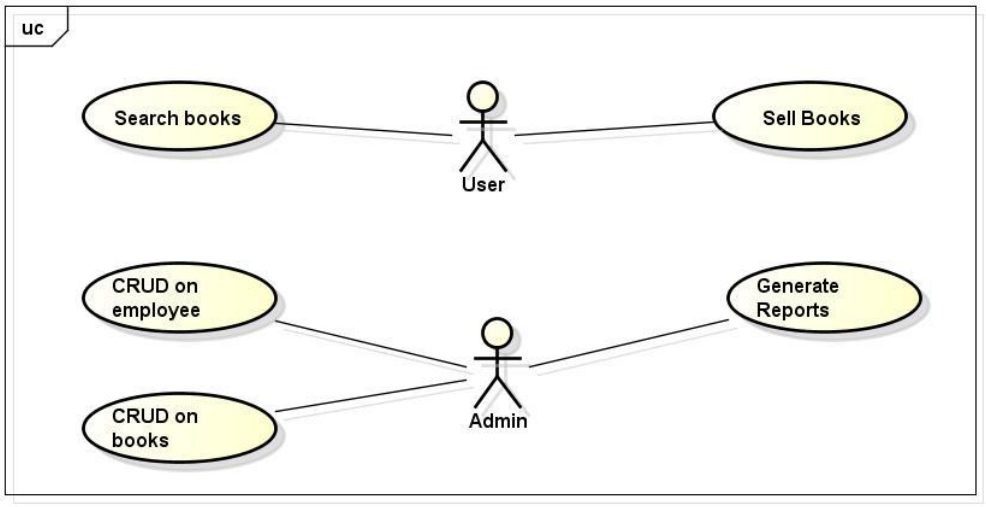
1.2 Functional Requirements

- Authentication: two types of users: regular user and administrator;
- Authorization levels: admin role separated from user role;
- Administrative functions: perform CRUD operations on books and employees;
- Sell books;
- Search books by title, author, genre;
- Generate reports;
- The information about users, books and selling are stored in multiple XML files;
- Factory Method design pattern used for generating the reports;

1.3 Non-functional Requirements

- Performance: response time immediately;
- Availability: all the time;
- Security: obtained from login interface using Spring Security with different user roles;
- Recoverability: backup for the database;

2. Use-Case Model



Use case: Search books
Level: user-goal level
Primary actor: regular user
Main success scenario:

1. User logs in with the provided username and password.
2. User selects Search from menu.
3. User enters the title, author or genre to search for and submits the result.
4. The list of books which match the entered String are displayed.

Extensions: 4a. The entered String is empty.
-4a1. System does not display anything.

3. System Architectural Design

3.1 Architectural Pattern Description

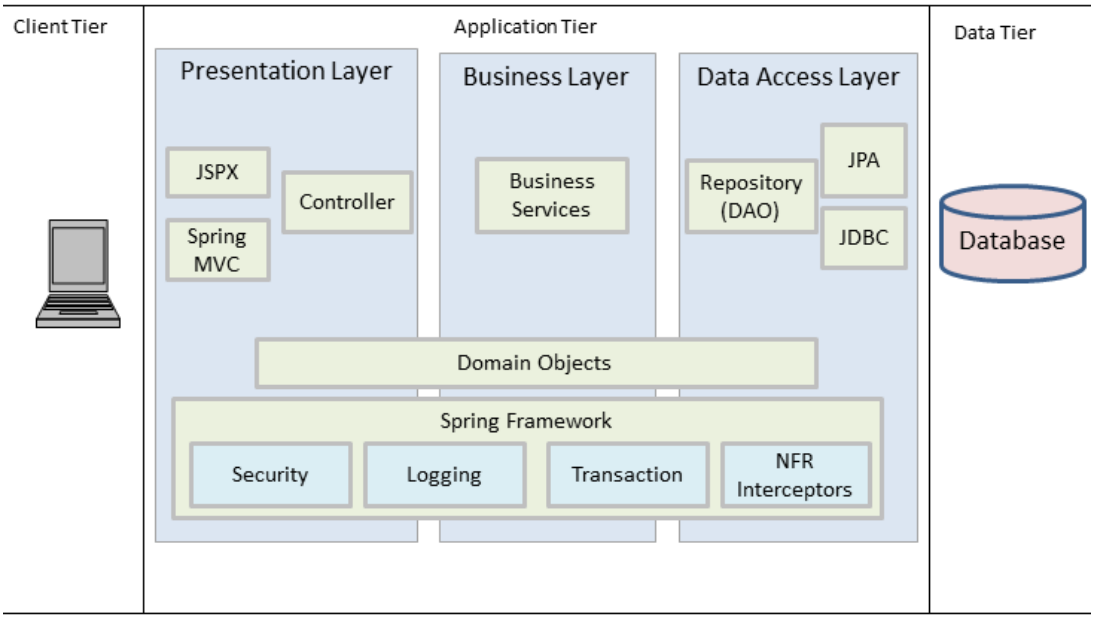
The DAO layer contains queries and updates to save the domain layer into the database (this time the database consists of XML files, each representing a table). The queries and updates are performed using JAXB where the objects are "marshalled" from files. Interfaces are used to abstract DAO away from the actual datastore in case I want to change datastores.

