Student:Camara Carina
Group:30235

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

The project is an implementation of an application for managing an hospital. The application has three types of users: the administrator, the secretary and the doctor. All the users need to provide an username and a password in order to use the application.

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.

The data is stored in a database as is described in the diagrams. The application uses the Layers architectural pattern for the general organization. Also, all the inputs of the application is validates against invalid data before submitting the data and saving it in the database.

1.2 Functional Requirements

The application uses the Tomcat8.0 server for displaying the views. The data is stored in a database as is described in the diagrams. The application uses the Layers architectural pattern for the general organization. Also, all the inputs of the application is validates against invalid data before submitting the data and saving it in the database.

1.3 Non-functional Requirements

1.4 Availability

Quality attribute definition: <<Availability defines the proportion of time that the system is functional and working. It can be measured as a percentage of the total system downtime over a predefined period. Availability will be affected by system errors, infrastructure problems, malicious attacks, and system load.>>[1]
Source of stimulus: External to System (user sending messages to administrator)
Stimulus: Unanticipated Message (wrong format)
Environment: Normal Operation
Artifact: Process
Response: Inform Operator; Continue to Operate
Response measure: 350 days every year
Tactics[2]:

- Avoid fault become failures
- Fault Detection:
  - between processes (ping/echo, heartbeat message)

  - - within a process (exceptions)
- Fault Recovery
  - - preparation and repair (active/passive redundancy, exception handling, retry, ignore faulty behavior, reconfiguration)
  - - reintroduction (state resynchronization)
- Fault Prevention (Removal from service, predictive model, exception prevention)

1.5 Performance

Quality attribute definition: <<Performance is an indication of the responsiveness of a system to execute specific actions in a given time interval. It can be measured in terms of latency or throughput.>>[1]
Source of stimulus: Users
Stimulus: Initiate transactions
Environment: Under normal operations
Artifact: System
Response: Transactions are processed
Response measure: minim 1 seconds and maxim 10 seconds
Tactics[3]:

- Resource demand
  - - increase computational efficiency
  - - reduce computational overhead
  - - manage event rate
  - - control frequency of sampling
  - - bound execution times and queue sizes
- Resource management
  - - introduce concurrency
  - - maintain multiple copies of data/computations
  - - increase available resources
- Resource arbitration
  - - FIFO scheduling
  - - fixed/dynamic priority scheduling

1.6 Security

Quality attribute definition: <<Security is the capability of a system to reduce the chance of malicious or accidental actions outside of the designed usage affecting the system, and prevent disclosure or loss of information.>>[1]
Source of stimulus: Unauthorized employee
Stimulus: Attempts to modify pay rate
Environment: Normal operations
Artifact: Data within the system
Response: System maintains audit trail
Response measure: Correct Data is Restored within 24 hours and source of tampering is identified
Tactics[2]:

- Detect attacks
  - - detect intrusion
  - - detect service denial
  - - verify message integrity
  - - detect massage delay
- Resist attacks

- - identify actors
  - - authenticate actors
  - - authorize actors
  - - limit access
  - - limit exposure
  - - encrypt data
  - - separate entities
- React to attacks
  - - revoke access
  - - lock computers
  - - inform actors
- Recover from attacks
  - - maintain audit trail
  - - restore

1.7 Testability

Quality attribute definition: <<Testability is a measure of how well system or components allow you to create test criteria and execute tests to determine if the criteria are met. Testability allows faults in a system to be isolated in a timely and effective manner.>>[1]
Source of stimulus: Unit tester
Stimulus: **C**ode unit completed
Environment: Development
Artifact: Code unit
Response: Results captured
Response measure: 50% Path coverage in 3 hours
Tactics[2]:

- Control and Observe System Stat
  - - specialized interfaces
  - - record/playback
  - - localize state
  - - abstract data sources
  - - executable assertions
- Limit Complexity
  - - limit structural complexity
  - - limit nondeterminism

1.8 Usability

Quality attribute definition: <<Usability is the degree to which a software can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use.>>[4]
Source of stimulus: User
Stimulus: Downloads a new application
Environment: Runtime
Artifact: System
Response: User uses application productively
Response measure: Within 2 min of experimentation
Tactics[2]:

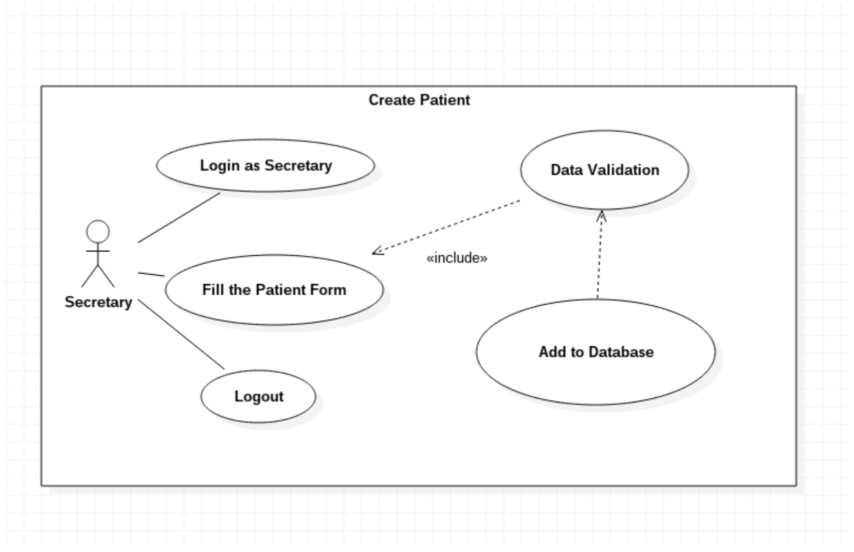
- Support User Initiative
  - - cancel
  - - undo

    - - pause/resume
    - - aggregate
- Support System Initiative
  - maintain task model
  - maintain user model
  - maintain system model

2. Use-Case Model

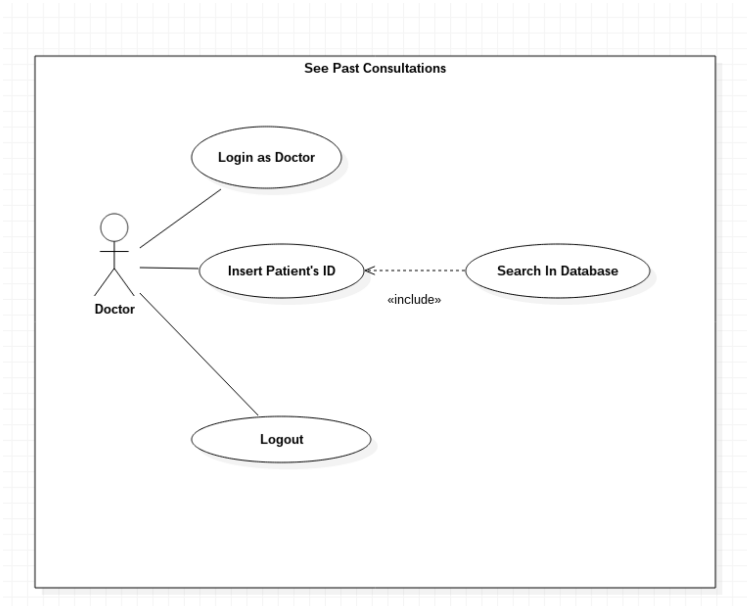
1.1 Create Patient

Use case: Create Patient
Level: User Goal (level 2 sea)
Primary actor: Registered Secretary
Main success scenario: The secrretary authenticate successfully in the application. In the main window, he selects the New Patient button and the page will be redirected to the Patient form. After he inserts all the valid data, he needs to press the Save button and the costumer will be save in the database. The button will redirect the page to the table which contains all the costumers.
Extensions: If the data is valid, the new costumer is inserted in the database and the information will be visible. In case of invalid data, a message will appear next to the invalid field notifying the employee about the error.



1.2 See Past Consultation

Use case: See Past Consultation
Level: User Goal (level 2 sea)
Primary actor: Registered Doctor
Main success scenario: The doctor authenticate successfully in the application. In the main window, he selects the See Consultation link and the page will be redirected to the Search form. The doctor needs to complete the form with a valid patient id and press the Find button. Then, the page will be redirected to the page where all the consultations are listed. He will be able to see only the consultations that are completed.
Extensions: If the data is valid, the consultations will be listed. Otherwise, the page will be empty.



