Clinic

Student: Tomuș Alexandra

**Group:30234**

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. Class Design 3

5. Data Model 3

6. System Testing 3

7. Bibliography 3

1. Requirements Analysis

# Assignment Specification

Use the C# API to design and implement a client-server application for managing the consultations of doctors in a clinic. The application has three types of users: the clinic secretary, the doctors and an administrator.

# Functional Requirements

The clinic secretary can perform the following operations:

* Add/update patients (patient information: name, identity card number, personal numerical code, date of birth, address).
* CRUD on patients’ consultations (e.g. scheduling a consultation, assigning a doctor to a patient based on the doctor’s availability).

The doctors can perform the following operations:

* Add/view the details of a patient’s (past) consultation.

The administrator can perform the following operations:

* CRUD on user accounts.

In addition, when a patient having a consultation has arrived at the clinic and checked in at the secretary desk, the application should inform the associated doctor by displaying a message.

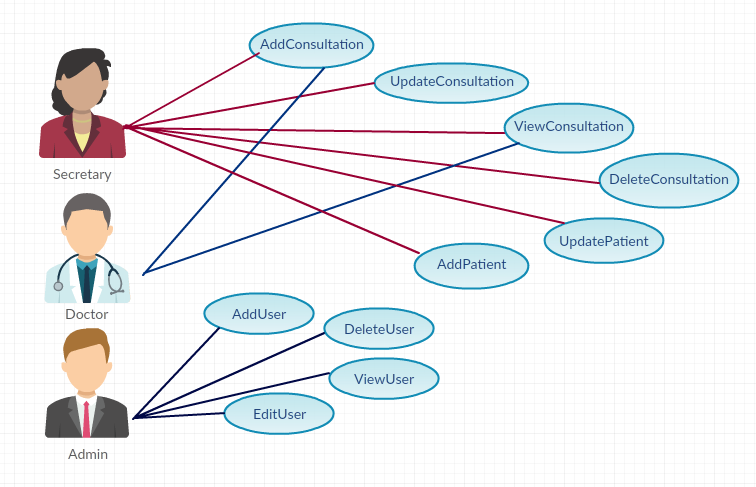
**Application Constraints**

There will be 3 applications, on for every type of user and including only the functionality for that type of user. The architecture should be client-server and the data will be stored in a database. The client applications will be Windows Forms applications. Use a .Net Web API to expose the server functionality to the client applications. Create a pooling mechanism in a separate Thread which will query the server at predefined short intervals to find out if a new patient has checked in.

2. Use-Case Model

Un model de utilizare constă dintr-un număr de elemente de model. Cele mai importante elemente de model sunt: cazuri de utilizare, actori și relațiile dintre ele.  
 Un actor este reprezentat de administrator, care face operatii CRUD pe informatia utilizatorilor.  
 Un alt actor este doctorul, care se poate sa faca operatiile de adaugare si vizualizare a consultatiilor.

Un alt actor este secretara care poate adauga / actualiza / sterge/ vizualiza informații despre consultatii si deasemnea poate adauga si actualiza datele despre pacienti.

**

3. System Architectural Design

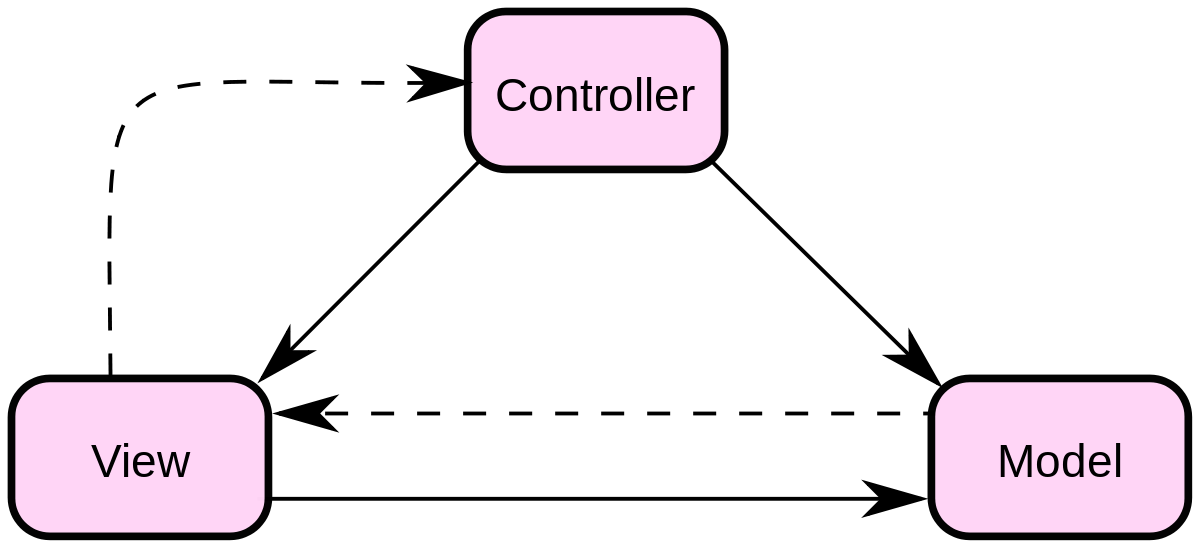
**3.1 Architectural Pattern Description**