The Service layer contains the business logic and orchestrates the interaction between the domain layer and the DAOs.

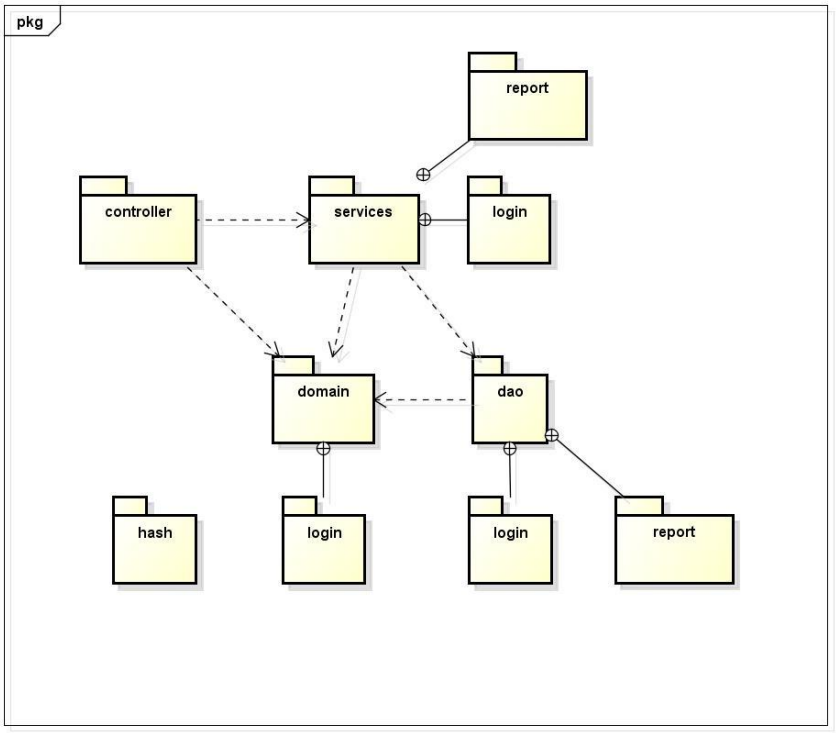
Also Spring MVC is used to ensure the Model-View-Controller pattern.

For the front-end part, view level, JSP files are used, with the help of JSTL Library for being able to use C-like syntax.

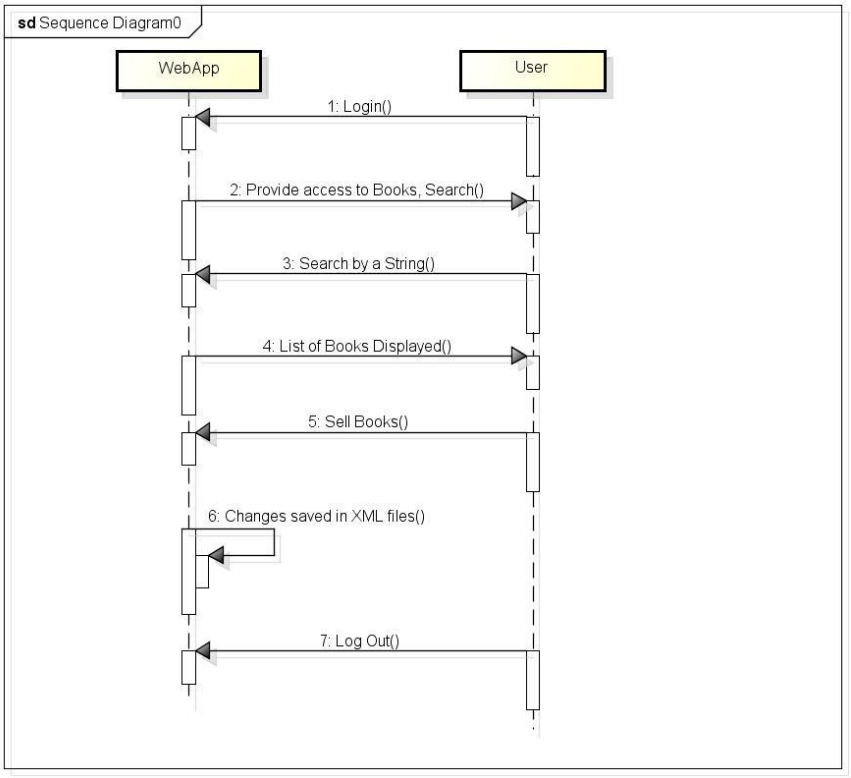
3.2 Diagrams



Package Diagram:



