Book Store

Student:Milas Bogdan-Adrian

**Group:30233**

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

Use Java/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.

# 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:

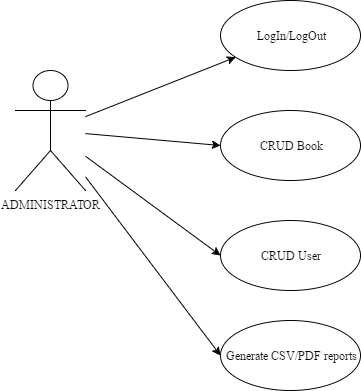
* CRUD on books (book information: title, author, genre, quantity, and price).
* CRUD on regular users’ information.

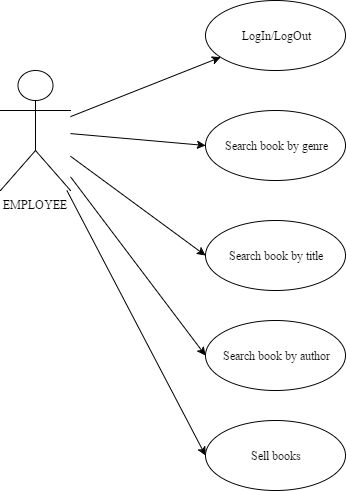
Generate types of reports files, one in csv format, with the books out of stock.

# Non-functional Requirements

* Generarea raportului PDF nu este implementata.
* Exista posibilitatea de generare a mai multor tipuri de rapoarte.

2. Use-Case Model





3. System Architectural Design

**3.1 Architectural Pattern Description**

Am folosit Pattern-ul Architectural Model View Controller sau MVC Pattern.

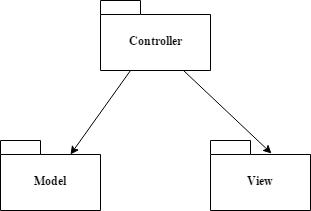
Model-ul se ocupa de comportarea si datele aplicatiei; raspunde la cereri despre starea sistemului, la cereri de schimbare de stare si notifica utilizatorul atunci cand aceste schimbari au avut loc pentru ca acesta sa poata reactiona.

View-ul transpune model-ul intr-o forma care permite o interactionare usoara, in mod tipic o interfata vizuala. Pot exista multiple view-uri pentru un singur model pentru scopuri diferite.

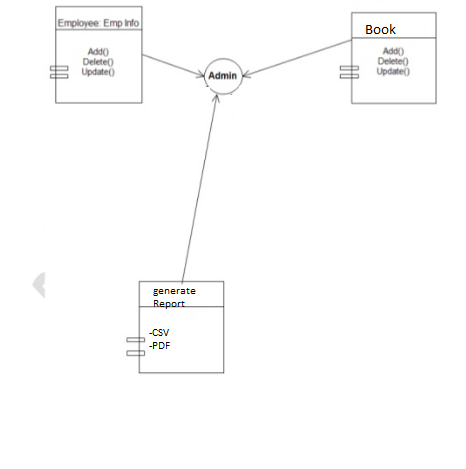
Controller-ul primeste input de la utilizator si initiaza un raspuns in urma cererilor catre obiectele model. Controller-ul este cel care controleaza celelalte doua clase de obiecte, view si model, instructandu-le sa execute operatii pe baza input-ului primit de la utilizator.

**3.2 Diagrams**

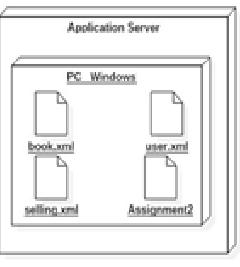
**P**ackage Diagram



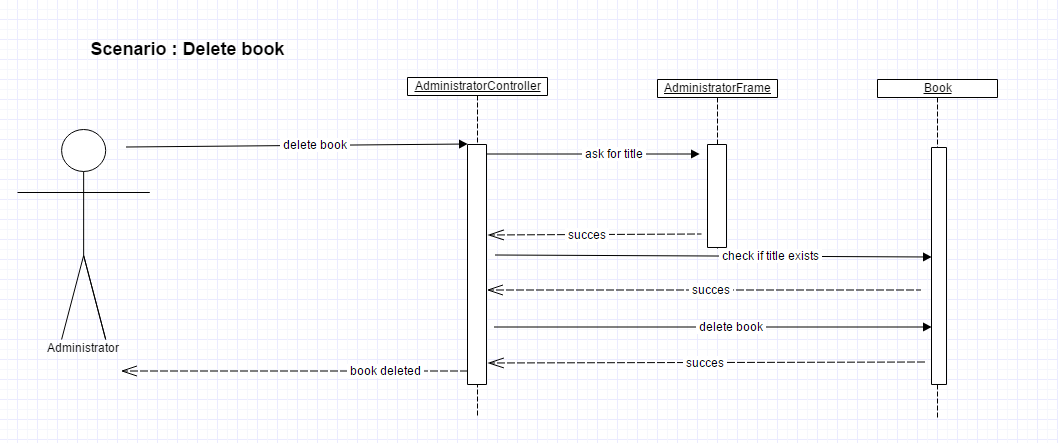
Component Diagram:



Deployment Diagram:



