Book Store

Student: Baidoc Vlad

**Group: 30641**

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

Folositi Swing/C# API pentru a implementa o aplicatie pentru angajatii unui magazin de carti. Aplicatia va trebui sa aiba 2 tipuri de utilizatori(un angajat si un administrator) cate trebuie sa aiba si sa introduca un nume de utilizator si parola pentru a folosi aplicatia.

Angajatul poate sa efectueze urmatoarele operatii:

* Sa caute carti dupa titlu, autor si an
* Sa vanda carti(orice titlu si orice cantitate

Administratorul poate sa efectueaza urmatoarele operatii:

- CRUD pe carti

- CRUD pe utilizatori

- Sa genereze rapoarte in format TXT sau XML cu cartile care lipsesc din stoc

# Functional Requirements

Utilizatorul: Sa caute carti dupa titlu, autor si an

Sa vanda carti(orice titlu si orice cantitate

Administratorul: CRUD pe carti

CRUD pe utilizatori

Sa genereze rapoarte in format TXT sau XML cu cartile care lipsesc din stoc

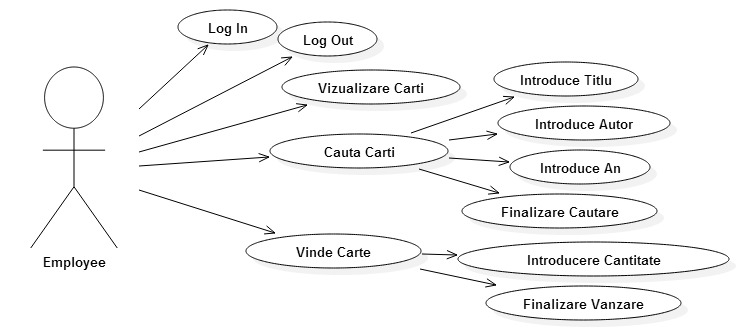
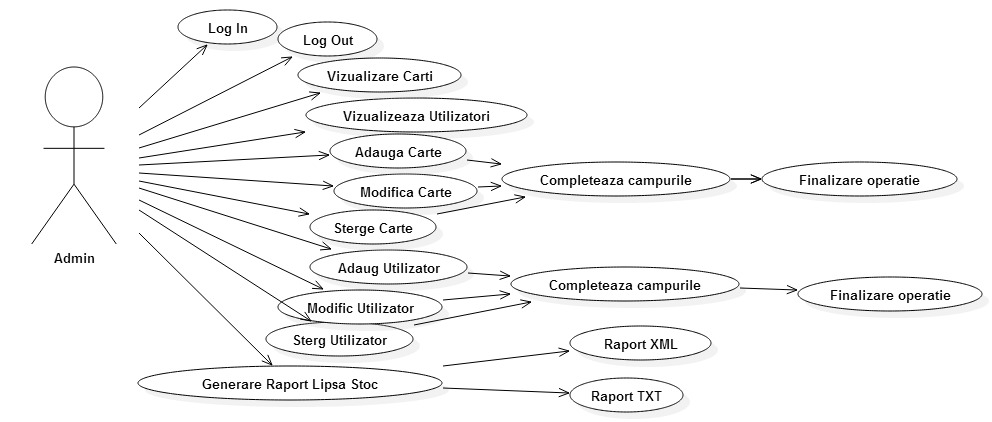
# Non-functional Requirements

O interfata grafica usor de folosit care permite folosirea tuturor functionalitatilor

Sa detecteze toate erorile utilizatorilor cu privire la introducerea datelor in formate corespunzatoare.

Aplicatia sa ruleze pe orice sistem care suporta libraria JAVA

2. Use-Case Model



Use-Case: Vanzare carte

Nivel: User Goal

Primary Actor: Employee

Main Success scenario: Angajatul se logheaza folosindu-si numele de utolizator si parola. Alege din tabel cartea pe care doreste sa o vanda si introduce cantitatea pe care o vinde(in limita stocului). Apasa butonul de vanzare.

3. System Architectural Design

**3.1 Architectural Pattern Description**

***Model-view-controller****(****MVC****)* este un model arhitectural utilizat în ingineria software. Succesul modelului se datorează izolării logicii de business față de considerentele interfeței cu utilizatorul, rezultând o aplicație unde aspectul vizual sau/și nivelele inferioare ale regulilor de business sunt mai ușor de modificat, fără a afecta alte nivele.

Model

Această parte a controlatorului manipulează operațiunile logice și de utilizare de informație (trimisă dinainte de către rangul său superior) pentru a rezulta de o formă ușor de înțeles.

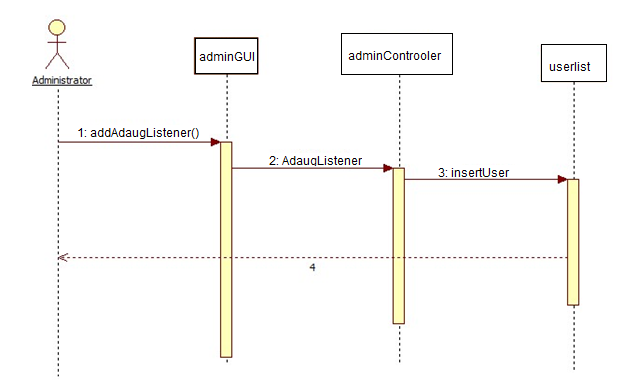
View

Acestui membru al familiei îi corespunde reprezentarea grafică, sau mai bine zis, exprimarea ultimei forme a datelor: interfața grafică ce interacționează cu utilizatorul final. Rolul său este de a evidenția informația obținută până ce ea ajunge la controlator.