4. UML Sequence Diagrams



5. Class Design

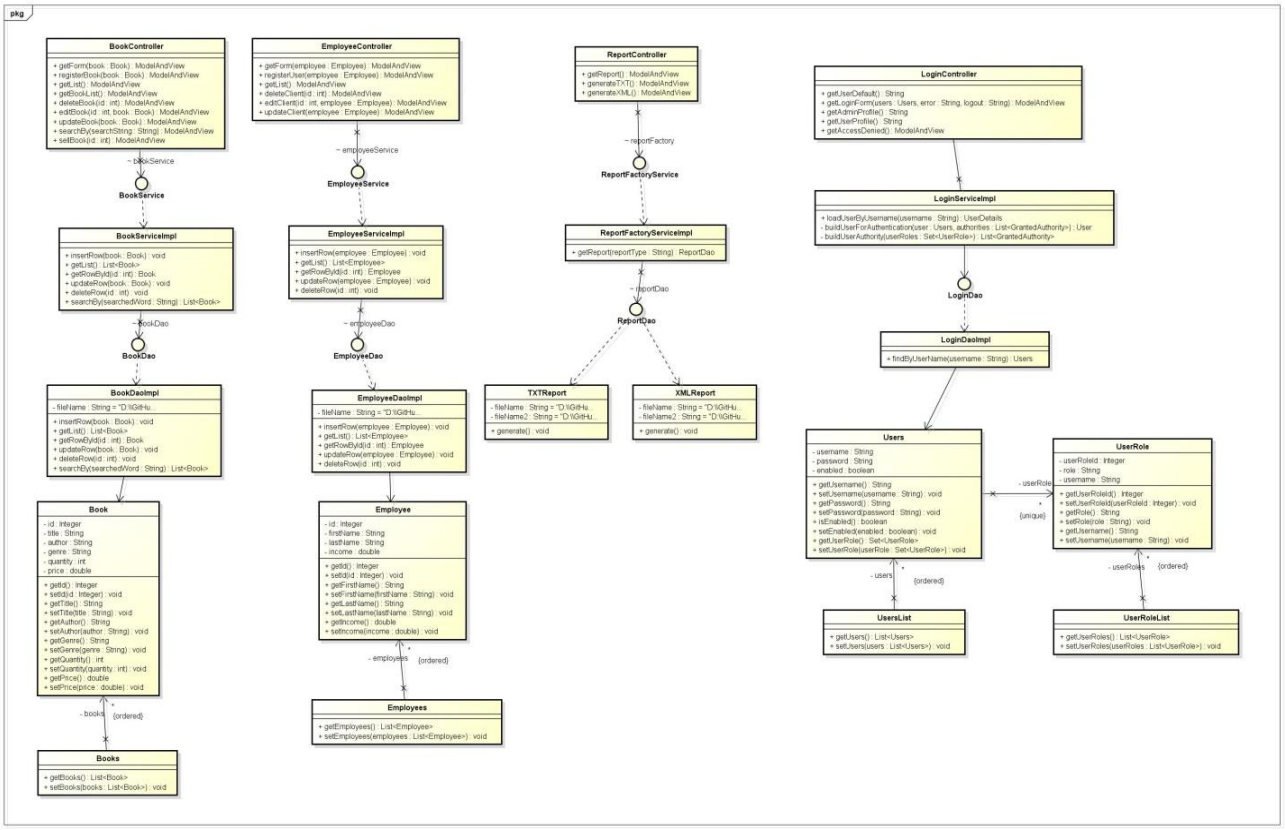
5.1 Design Patterns Description

Spring MVC uses the following patterns:

- Dependency injection/ or IoC (inversion of control) – Is the main principle behind decoupling process that Spring does;
- Factory – Spring uses factory pattern to create objects of beans using Application Context reference;
- Proxy – used heavily in AOP, and remoting;
- Singleton – by default, beans defined in spring config file (xml) are only created once. No matter how many calls were made using getBean() method, it will always have only one bean. This is because, by default all beans in spring are singletons.

- Factory Method design pattern used for generating the reports into txt or xml.

5.2 UML Class Diagram



6. Data Model

An example of how the books are stored in an xml file:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<books>
    <book>
        <id>1</id>
        <title>Godfather</title>
        <author>Mario Puzo</author>
        <genre>Thriller</genre>
        <quantity>0</quantity>
        <price>100.99</price>
    </book>
    <book>
        <id>2</id>
        <title>The Castle</title>
        <author>Franz Kafka</author>
        <genre>Fantasy</genre>
        <quantity>9</quantity>
        <price>50.45</price>
    </book>
    <book>
        <id>3</id>
        <title>Robinson Crusoe</title>
        <author>Daniel Defoe</author>
        <genre>Adventure</genre>
        <quantity>1</quantity>
        <price>40.01</price>
    </book>
</books>
```

7. System Testing

The Spring MVC Test framework provides first class JUnit support for testing client and server-side Spring MVC code through a fluent API. Typically it loads the actual Spring configuration through the TestContext framework and always uses the DispatcherServlet to process requests thus approximating full integration tests without requiring a running Servlet container.

Client-side tests are RestTemplate-based and allow tests for code that relies on the RestTemplate without requiring a running server to respond to the requests.

8. Bibliography

http://howtodoinjava.com/2013/07/30/jaxb-exmaple-marshalling-and-unmarshalling-list-or-set-of-objects/
http://stackoverflow.com/questions/12323397/how-to-marshalling-the-muliptle-object-using-jaxb
http://www.bookdepository.com/