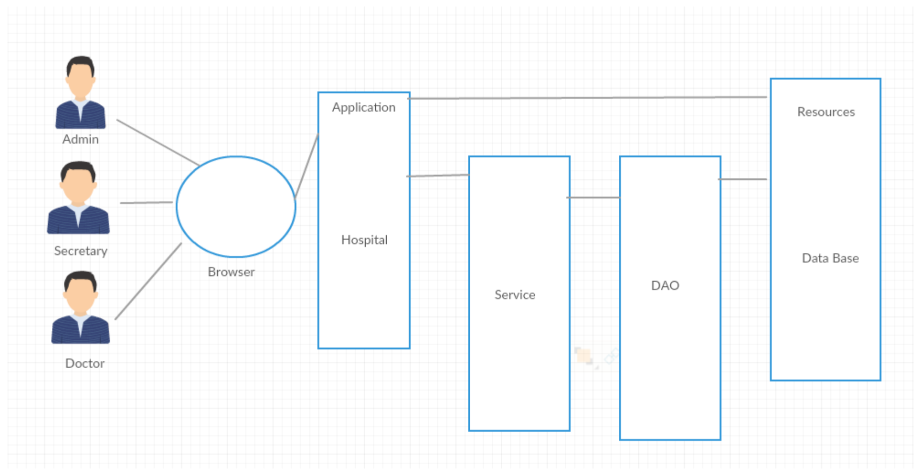
3. System Architectural Design

3.1 Architectural Pattern Description

The Layers architectural pattern has three primary layers: presentation, domain and data source.
Presentation logic is about how to handle the interaction between the user and the software. The primary responsibilities of the presentation layer are to display information to the user and to interpret commands from the user into actions upon the domain and data source.
Data source logic is about communicating with other systems that carry out tasks on behalf of the application. These can be transaction monitors, other applications, messaging systems, and so forth. The biggest piece of data source logic is a database that is primarily responsible for storing persistent data.
The domain logic, also referred to as business logic. This is the work that this application needs to do for the domain we are working with. It involves calculations based on inputs and stored data, validation of any data that comes in from the presentation, and figuring out exactly what data source logic to dispatch, depending on commands received from the presentation.

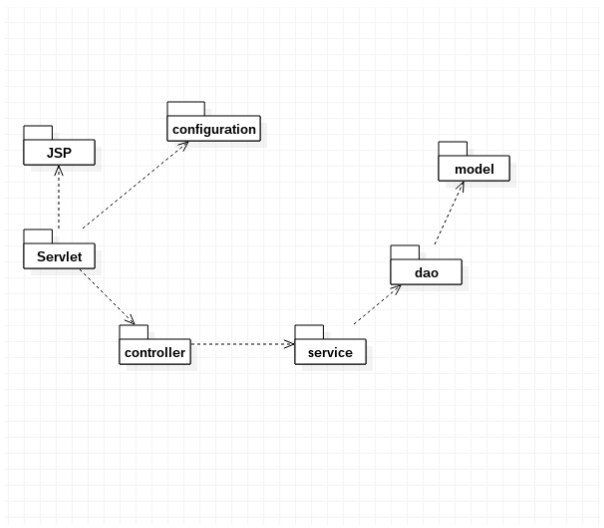
3.2 Diagrams

System's conceptual architecture



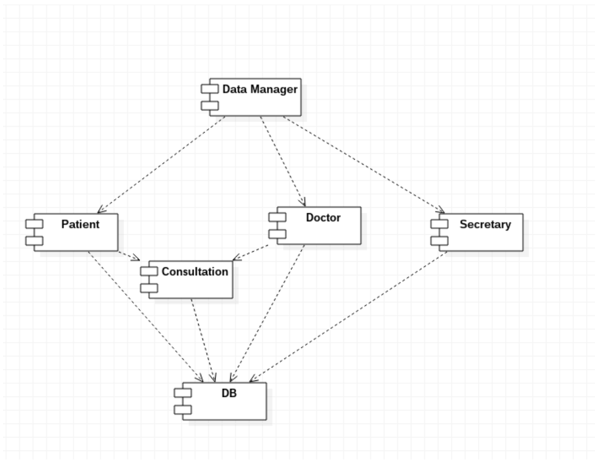
The users interact with the applications through the browser. The navigation is controlled by the application and all the inputs are sent to the application. The service contains all the methods needed and the resources are taken from the database.