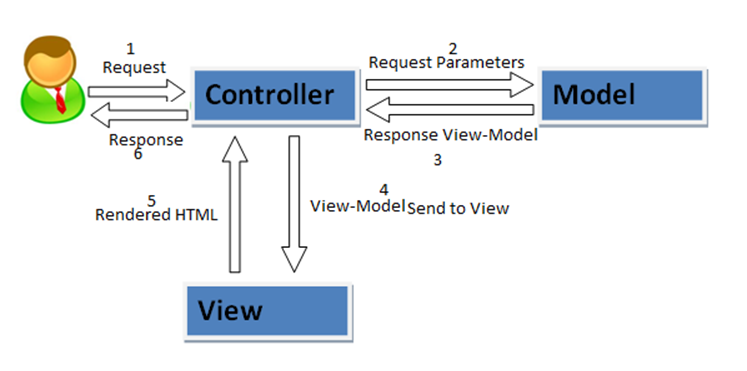
Am utilizat un model arhitectural care separa o aplicatie in trei **Model-View-Controller(MVC).** Fiecare dintre aceste trei componente sunt construite astfel incat sa se ocupe de aspectele specifice de dezvoltare a unei plicatii. Modelul arhitectural **MVC** este cel mai frecvent utilizat.

**MVC:**

* **Model** – partea de model se ocupa de comportarea si datele aplicatiei, raspunde la cereri despre starea sistemului, la cereri de schimbare de stare si notifica utilizatorul atunci cand aceste schimbari au avut loc pentru ca acesta sa poata rectiona.
* **View-** partea de vizualizare transpune model-ul intr-o forma care permite o interctiune usoara, in mod tipic o interfata vizuala. Pot exista multiple view-uri pentru un singur model pentru scopuri diferite.
* **Controller** – partea de control primeste date de la utilizator si initiaza un raspuns in urma cererilor catre obiectele model. Controller-ul este cel care controleaza celalalte doua clase de obiecte, view si model, invatandu-le sa execute operatii pe baza datelor primite de la utilizator.

