<Ade`s Library>

Analysis and Design Document

Student: Tuluc Adelina

**Group: 30643**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Use Case Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

Proiectul implementat reprezintă o aplicaţie folosită de angajaţii unei librării. Aplicaţia are două tipuri de utilizatori. Aceştia pot fi utilizatori normali, adică angajaţi ai librăriei sau administrator. Pentru ca aceştia să utilizeze aplicaţia trebuie să introducă numele şi parola corespunzătoare.

Un angajat poate efectua următoarele operaţii:

* Căutarea cărţilor ȋn funcţie de gen, titlu sau autor;
* Vânzarea cărţilor.

Administratorul poate efectua următoarele operaţii:

* CRUD asupra informaţiilor despre cărţi (titlu, autor, gen, cantitate, pret);
* CRUD asupra informaţiilor despre angajaţi;
* Generarea unui raport despre cărţile care nu mai sunt ȋn stoc.

# Elaboration – Iteration 1.1

# Domain Model

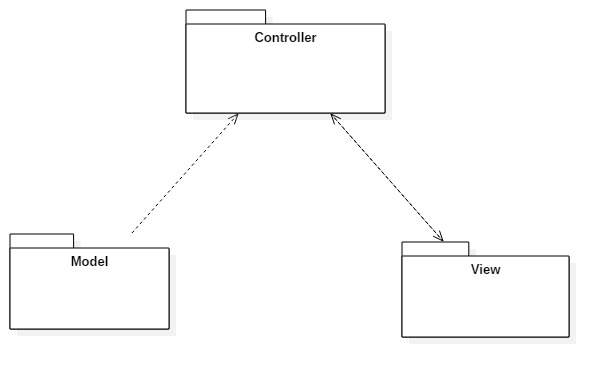
Proiectul este stucturat, folosind pattern-ul arhitectural MVC ( *Model View Controller*). Astfel, am creat 3 pachete, şi anume: Model, View şi Controller. Pachetul Model, care reprezintă modelul structurării datelor, conţine clasele *Book* şi *User*. Pachetul View, care se ocupă de interfaţa cu utilizatorul, conţine clasele: *LoginFrame, AdminFrame, UserFrame, Error, FoundMessageBox* şi *NotFoundMessageBox*. Pachetul Controller, care se ocupă cu funcţionalitatea aplicaţiei, conţine o singură clasă, şi anume: *LibraryController*.

# Architectural Design

## Conceptual Architecture

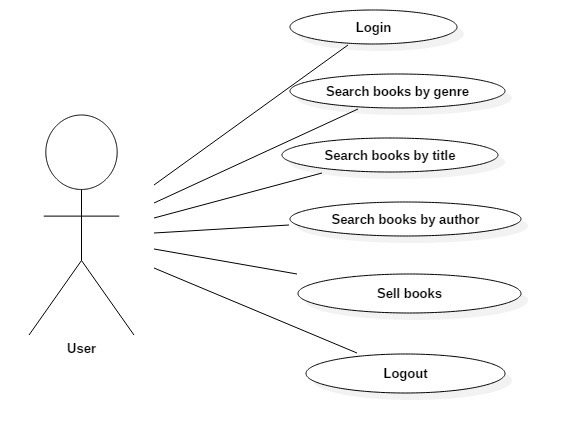
Am folosit pattern-ul Factory Method pentru generarea rapoartelor.

## Package Design



## Use Case Diagrams

# 

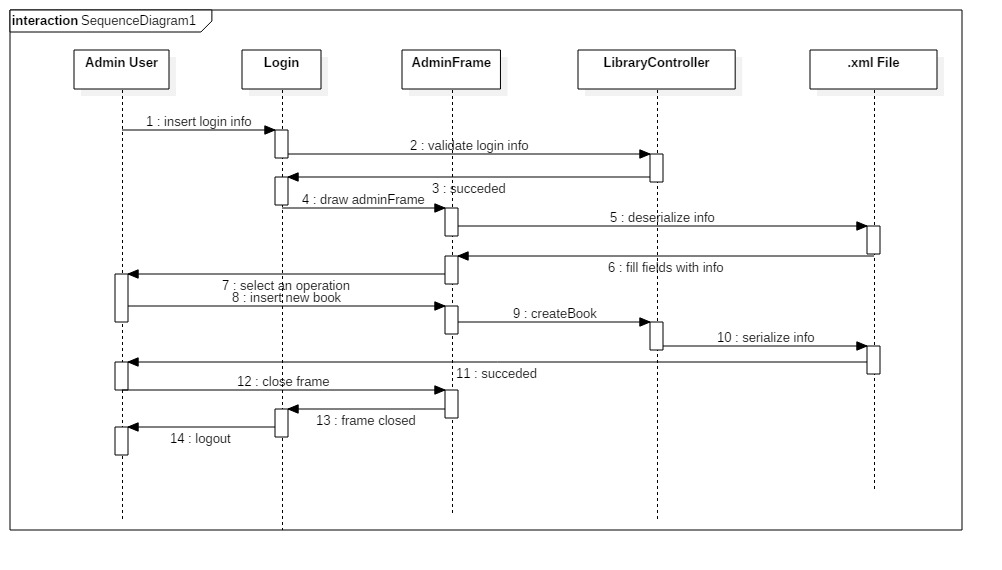


# Elaboration – Iteration 1.2

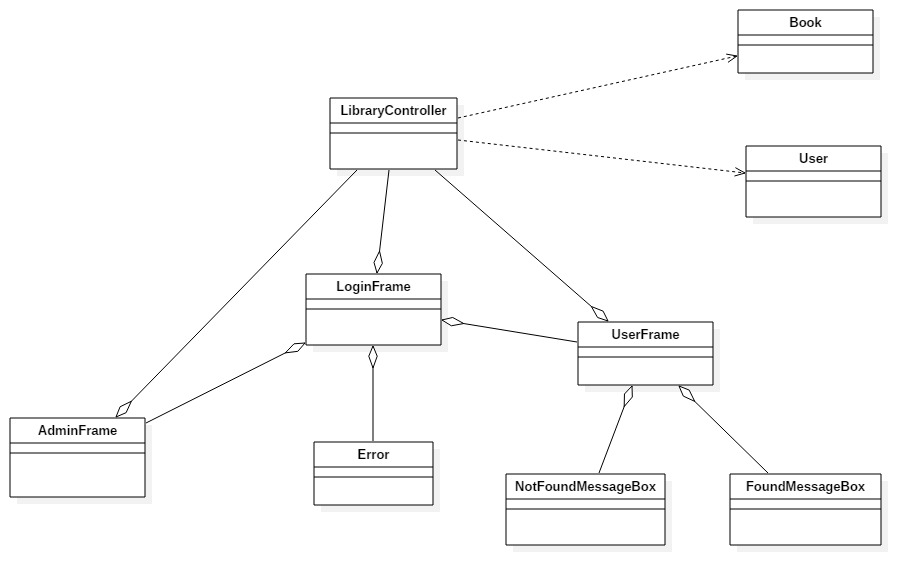
# Design Model

## Dynamic Behavior

Ȋn figura de mai jos, este o diagramă de secvenţă. Aceasta reprezintă operaţiile necesare introducerii unui nou client ȋn “baza de date”. Această operaţie este efectuată de către administrator. După ce acesta introduce datele de intrare (username şi password), aceste date vor fi validate ȋn cadrul clasei LibraryController, iar dacă sunt valide se va desena fereastra AdminFrame. Ȋn momentul deschiderii acestei ferestre, se vor deserializa informaţiile introduse ȋn sesiunile anterioare. Ȋn cadrul ferestrei, administratorul poate alege o mulţime de operaţii pe care le poate efectua asupra informaţiilor despre cărţi sau despre utilizatori. Presupunem că doreşte să adauge o carte nouă. Noua carte introdusă se va salva ȋn acelaşi fişier .xml, serializând obiectul actual. După efectuarea operaţiei dorite, administratorul se deloghează.



## Class Design



# Data Model

Pentru salvarea datelor, am utilizat un fişier de tip *.xml,* ȋn care am salvat informaţiile necesare cu ajutorul serializării. Ȋn clasa LibraryController, am implementat două metode care realizează acest lucru, şi anume: *serializeObj()* şi *deserializeObj().* La pornirea aplicaţiei, datele salvate anterior se vor “deserializa”, adică se vor ȋncărca pentru a fi folosite. La ȋnchiderea aplicaţiei, noile date se salvează ȋn acelaşi fişier.

# Unit Testing

*[Present the used testing methods and the associated test case scenarios.]*

# Elaboration – Iteration 2

# Architectural Design Refinement

Proiectul este stucturat, folosind pattern-ul arhitectural MVC ( *Model View Controller*). Astfel, am creat 3 pachete, şi anume: Model, View şi Controller. Pachetul Model, care reprezintă modelul structurării datelor, conţine clasele *Book* şi *User*. Pachetul View, care se ocupă de interfaţa cu utilizatorul, conţine clasele: *LoginFrame, AdminFrame, UserFrame, Error, FoundMessageBox* şi *NotFoundMessageBox*. Pachetul Controller, care se ocupă cu funcţionalitatea aplicaţiei, conţine o singură clasă, şi anume: *LibraryController*.

# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

# Construction and Transition

# System Testing

*[Describe how you applied integration testing and present the associated test case scenarios.]*

# Future improvements

Această aplicaţie paote fi ȋmbunătăţită pe mai multe planuri. Una dintre ȋmbunătăţirile posibile este de a dezvolta mai mult acţiunile permise utilizatorilor normali, adică angajaţilor.

# Bibliography