Student: Victor-Petru Almasan

**Group: 30235**

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 5

4. UML Sequence Diagrams 6

5. Class Design 7

6. Data Model 8

7. System Testing 8

8. Bibliography 9

1. Requirements Analysis

# Assignment Specification

Aceasta aplicatie este destinata unei agentii de turism. Aplicatia ajuta agentia in gestionarea facila a numarului mare de turisti si de rezervari din aceste vremuri.

# Functional Requirements

Aplicatia are doua tipuri de useri: administrator si user normal. Administratorul poate sa vizualizeze/editeze/adauge/stearga agenti. De asemenea acesta poate sa genereze un fisier .pdf cu activitatea tuturor agentilor din agentia de turism. Userul normal poate sa adauge/editeze/vizualizeze clienti, vizualizeze/editeze/adauge rezervari pentru un client.

# Non-functional Requirements

* *Update automat la lista userilor in interfata*
  + Source of stimulus: Administrator
  + Stimulus: Interfata trebuie sa fie prietenoasa la adaugarea unui user
  + Environment: Logare ca administrator
  + Artifact: AdminLayoutController
  + Response: Lista tuturor userilor .este actualizata
* *Adaugarea unui payment pentru o rezervare*
  + Source of stimulus: User
  + Stimulus: Interfata ofera vizual ce fel de constrangeri trebuie aplicate
  + Environment: Logare ca user normal
  + Artifact: UserLayoutController
  + Response: Butonul si TextField-ul sunt disable/enable.