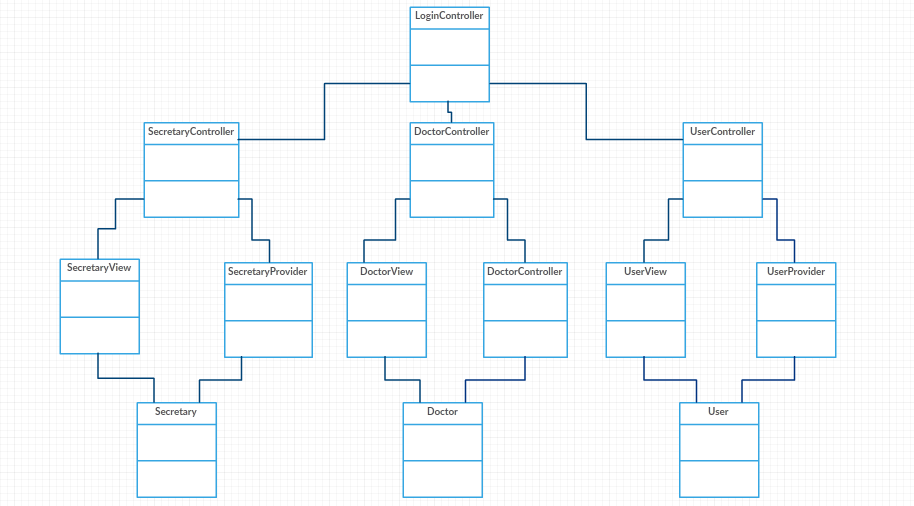
**3.2 Diagrams**

****

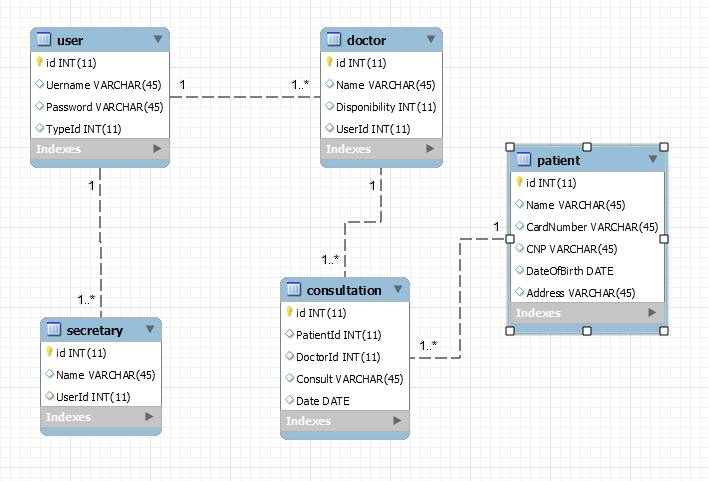
****

4. Class Design

**4.1 UML Class Diagram**

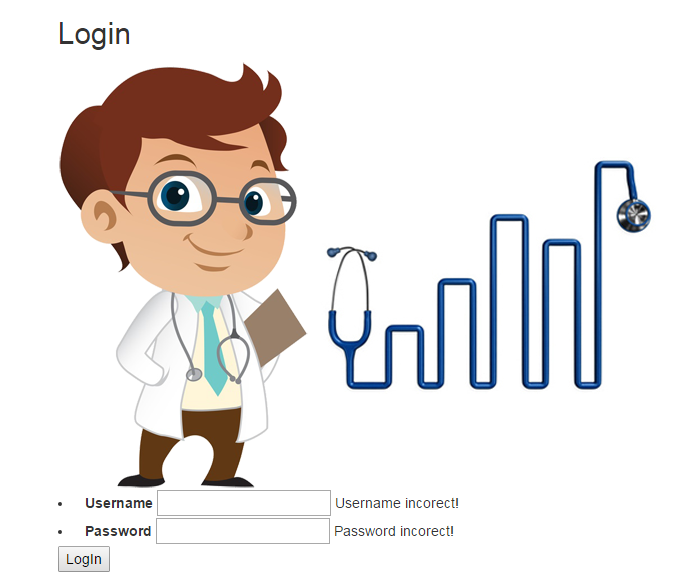
****

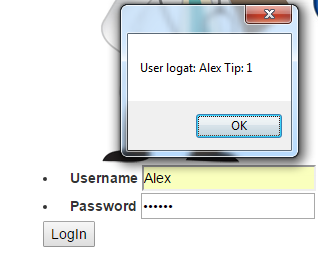
5. Data Model

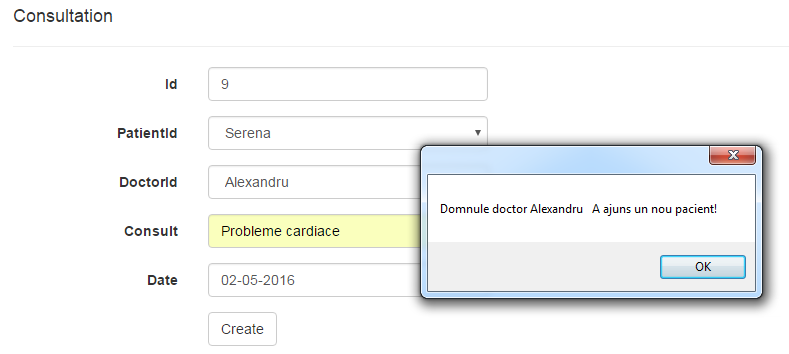
**

6. System Testing

Pentru a testa aplicatia, am creat dialoguri cu mesaje care vor aparea in interfata cu scopul de a anunta utilizatorul ce user e logat sau daca datele introduse pentru logare sunt incorecte. Pentru restul operatiior am introdus date in formulare si am verificat corectitudinea.







7. Bibliography

1. <https://msdn.microsoft.com/en-us/library/jj193542(v=vs.113).aspx>
2. <https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm>
3. <https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx>