Magazin de carti

Document de analiza si design

Student:Suciu Vlad

**Groupa :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 4

4. UML Sequence Diagrams 5

5. Class Design 6

6. Data Model 7

7. System Testing 7

8. Bibliography 8

1. Requirements Analysis

# Assignment Specification

Acest assigment a dorit sa ne aduca in prim plan o aplicatie uzitata la un book store, care trebuie implementata in orice limbaj vrem noi.

# Functional Requirements

*Adiminstratorul poate:*

*-CRUD pe angajatii magazinului.*

*-sa genereze rapoarte cu privire la un anumit titlu de carte referitor la stoc.*

*Angajatul magazinului poate:*

*-CRUD pe Clientii bancii.*

*-CRUD pe conutrile clientilor la banca X.*

*-sa faca operatiuni de transfer*

*- sa efectueze plata utilitatilor.*

# Non-functional Requirements

*Cerinte nonfunctionale exista ca peste tot , mai ales in sectorul bancar , aici aduc vorba de securitate in primul rand unde am folosit un sistem de logare//*

*Apoi incapsularea datelor , era absolut necesara , unele date precum cnp ul raman private angajatului bancii etc*

2. Use-Case Model

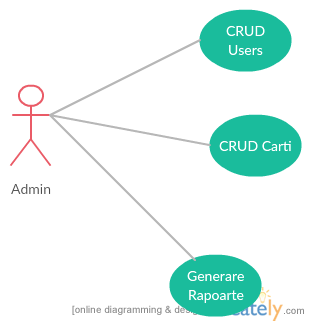
*Use case: verificare admin cantitati /exemplare dintr-o titulatura*

*Level: user goal level*

*Primary actor: adminul magazinului(logat ca admin)*

*Main success scenario:exemplarei suficiente*

*Extensions: numar de exemplare insuficient, sau logare invalida.*



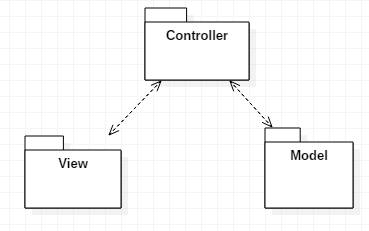
3. System Architectural Design

**3.1 Architectural Pattern Description**

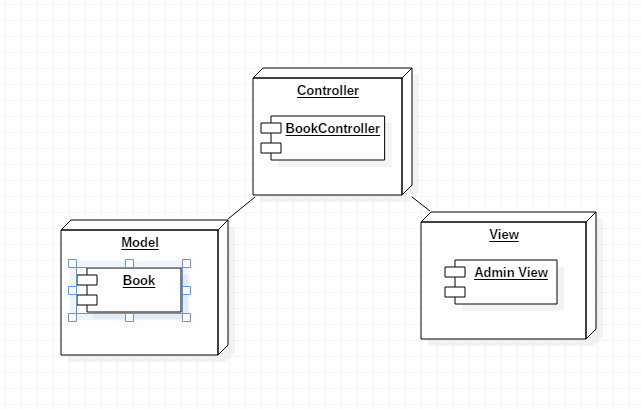
**Am folosit MVC care separa structura aplicatiei in model , vizualizare si controler/bussiness .Modelu se ocupa de comportamentul si datele aplicatiei. View-ul adduce UI ul necesar aplicatiei. El ilustreaza datele de la model intr-o reprezentare concreta. Controllerul primeste inputul de la user si face apel la obiece model si view pentru a face operatiile necesare.**

**3.2 Diagrams**

Package Diagram



Component Diagram

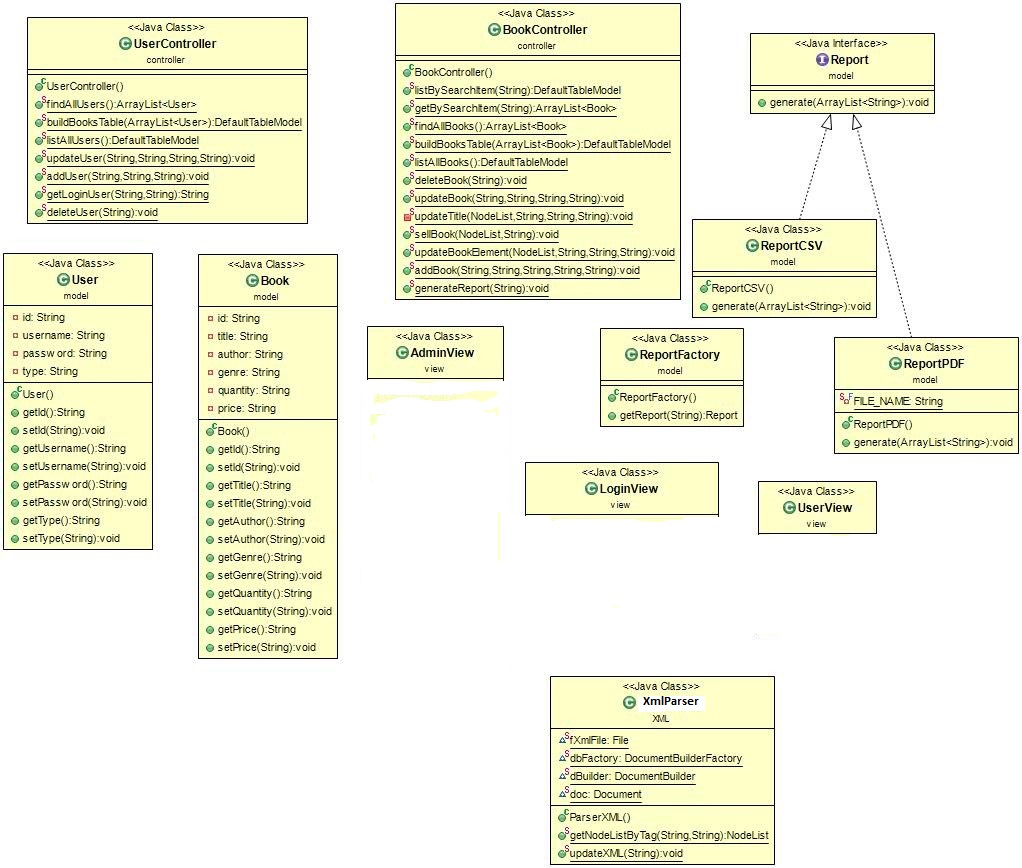


5. Class Design

**5.1 Design Patterns Description**

*Am folosit probabil Factory Method Pattern . Am interfata report si de acolo subclasele sunt responsabile pentru crearea intantei corecte CSV sau PDF.*

**5.2 UML Class Diagram**

**

6. Data Model .

*2 xml-uri unul pentru a tine cartile disponibile altul pentru useri. Ex:*

*<book id="1">*

*<title>Ion</title>*

*<author>Liviu Rebreanu</author>*

*<genre>realism</genre>*

*<quantity>0</quantity>*

*<price>30</price>*

*</book>*

*In baza mea de date am si xml pentru useri:*

<user id="1">

<username>Vlad</username>

<password>123</password>

<type>admin</type>

</user>

7.System Testing.

Testarea am facut-o atat din cod la implementare cat si din interfata si dupa, nerealizand un sistem de testare cu Junit test. In interfata am afisat mesaje de validare a datelor introduse precum si ascunderea parolei la introducere pentru a nu fi vizibila celor din jur.

8. Bibliography

1.<https://stackoverflow.com/>

2.Martin Fowler et. al, Patterns of Enterprise Application Architecture, Addison Wesley, 2003