<Assignment A3>
Analysis and Design Document

Student:Budusan Ciprian
Group:30434

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

Use Swing/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.

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.

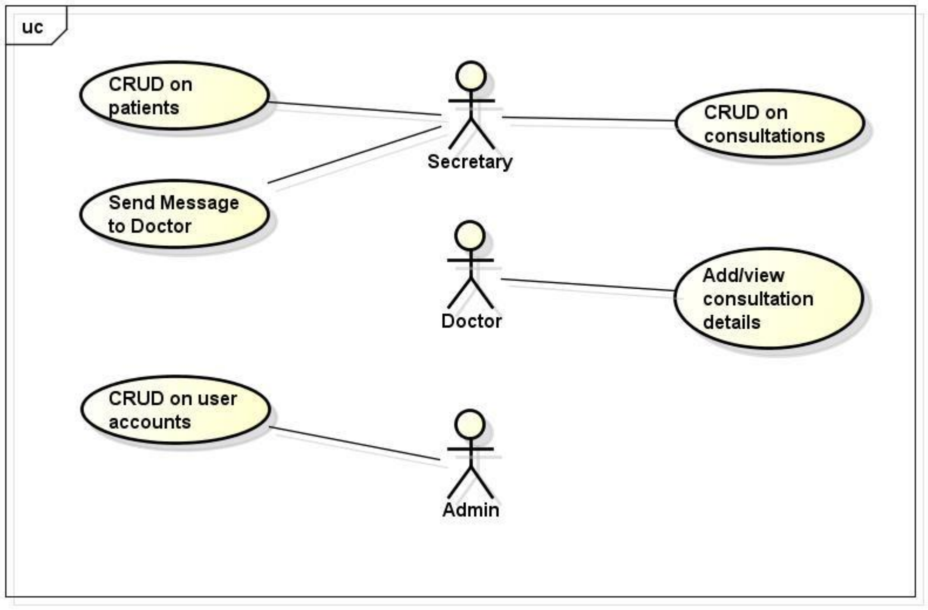
1.2 Functional Requirements

- Authentication: three types of users: clinic secretary, doctor and administrator;
- Authorization levels: admin role separated from user role;
- Administrative functions: perform CRUD operations on patients, consultations and user accounts;
- Send message to doctor informing that the patient has arrived;

1.3 Non-functional Requirements

- Performance: response time immediately;
- Availability: all the time;
- Security: obtained from login interface using Spring Security with different user roles;
- Recoverability: backup for the database;
- Reliability: high, due to banking transactions;

2. Use-Case Model



Use case: Notify Doctor that patient has arrived
Level: user-goal level
Primary actor: clinic secretary
Main success scenario:

1. Secretary logs in with the provided username and password.
2.The secretary panel is displayed automatically by using Spring Security Roles.
3. Secretary needs to connect using the button to start the transmission.
4. Secretary enters the message for the doctor and submits it.
5. The message is being sent by the Server to the doctor.

Extensions: 5a. The doctor is not connected to the transmission.
-5a1. He will not receive the message.

3. System Architectural Design

3.1 Architectural Pattern Description

The DAO layer contains queries and updates to save the domain layer into the database. Interfaces are used to abstract DAO away from the actual datastore in case I want to change datastores.

The Service layer contains the business logic and orchestrates the interaction between the domain layer and the DAOs.

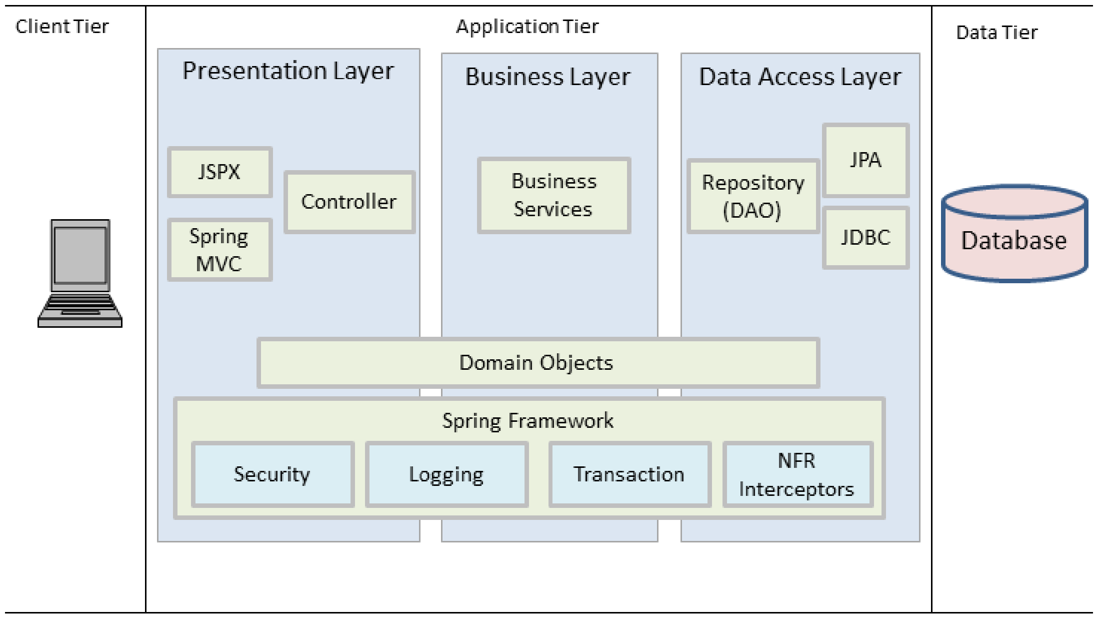
Also Spring MVC is used to ensure the Model-View-Controller pattern and Hibernate ORM framework to persist domain objects to datastore without the need for writing cumbersome SQL statements, acquiring connections and cleaning up those connections.