4. UML Sequence Diagrams

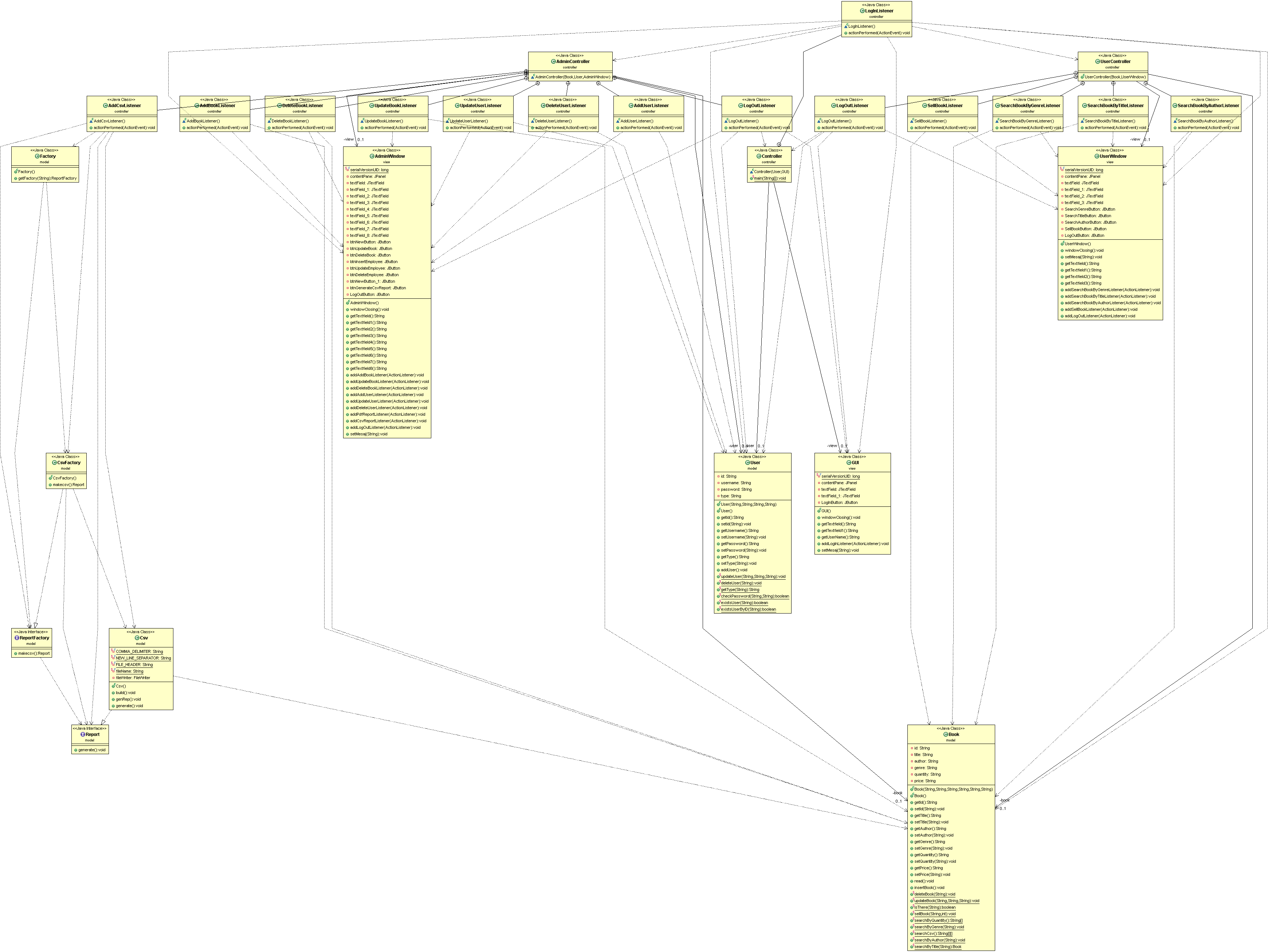


5. Class Design

**5.1 Design Patterns Description**

La crearea acestei aplicatii am folosit Factory Pattern pentru rapoartele cartilor care sunt out of stock. Patternul Factory face parte din categoria *Creational Patterns* si ca atare rezolva problema crearii unui obiect fara a specifica exact clasa obiectului ce urmeaza a fi creat. Acest lucru este implementat prin definirea unei metode al carei scop este crearea obiectelor. Metoda va avea specificat ca parametru de intors in antet un obiect de tip parinte, urmand ca, in functie de alegerea programatorului, aceasta sa creeze si sa intoarca obiecte noi de tip copil.

**5.2 UML Class Diagram**



6. Data Model

Datele sunt pastrate in 3 fisiere xml :book.xml , user.xml si selling.xml. Fisierul book.xml contine pentru fiecare carte un id , titlul ,genul , autorul ,cantitatea si pretul .Fisierul user.xml contine pentru fiecare utilizator un id ,username-ul ,parola si tipul(administrator/regular user).Ultimul fisier , selling.xml contine informatii despre cartile care trebuie vandute.

7. System Testing

Pentru o buna functionalitate a aplicatii am implementat cateva teste care verifica ca datele de intrare sa fie valide si astfel sa nu putem introduce date gresite . Pentru inceput am creat teste la momentul logarii ,astfel ca doar introducand date care exista in fisierul user.xml se va putea face autentificarea.

8. Bibliography

* <https://stackoverflow.com/>
* <https://examples.javacodegeeks.com/core-java/writeread-csv-files-in-java-example/>
* <https://docs.oracle.com/javase/tutorial/>
* <http://docs.oracle.com/javase/tutorial/uiswing/>
* <http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>
* <https://www.tutorialspoint.com/>