Package diagram



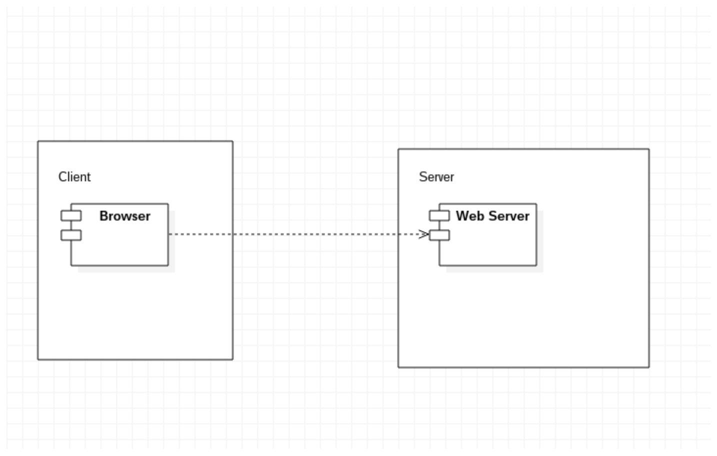
As we can see in the package diagram, the project implements an MVC architecture. The package Servlet which contains the web.xml file and the dispatcher file calls the Controller package. This package contains the functions with the annotation @RequestMapping which control what pages to be displayed. The JSP package contains the files for the diplay content (View). The Controller calles the Service package for executing the requested services.

Component diagram



The component diagram model the architecture of the application in terms of its physical inter-dependent components.
The Client and the Transactions are accessed through the Data Manager called by JSP. Those two components are binded with the Consultation. All the informations for these components are accessed from the database.

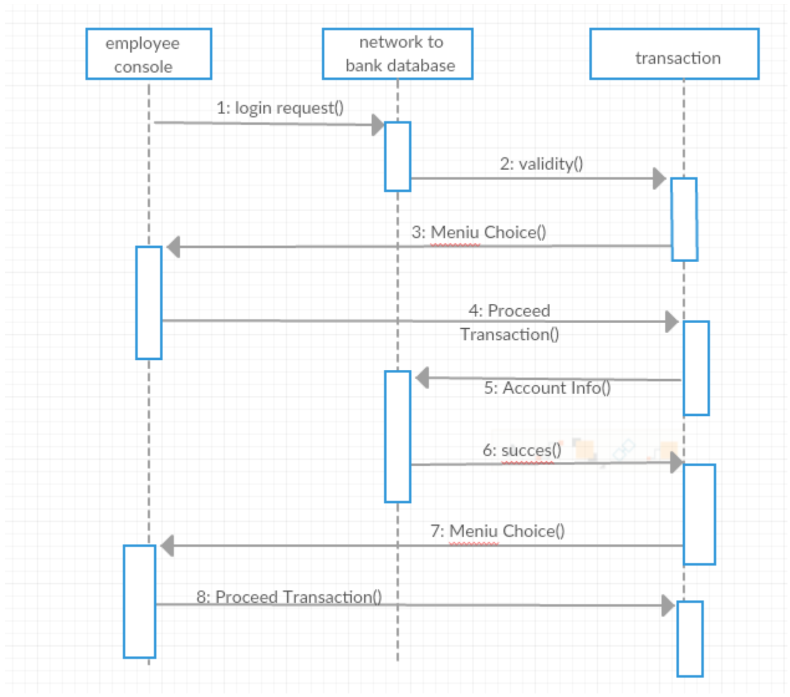
Deployment diagram



Deployment diagrams model the hardware used in system implementations, the system components deployed on that hardware, and the associations between those components. Also, it tells what is installed where and with what it communicates. [1]
In this diagram we can see how a JSP file is viewed. The Client, through the browser, connects

with the Server. The Web Server sends a HTTP Request to the JSP engine. After the Servlet Engine takes the request, it gives a response to the Web Server which displays the JSP file content.