Controller

Cu acest element putem controla accesul la aplicația noastră. Pot fi fișiere, scripts sau programe, in general orice tip de informație permisă de interfață. În acest fel putem diversifica conținutul nostru de o formă dinamică și statică, în același timp.

4. UML Sequence Diagrams



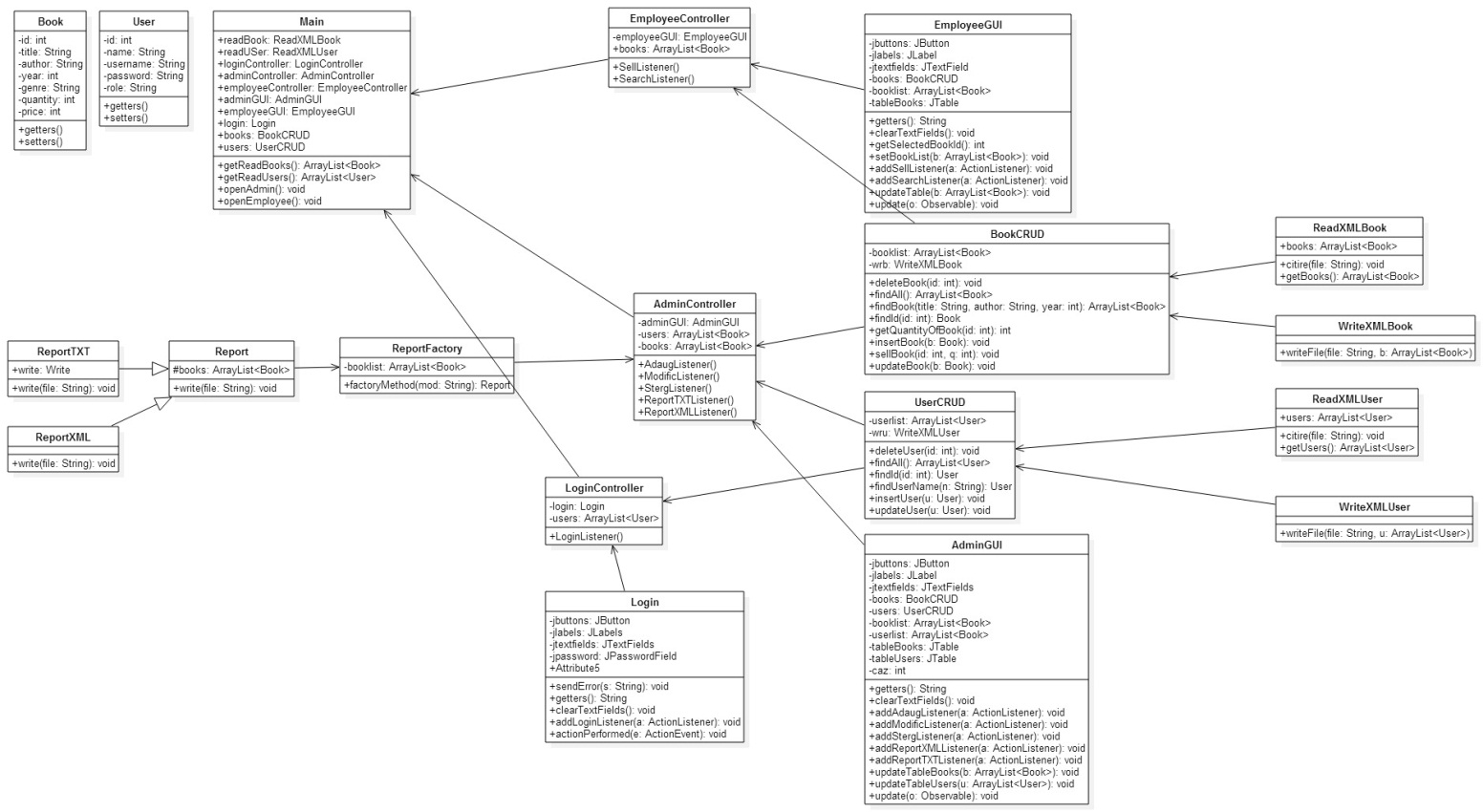
5. Class Design

**5.1 Design Patterns Description**

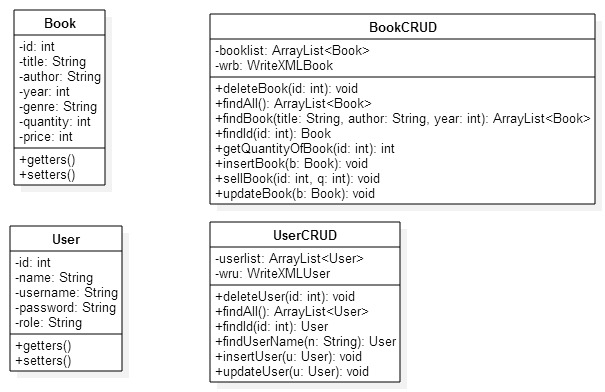
**Pentru generarea rapoartelor am folosit “Factory Method”. Administratorul poate alege formatul fisierului in care va fi generat raportul cu cartile care nu mai exista in stoc.**

**In programarea obiectuala, aceasta metoda este folosita pentru a indeplini acelasi scop dar prin metode diferite, obtinand obiecte diferite dar asemanatoare. Se creaza o clasa parinte si clasele care o mostenesc si pentru metode se foloseste metoda de “Overriding”.**

**5.2 UML Class Diagram**



6. Data Model



7. System Testing

Testarea a fost facuta folosind instructiunea System.out.println() pentru a afisa date sau pentru a notifica in ce metode s-a intrat, ce clase au fost instantiate, si prin testarea manuala a aplicatiei.