<Assignment 2>

Analysis and Design Document

Student:Ionel Carmen

**Group:30237**

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

Implementati o aplicatie web pentru angajatii unui magazine de carti. Aplicatia trebuie sa aiba doua tipuri de utilizatori: angajatii magazinului si administratorul. Toti utilizatorii trebuie sa introduca un username si o parola pentru a utiliza aplicatia.

# Functional Requirements

Pentru aceasta aplicatie, datele vor fi stocate in baze de date in format xml.

Se va utiliza design pattern-ul Factory Method pentru a se genera doua tipuri de rapoarte (.pdf si .csv) si pattern-ul arhitecturat Model View Controller.

Utilizatorul reprezentat de catre administrator trebuie sa aiba urmatoarele operatii: CRUD asupra cartilor si a angajatilor si generare de raport.

Utilizatorul reprezentat de catre angajatul magazinului trebuie sa aiba urmatoarele opratii: CRUD asupra comenzilor si cautarea unei carti dupa titlu, autor si gen.

# Non-functional Requirements

# Availability: Reprezinta probabilitatea ca un sistem sa functioneze corect chiar si dupa folosirea lui un anumit timp. Atat datele cat si pagina web vor fi disponibile atat timp cat autorul sau administratorul nu le sterge.

## Performanta: Una dintre cele mai importante cerinte non-functionale dorite de catre client este un sistem cu o rata mare de performanta. Obiectivul de performanta este divizat in trei: timp, spatiu si receptivitate. Timpul este impartit in timp de raspuns si timp de procesare. Receptivitatea reprezinta cat de repede raspunde sistemul la interogari, iar spatiul reprezinta spatiul de stocare. Pentru o buna performanta, nu trebuie irosit spatiul.

## Security: Obiectivul de performanta este divizat in: confidentialitate, integritate si disponibilitate. Pentru a acoperi confidentialitatea este importanta autentificarea: utilizatorii trebuie sa se conecteze la sistem prin furnizarea unui username si a unei parole. Integritatea se refera la validarea datelor si asigura consistenta.

## Testability: Reprezinta gradul in care un sistem suporta testarea intr-un anumit context. Pentru a testa un sistem trebuie sa avem in vedere:

-proprietatile metodelor de test folosite

-cerintele trebuie sa fie consistente, complete, neambigue, realizabile

-componentele software trebuie sa fie: controlabile, observabile, heterogene, documentate.

## Usability: Orice sistem creat pentru a fi utilizat de catre oameni trebuie sa fie usor de folosit, usor de invatat, usor de amintit (instructiunile) si folositor utilizatorului.

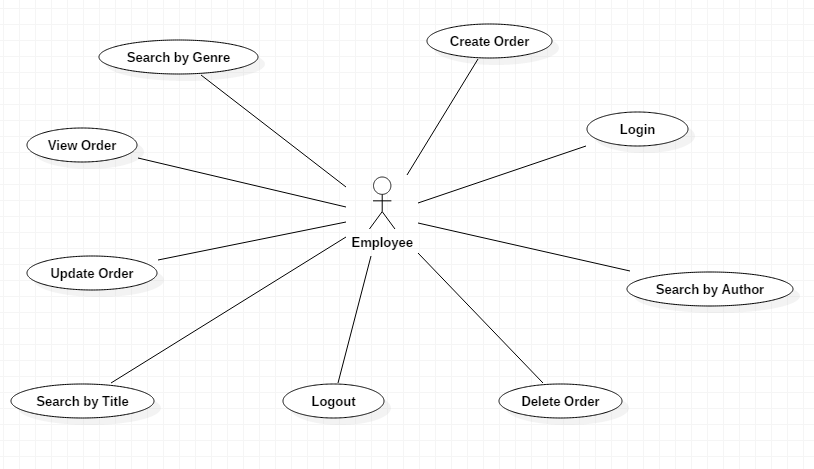
Principii:

-concentrat pe utilizatorii finali si pe nevoile acestora

-design iterative, acesta fiind imbunatatit in permanenta

Pentru asta. Aplicatia dispune de o interfata web, usor de utilizat si interactiva.

2. Use-Case Model

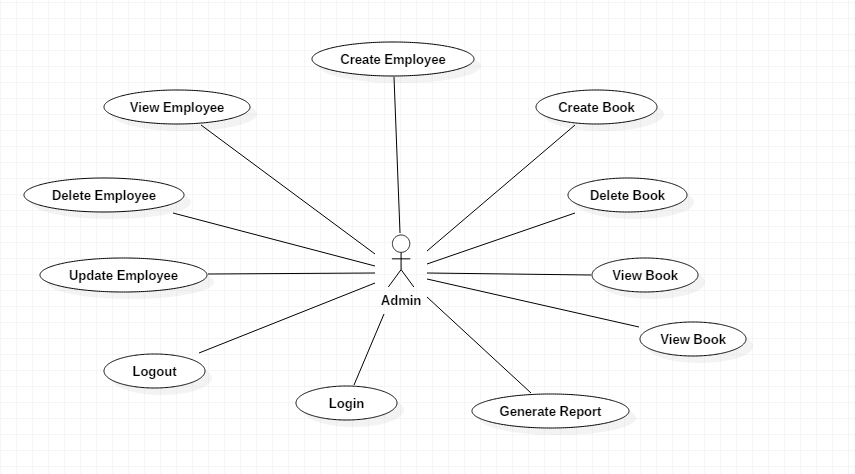
**

*Use case: <Create Order>*

*Primary actor: <Employee>*

*Main success scenario: <Angajatul introduce cartile pe care le doreste la comanda, introduce un nume afferent comentii si apasa butonul Add si comanda se salveaza in fisierul xml>*

*Extensions: <Daca cartile pe care le introduce, nu exista in stoc se va afisa eroare>*



*Use case: <Add Employee>*

*Primary actor: <Admin>*

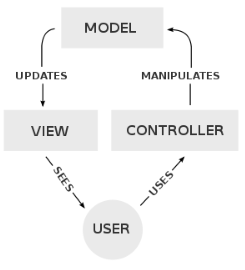
*Main success scenario: <Administratorul introduce datele unui angajat, apasa pe butonul Add si acesta se salveaza in fisierul xml>*

*Extensions: <Daca angajatul exista deja sau datele nu sunt valide se va afisa eroare>*

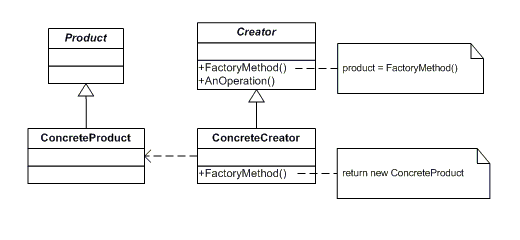
3. System Architectural Design

**3.1 Architectural Pattern Description**

MVC:

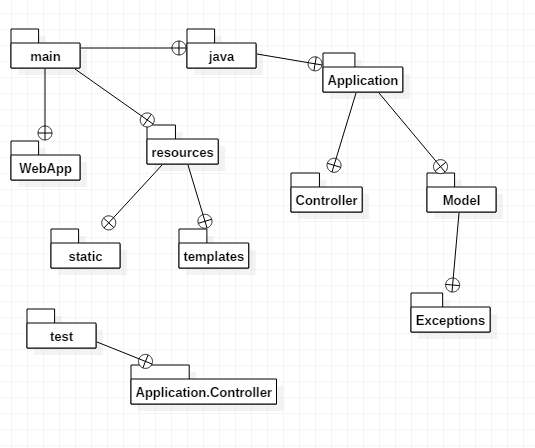


*Factory Method:*

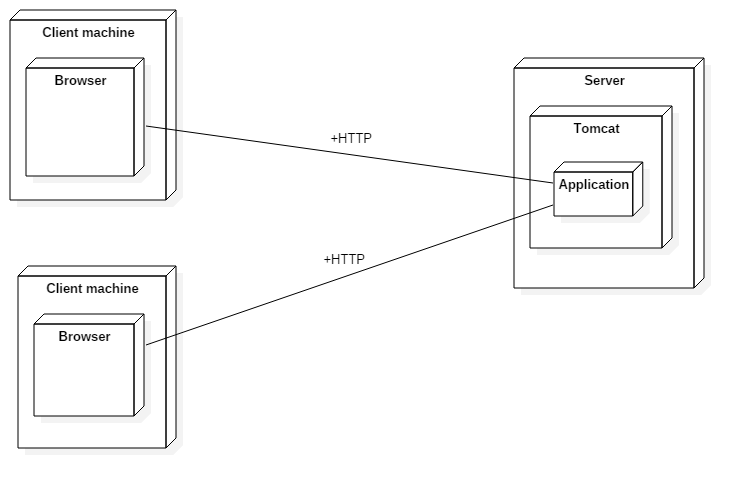
**

**3.2 Diagrams**

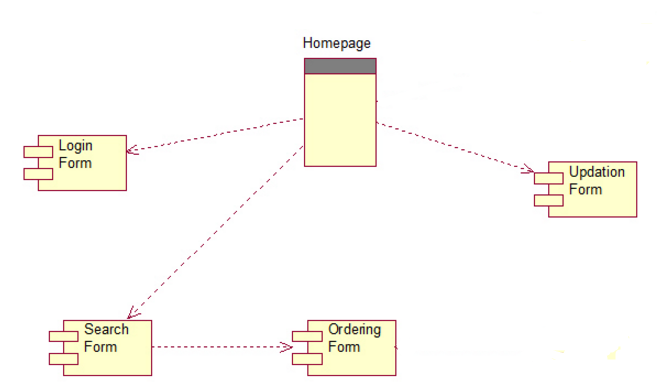
*Package diagram:*

**

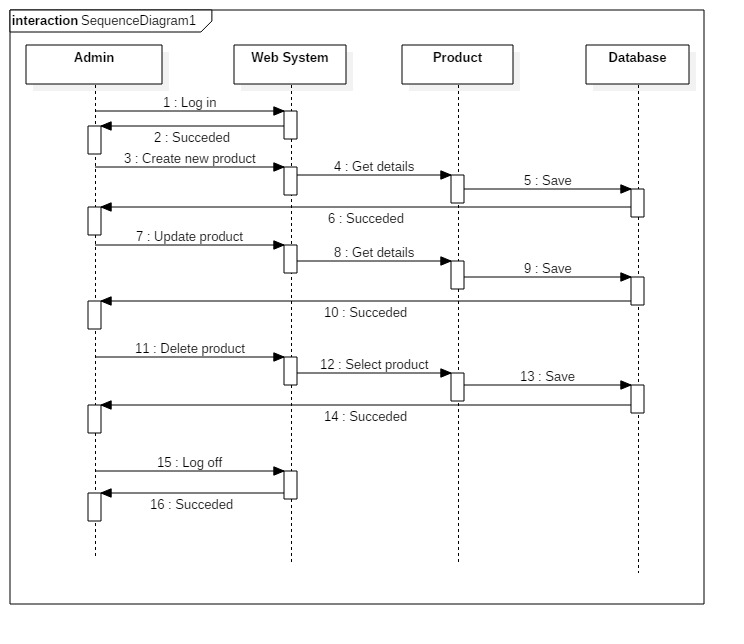
*Deployment diagram:*



*Component diagram:*

**

4. UML Sequence Diagrams

**

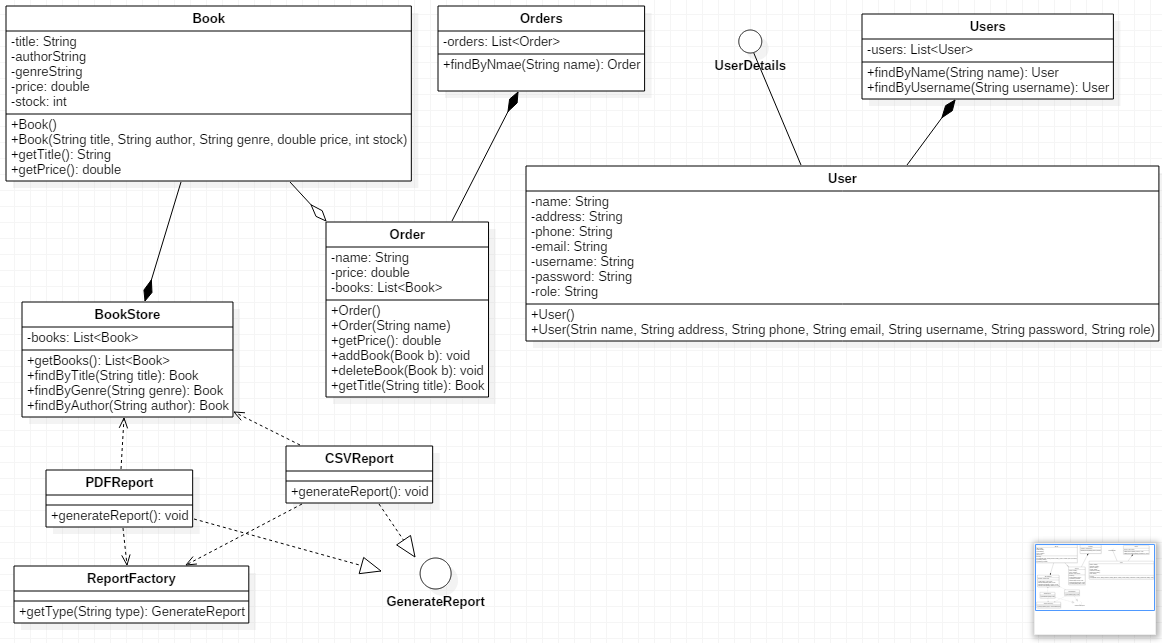
5. Class Design

**5.1 Design Patterns Description**

*Design patternul MVC: este utilizat pentru implementarea interfetelor utilizator. Divide o aplicatie in trei parti interconectate cu scopul de a separa reprezentarea datelor interne de modul in care acestea sunt prezentate utilizatorului.*

*Pattern-ul creationl Factory Mathod se utilizeaza cu scopul de a crea obiecte fara a specifica clasa exacta a obiectului care va fi creat. Acest lucru se realizeaza prin apelarea unei metode (specificata intr-o interfata) implementata de catre sub-clase.*

**5.2 UML Class Diagram**



6. Data Model

*Din specificatia aplicatiei reies clasele importante: User, Book, Order. Userul poate fi administratorul sau un angajat. Order , contine o lista de carti.*

*Pentru fiecare clasa, exista o un wrapper: BookStore, Oreders, Users reprezentand clasele care vor fi “scrise” in XML.*

7. System Testing

*Aplicatia a fost testat introducand date din interfata web si verificand baza de date in paralel.*

*Aplicatia a fost testate, de asemenea prin clase de test Junit, care verifica toata opratiile CRUD asupra bazelor de date xml.*

8. Bibliography

1. Curs M.Dansoreanu

2. Patterns of enterprise application architecture – Martin Fowler

3. Surse web

<http://docs.oracle.com/javase/tutorial/uiswing/>

<http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>

<https://msdn.microsoft.com/en-us/library/54xbah2z(v=vs.110).aspx>

<https://msdn.microsoft.com/en-us/library/e80y5yhx(v=vs.110).aspx>