4. UML Sequence Diagrams

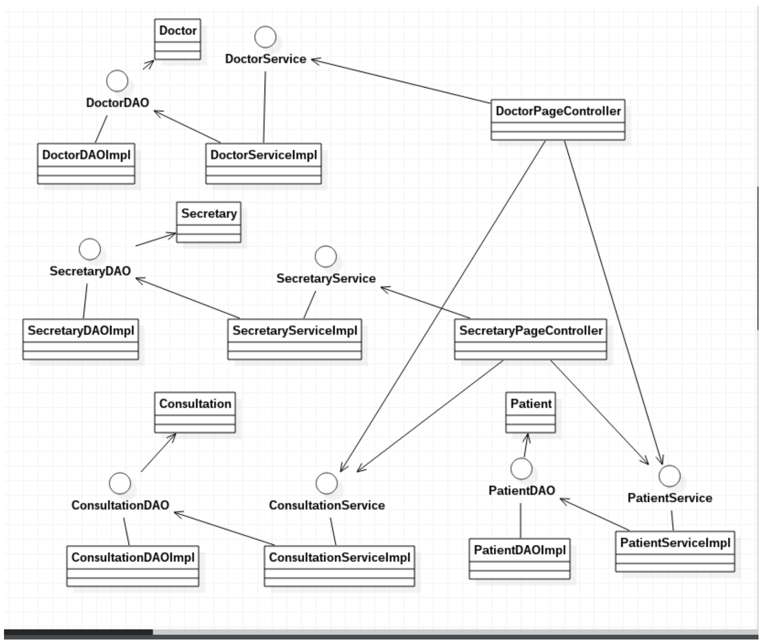


5. Class Design

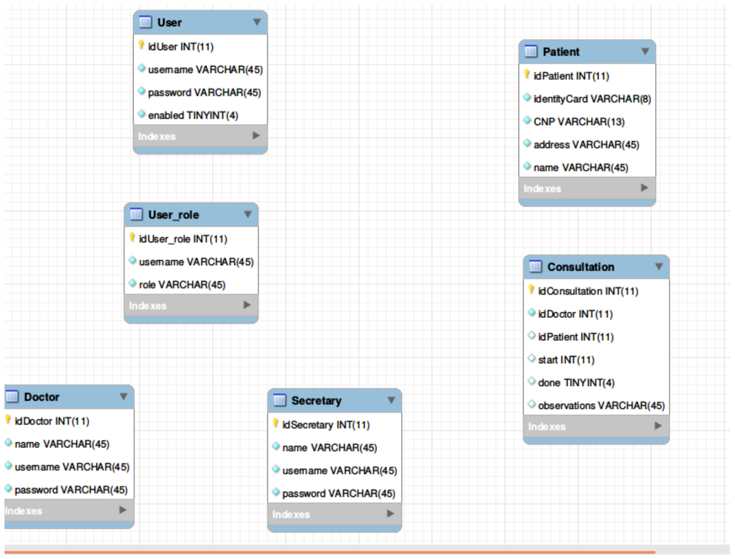
5.1 Design Patterns Description

The Transaction Script organizes all this logic primarily as a single procedure, making calls directly to the database. With this domain logic pattern the domain logic is primarily organized by the transactions that we carry out with the system.
A Table Data Gateway holds all the SQL for accessing a single table or view: selects, inserts, updates, and deletes. A Table Data Gateway has a simple interface, usually consisting of several find methods to get data from the database and update, insert, and delete methods. Each method maps the input parameters into a SQL call and executes the SQL against a database connection.

5.2 UML Class Diagram



6. Data Model



7. System Testing

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements.

Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

A test case is a tool used in the process. Test cases may be prepared for software verification and software validation to determine if the product was built according to the requirements of the user.

The application tests all the inputs against invalid data before inserting them in the database. Also, in the source folder src/java/test are stored the Junit tests. The class CostumerTest contains tests for CRUD operations. First a new costumer is inserted in the database, then the data is modified and deleted. All the tests are completed successfully.

8. Bibliography

https://www.tutorialspoint.com/software_testing_dictionary/validation_testing.htm
https://en.wikipedia.org/wiki/Software_verification_and_validation