For the front-end part, view level, JSP files are used with javascript for websockets and Bootstrap theme.

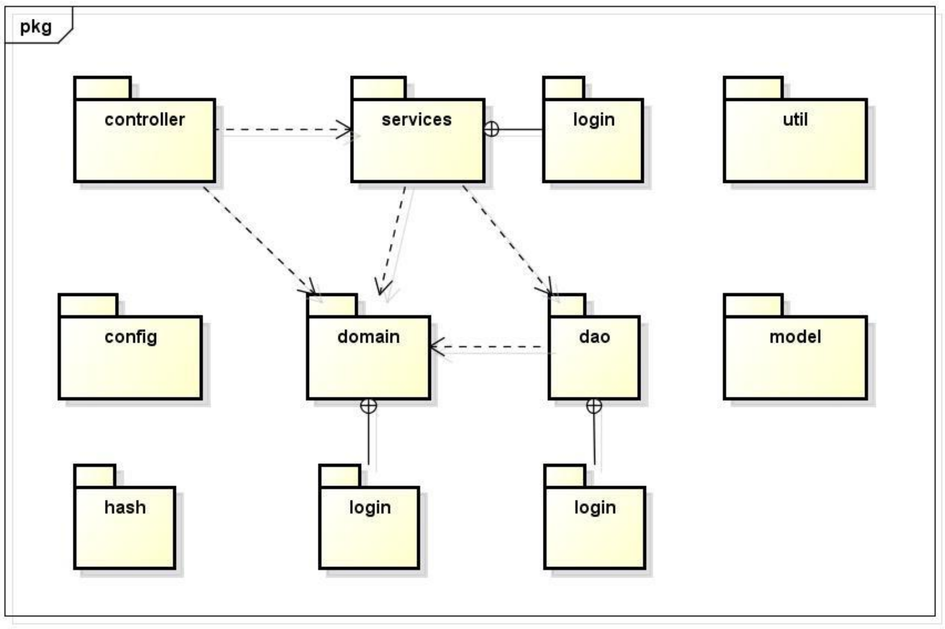
In truth, there is no big difference between DAO and TableDataGateway. Business Logic (Transaction Script) becomes “Service Layer”.

WebSocket is a protocol that provides full-duplex communication, typically between a browser (or another client) and a webserver.

