<Assignment3>

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

Utilizati JAVA API pentru a implementa o aplicatie client-server pentru managementul consultatiilor unei clinici. Aplicatia trebuie sa aiba trei tipuri de utilizatori: secretara clinicii, doctorii si administratorul.

Secretara trebuie sa aiba la dipozitie urmatoarele operatii: adaugarea si actualizarea pacientilor, adaugare, stergere, vizualizare si actualizarea consultatiilor si o metoda de a anunta doctoral cand pacientul a ajuns pentru consultative.

Doctorul trebuie sa aiba urmatoarele operatii disponibile: sa adauge si sa vizualizeze consultatiile unui pacient.

Administratorul trebuie sa aiba urmatoarele operatii: adaugare,stergere,actualizare,vizualizare useri.

# Functional Requirements

Pentru aceasta aplicatie, datele trebuie stocate intr-o baza de date.

Se va utiliza design pattern-ul Observer pentru a notifica un doctor in momentul in care pacientul sau a ajuns pentru consultative.

# 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: Add User*

*Level:* user-goal level

*Primary actor:* Admin

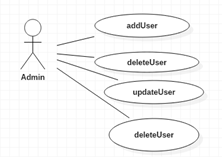
*Main success scenario:*

1. Login cu succes
2. Selectarea meniului de anim
3. Introduce date valide pentru un user
4. Userul nou este adaugat in maza de date

*Extensions:*

Cazuri de eroare:

1. Parola sau username gresit
2. Date invalide



*Use case: Add Patient*

*Level:* user-goal level

*Primary actor:* Secretary

*Main success scenario:*

1. Login cu secces

2. Selectarea meniului secretarei

3. Introducerea datelor valide a unui pacient

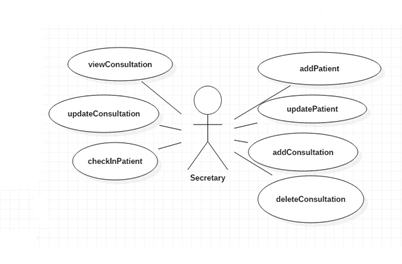
4. Pacientul este adaugat la baza de date

*Extensions:*

Cazuri de eroare:

1. Parola sau username gresite

2. Date invalide



*Use case: Add info*

*Level:* user-goal level

*Primary actor:* Doctor

*Main success scenario:*

1. Login cu succes

2. Selectarea meniului doctorului

3. Introduce numele pacientului, data si ora consultatiei

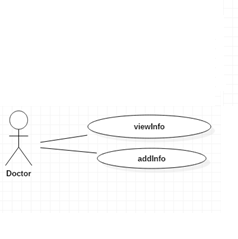
4. Adauga observatii la consultatii

*Extensions:*

Cazuri de eroare:

1. Parola sau username gresite

2. Nu exista consultatia

**

3. System Architectural Design

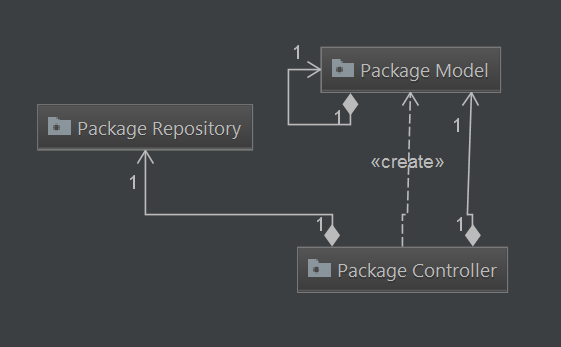
**3.1 Architectural Pattern Description**

*Design patternul Observer: este utilizat pentru implementarea sistemelor pentru managementul evenimentelor. Un obiect (numit subiect), mentine o lista a obiectelor dependente de el (numite observeri) si ii notifica automat la orice schimbare a starii, prin apelarea unei metode.*

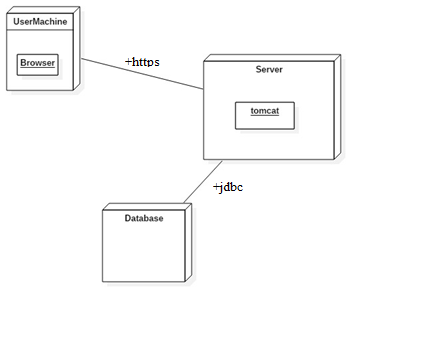
*Design patternul MVC: este utilizat pentru implementarea interfetelor utilizator. Divide o aplicatie in trei parti interconectate cu scopul de a separa reprezentarea datelor interne de modul in care acestea sunt prezentate utilizatorului.*

**3.2 Diagrams**

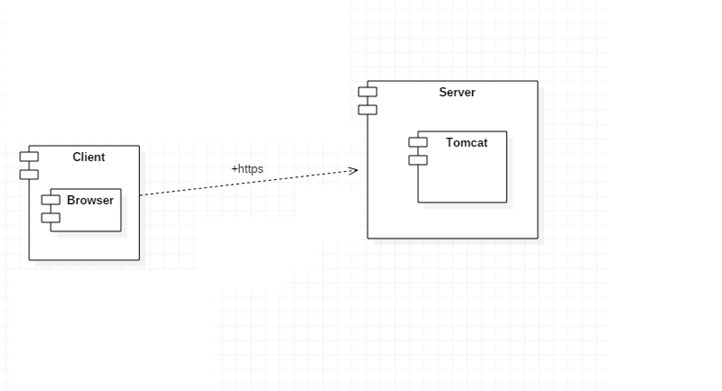
*Diagrama de pachete:*



*Deployment diagram:*

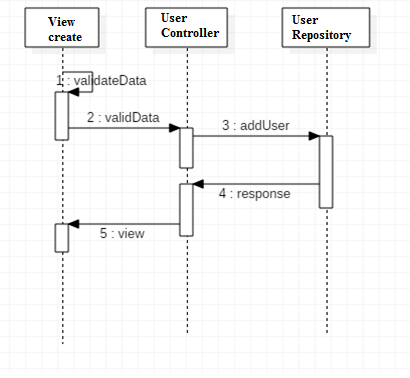
**

*Component diagram:*

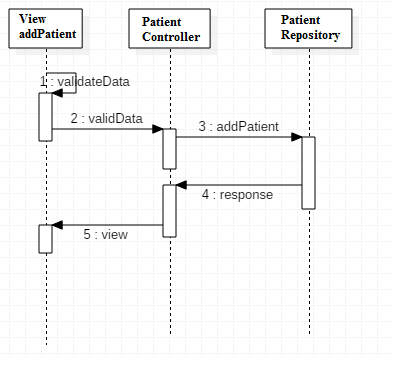
**

4. UML Sequence Diagrams

*addUser*



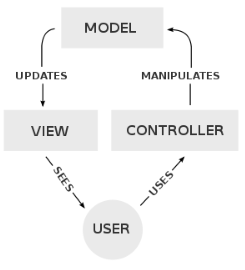
*addPatient*



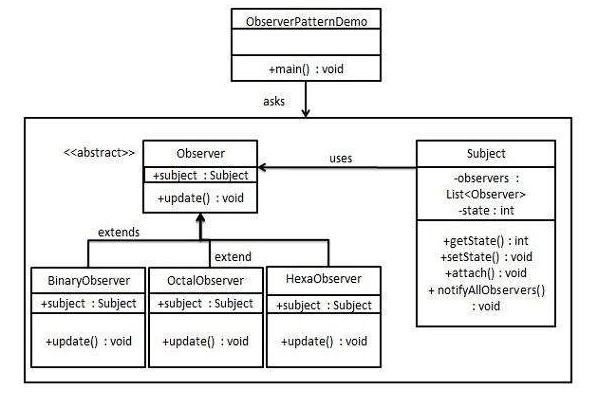
5. Class Design

**5.1 Design Patterns Description**

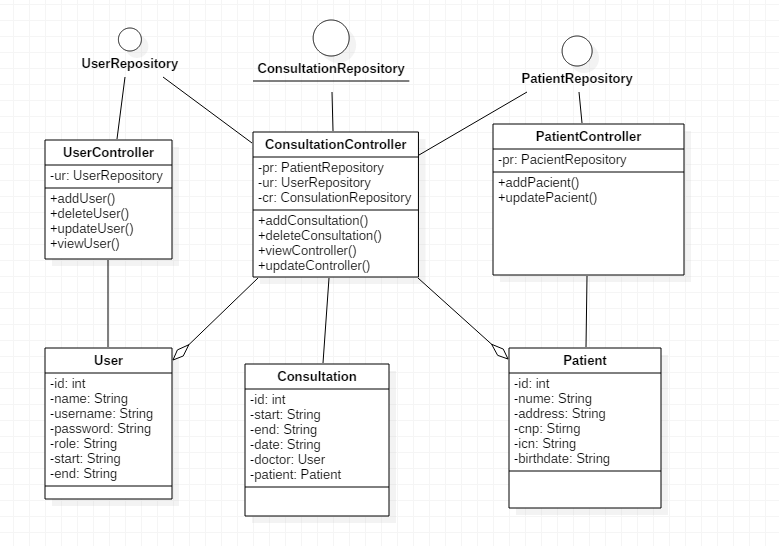
MVC:

**

Observer:



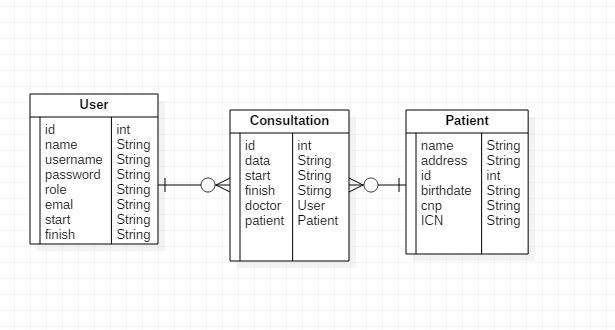
**5.2 UML Class Diagram**



1. Data Model

Datele aplicatiei reies din specificatia problemei.

Clasele principale ale aplicatiei sunt: User, Consultation si Patient. Tabela User contine toti utilizatorii: administratorul, secretara si doctorul. Un pacient poate avea mai multe consultatii, in timp ce o consultatie poate avea doar un pacient. Un doctor poate avea mai multe consultatii, dar o consultative poate avea doar un doctor.

7. System Testing

Aplicatia a fost testat introducand date din interfata web si verificand baza de date in paralel. Conexiunea cu baza de date a fost testat in prima faza a aplicatiei. In ceea ce priveste testarea metodelor prin unitTest, era necesara de o baza de date suplimentara, cea ce interfera cu baza de date actuala.

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>