2. Use-Case Model

Use case: Operatii pe agenti

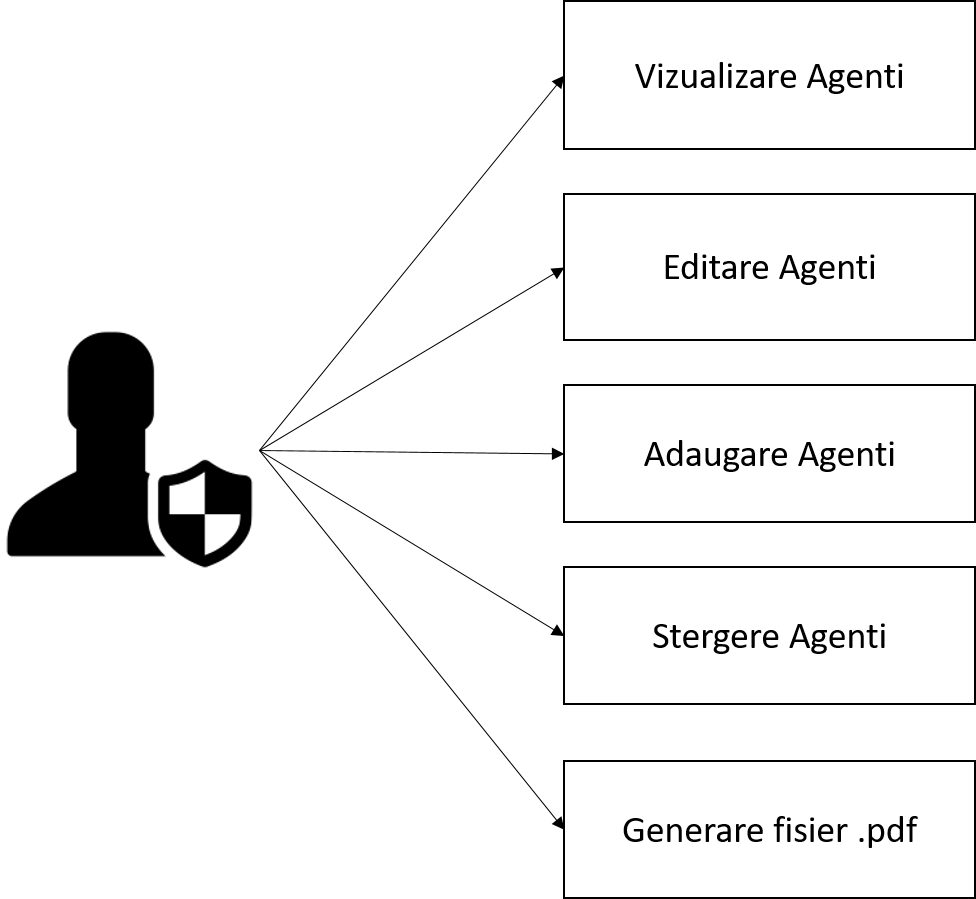
Level: summary level

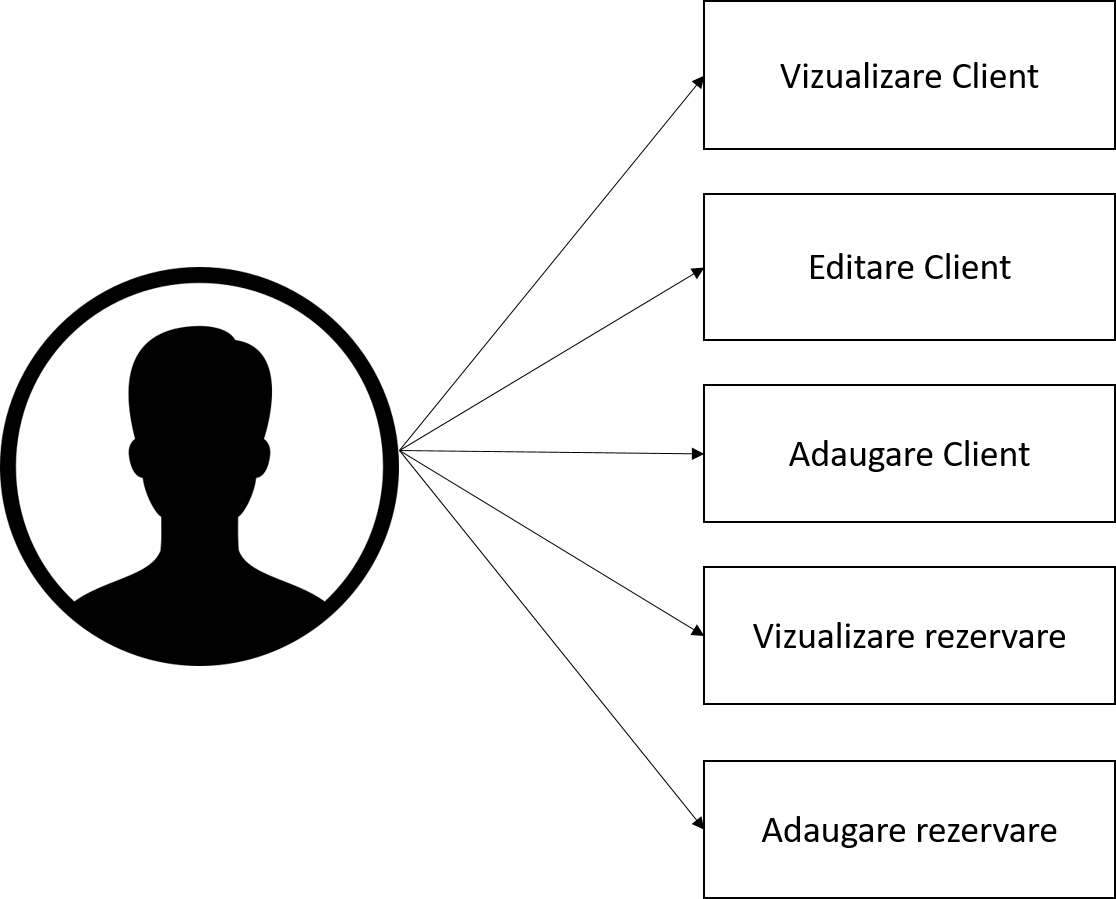
Primary actor: administrator

Use case: Operatii pe clienti si rezervari

Level: user-goal level

Primary actor: user





3. System Architectural Design

**3.1 Architectural Pattern Description**

In aceasta aplicatie s-au folosit Data mapper si Domain Model. A fost create o clasa Mapper pentru fiecare tabel cu care s-a interactionat direct in baza de date. Metodele din aceste clase prind niste exceptii specifice pentru fiecare tabel. S-a folosit Domain Model si astfel spre exemplu pentru fiecare client citit din baza de date se creaza un nou obiect continand clientul respective.

**3.2 Diagrams**

Diagrama de pachete:

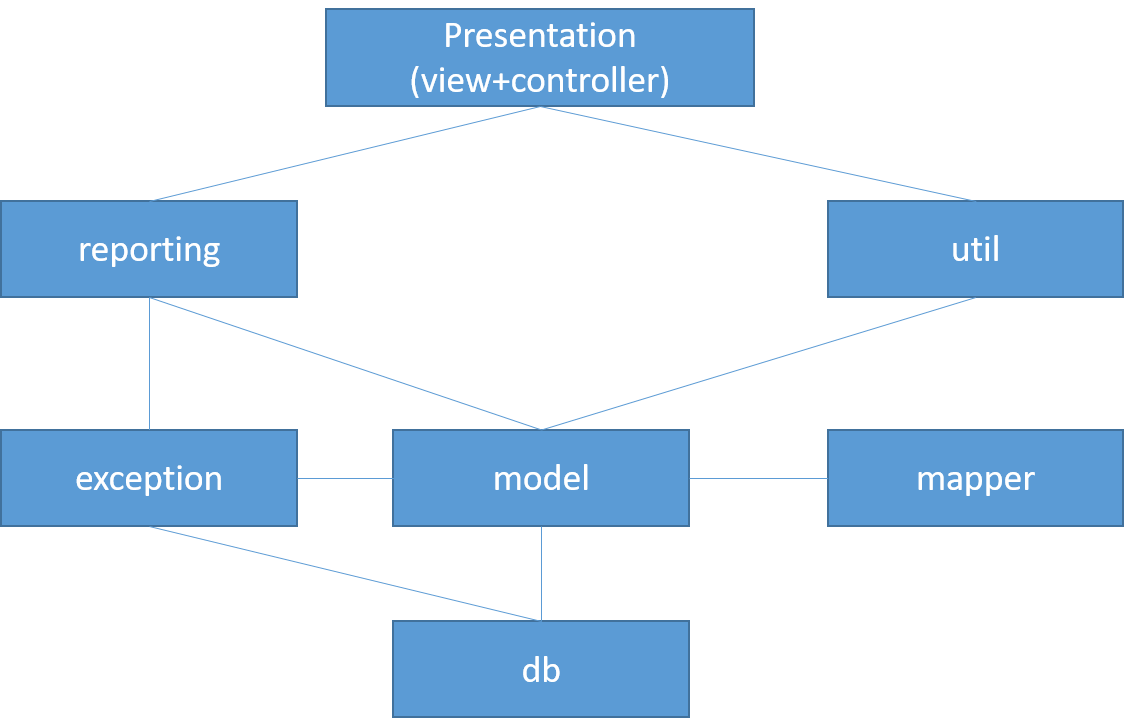
**

Diagrama de componente pentru un booking:

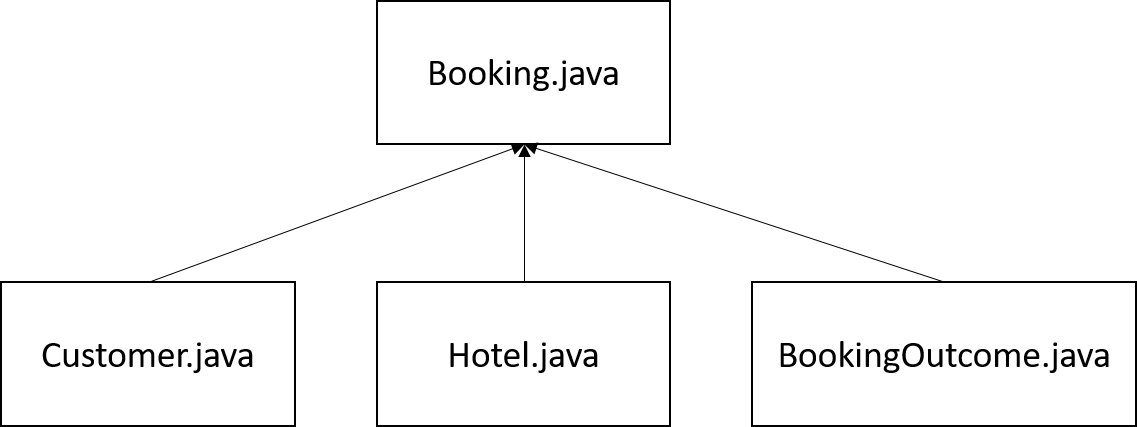
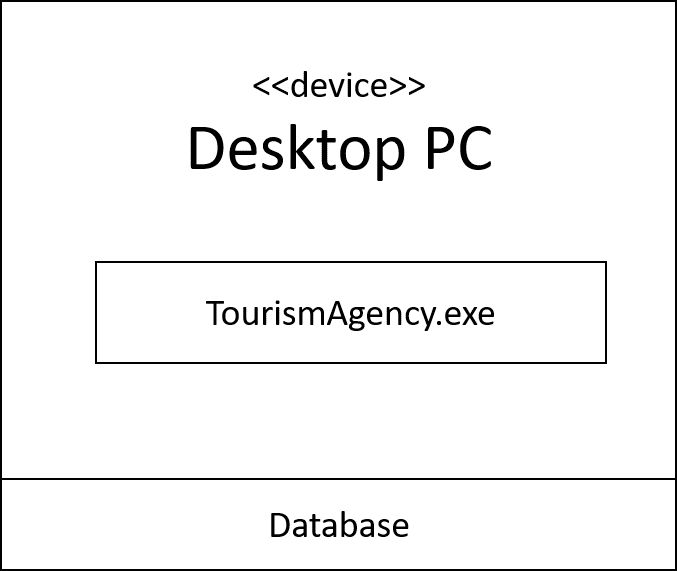
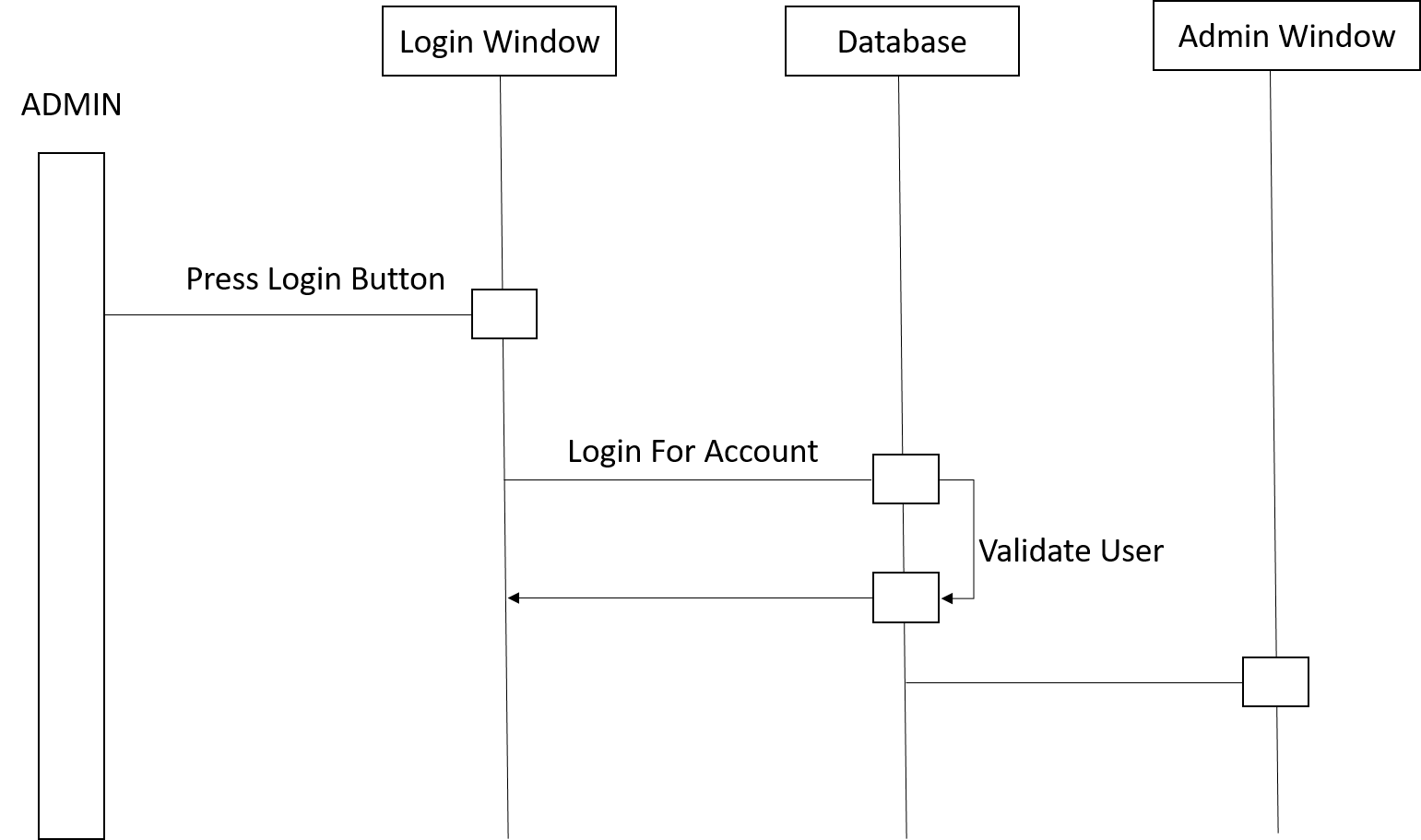
**

Diagrama Deployment:



4. UML Sequence Diagrams

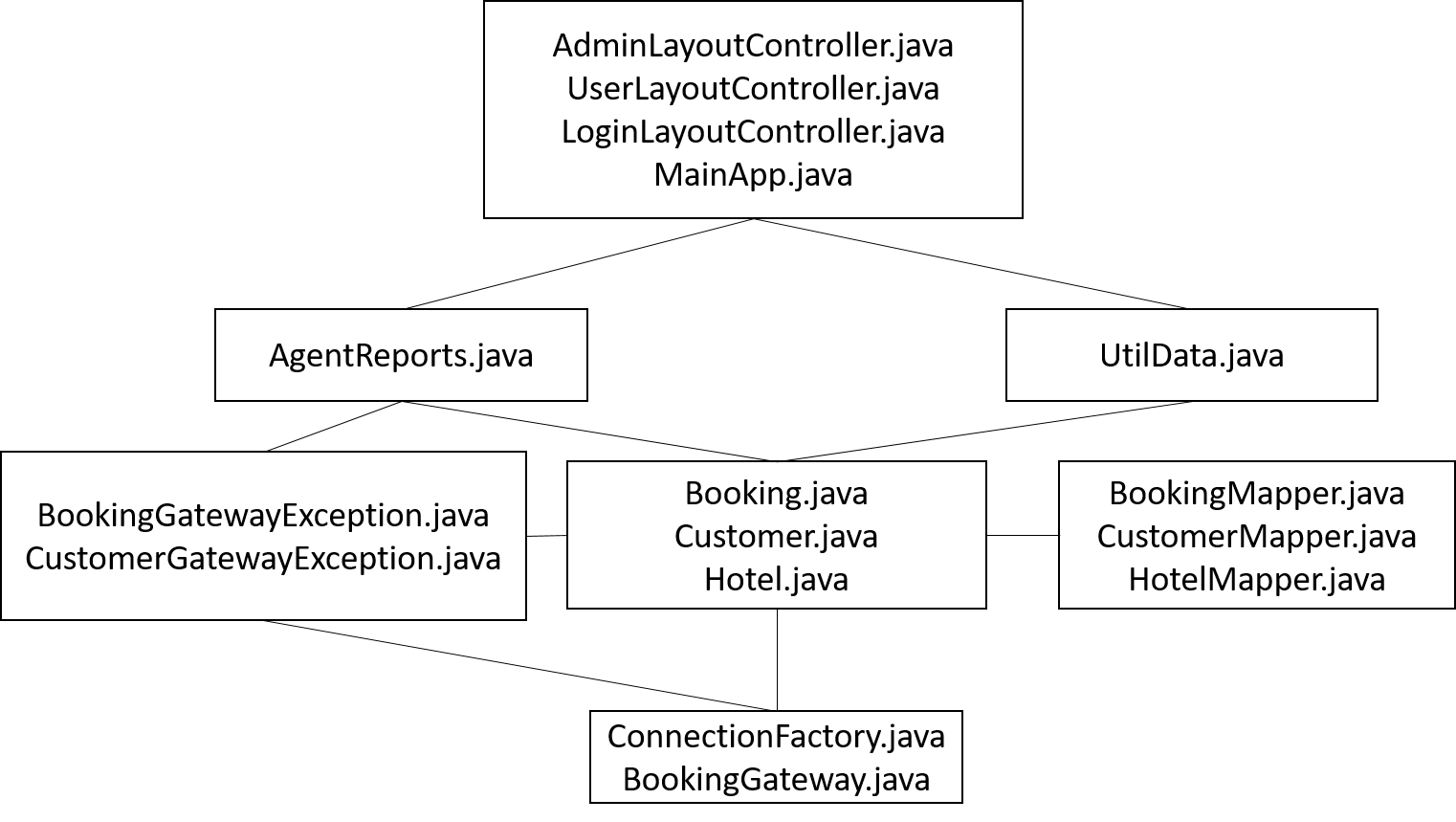
Diagrama de secvente pentru actiunea de logare cu un cont administrator:



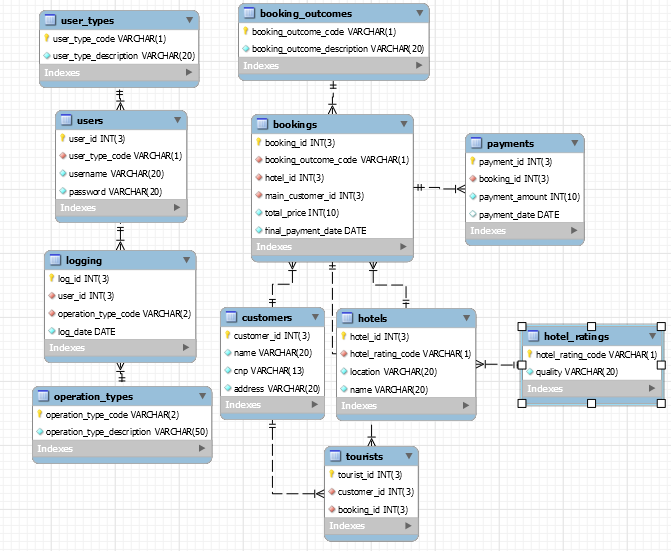
5. Class Design

**5.1 UML Class Diagram**

Deoarece in proiect sunt multe clase am adaugat dooar cateva dintre ele in diagrama de clase:

**

6. Data Model

**

7. System Testing

Aplicatia are numeroase validari. Aceste validari sunt vizibile chiar si din interfata utilizator prin punerea unor componente in statutul de disable atunci cand o anumita operatie nu este permisa.

8. Bibliography

<http://richard.jp.leguen.ca/tutoring/soen343-f2010/tutorials/implementing-data-mapper/>

<https://stackoverflow.com/questions/433819/table-module-vs-domain-model>

<http://code.makery.ch/library/javafx-8-tutorial>

<https://docs.oracle.com/javase/8/docs/api>