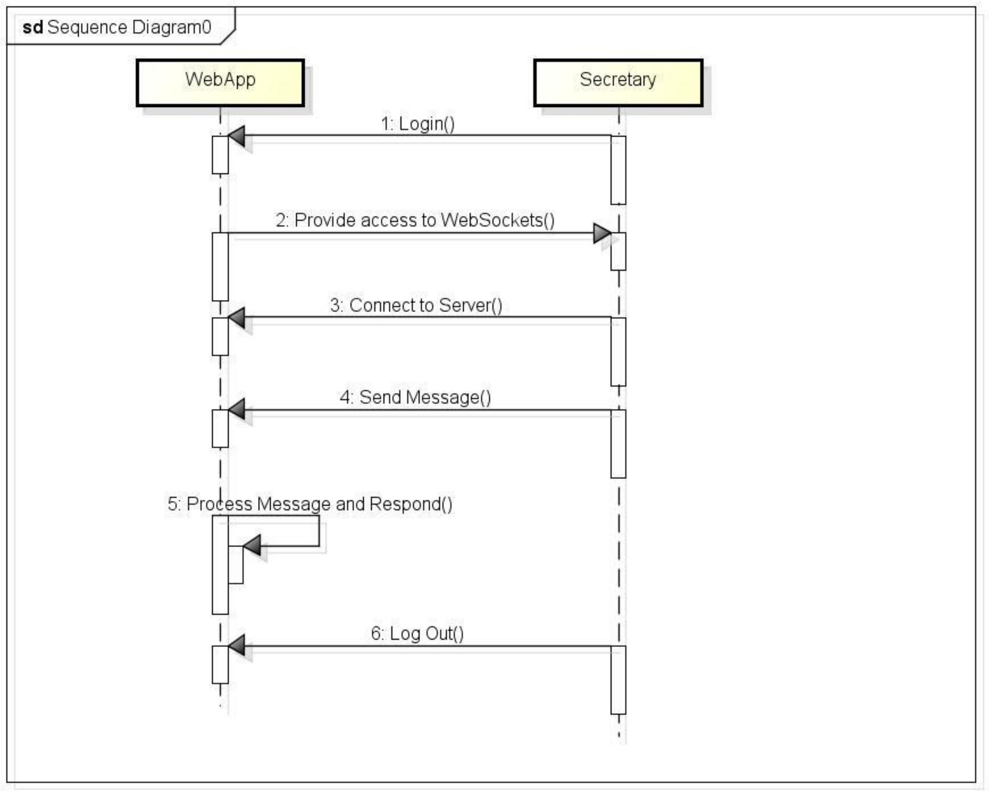
3.2 Diagrams



Package Diagram:



4. UML Sequence Diagrams



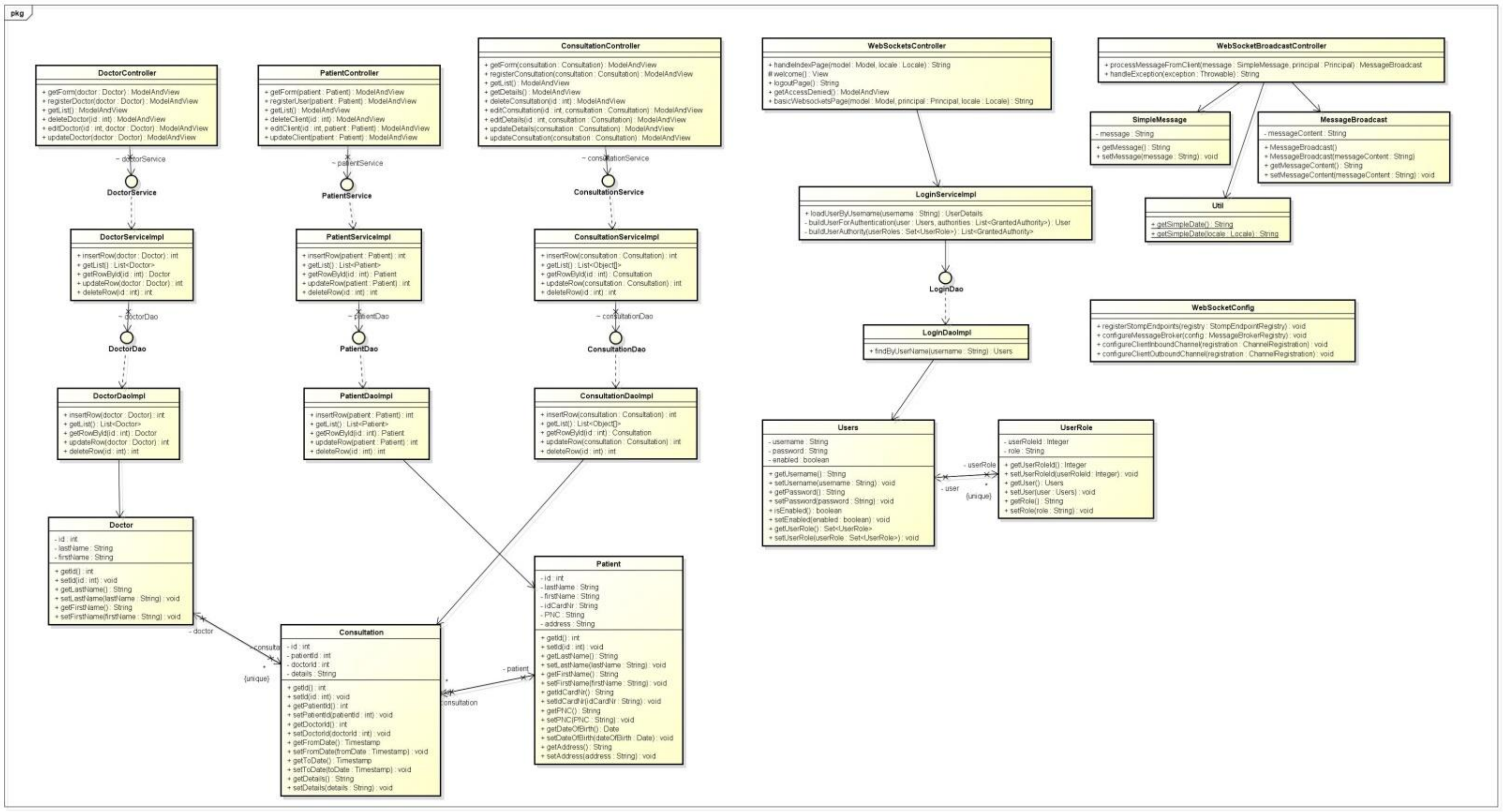
5. Class Design

5.1 Design Patterns Description

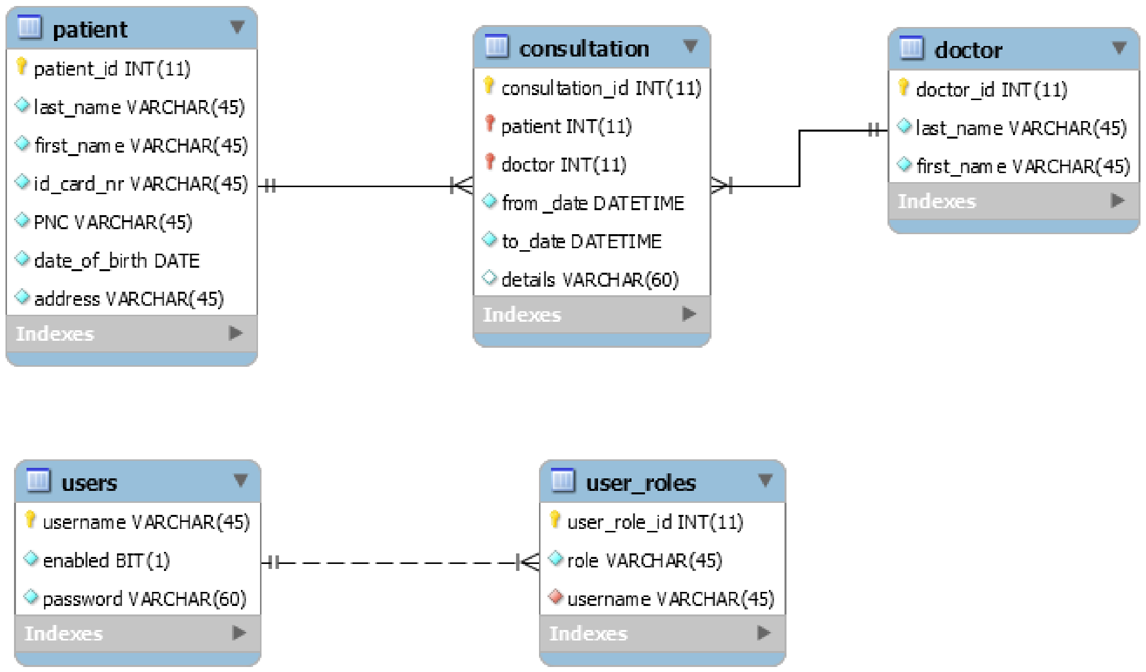
Spring MVC uses the following patterns:

- Dependency injection/ or IoC (inversion of control) – Is the main principle behind decoupling process that Spring does;
- Factory – Spring uses factory pattern to create objects of beans using Application Context reference;
- Proxy – used heavily in AOP, and remoting;
- Singleton – by default, beans defined in spring config file (xml) are only created once. No matter how many calls were made using getBean() method, it will always have only one bean. This is because, by default all beans in spring are singletons.

5.2 UML Class Diagram



6. Data Model



7. System Testing

The Spring MVC Test framework provides first class JUnit support for testing client and server-side Spring MVC code through a fluent API. Typically it loads the actual Spring configuration through the TestContext framework and always uses the DispatcherServlet to process requests thus approximating full integration tests without requiring a running Servlet container.

Client-side tests are RestTemplate-based and allow tests for code that relies on the RestTemplate without requiring a running server to respond to the requests.

8. Bibliography

http://www.concretepage.com/spring-4/spring-4-websocket-sockjs-stomp-tomcat-example
http://www.w3schools.com/bootstrap/
http://forum.codecall.net/topic/72772-integrating-jsp-hibernate-in-an-mvc-application/