MedPat: A Doctor - Patient Platform

Analysis and Design Document

Student: Turcu Lucian Andrei

**Group: 30432**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 19/Apr/18 | 1.0 | First iteration. Domain Model, Architectural Design, Deployment diagram | Turcu Lucian Andrei |
| 24/Apr/18 | 1.1 | Design model, Data model | Turcu Lucian Andrei |
| 28/May/18 | 2 | Final Documentation | Turcu Lucian Andrei |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

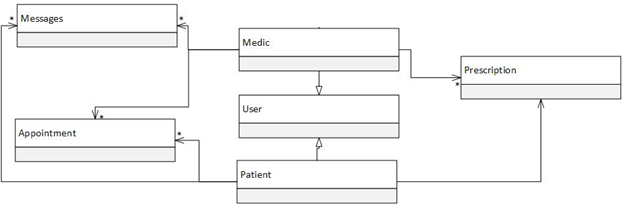
MedPat is a platform whose purpose is to facilitate the communication between medical staff and their patients. Using MedPat, a medic can answer the patient’s questions remotely and write prescriptions for a patient. A medic can also keep track of upcoming appointments and remove the ones that are past their date.

The patients can book appointments to a specific doctor, view prescriptions and send messages to their doctor.

# Elaboration – Iteration 1.1

# Domain Model

This is the general representation of the relevant concepts that have to be modelled in the software application, including the relations between them and the multiplicity factors.

**

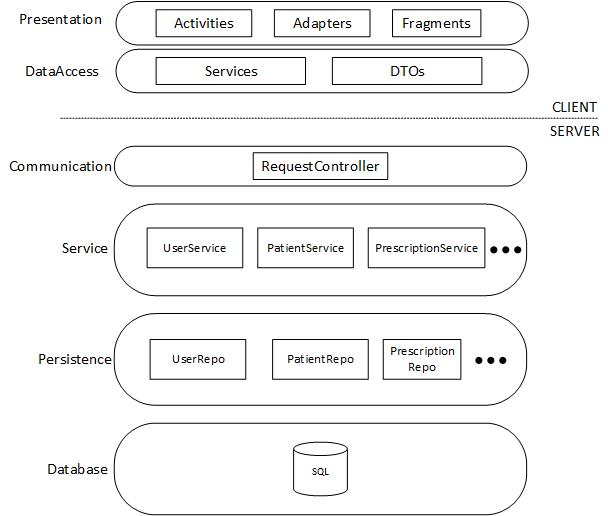
# Architectural Design

## Conceptual Architecture

The system is based on a client-server architecture. The client is the Android application, through which the users interact with the system. It will implement the Builder design pattern to instantiate data transfer objects using data received from user input.

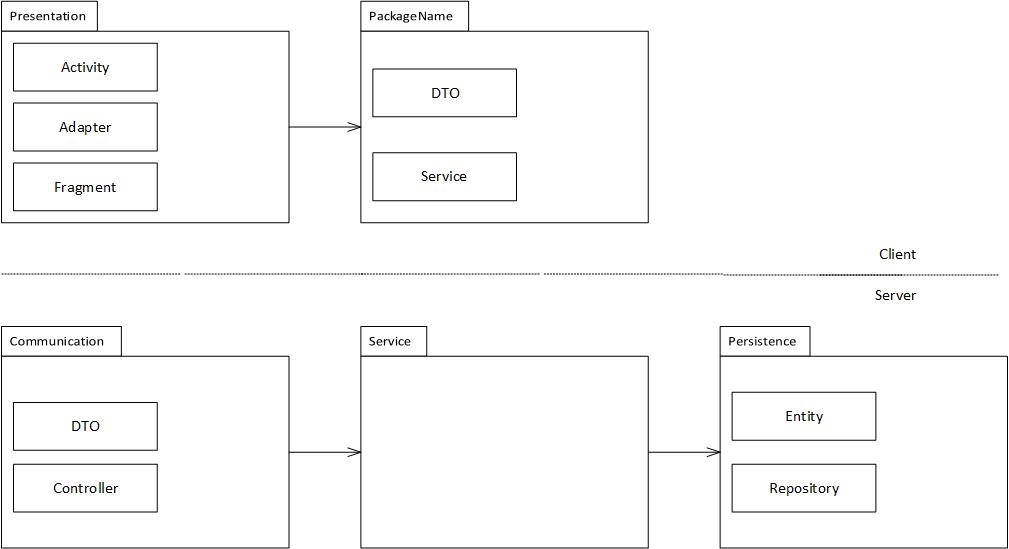
The desktop application processes the client’s requests and retrieves data from the database. This way, multiple users can use the application at once, using a common data set. The RequestController class is representing the entire server system, according to the Façade design pattern.

The server application’s architecture is a layered one, to achieve separation between the different levels of functionality. The layers are repository, service and communication. Between the layers, communication is only available from a layer to the one directly above or below.

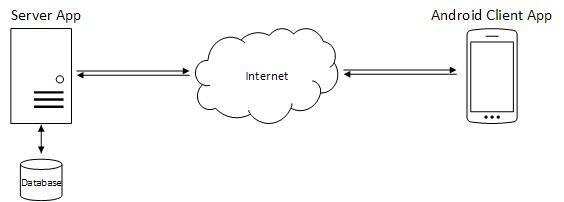


*<layered architecture>*

## Package Design

**

## Deployment Diagram

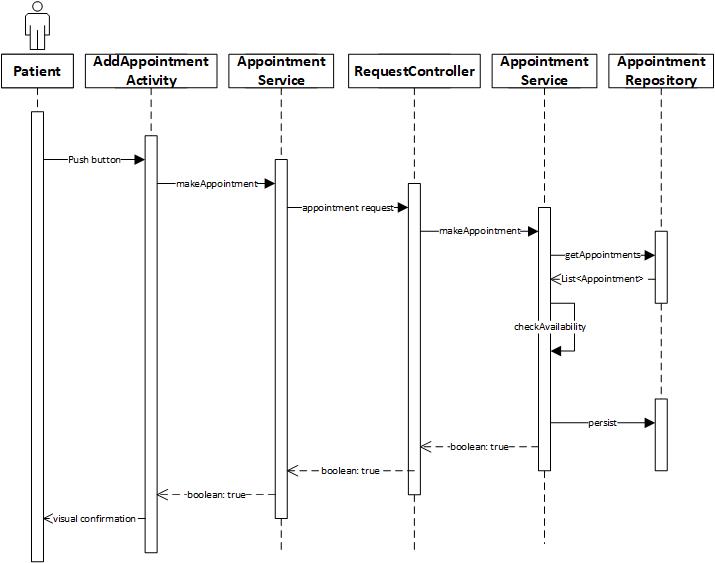


# Elaboration – Iteration 1.2

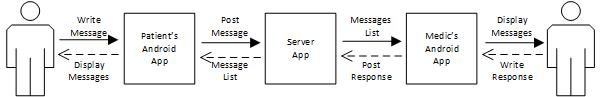
# Design Model

## Dynamic Behavior

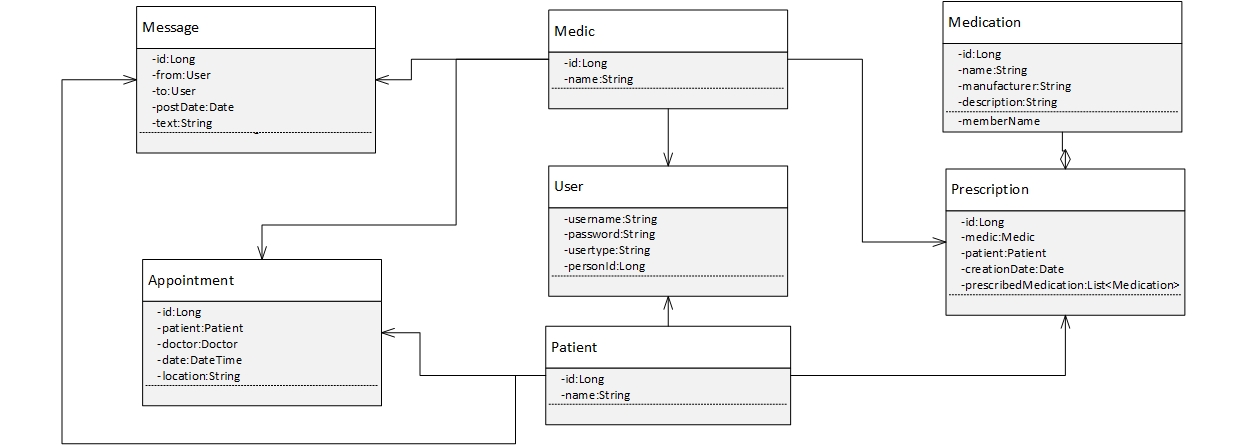
*Sequence diagram for the “Make Appointment” scenario*

**

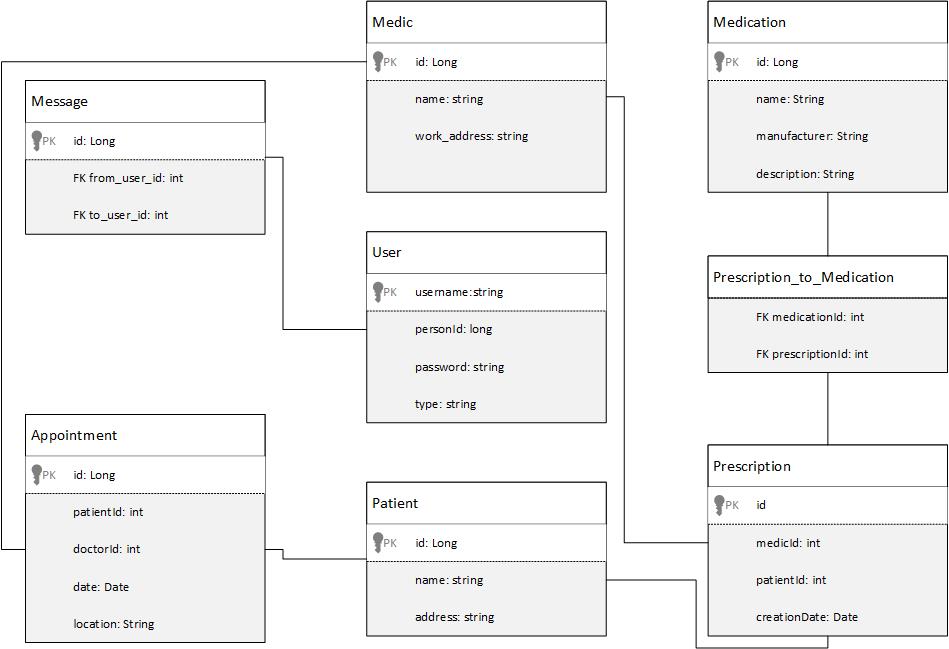
*Communications diagram for the “Request Remote Consultation” scenario*



## Class Design

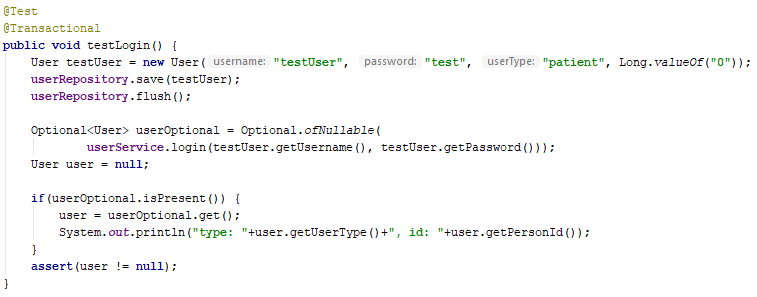
**

# Data Model

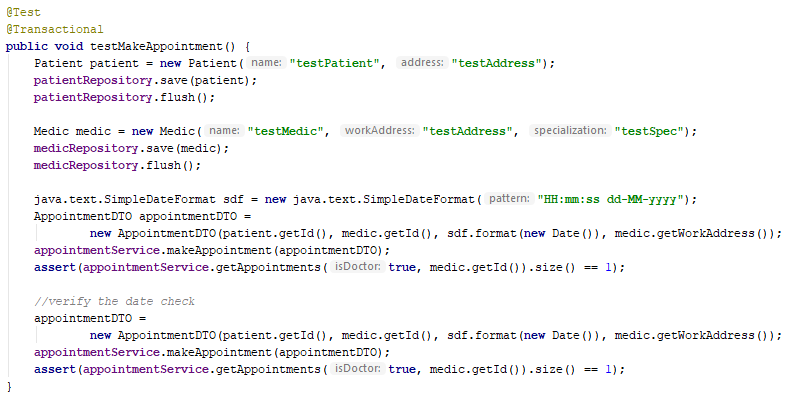
**

# Unit Testing

The following method tests the user login by creating a fictive test user account. Then checking that the value returned by the call to userService.login() is the expected one.

**

The other method has the purpose of testing the makeAppointment method in the appointment service. It creates a medic and a patient account with fictive data then adds two appointments with very close dates to verify if the checks work accordingly.



# Construction and Transition

# Future improvements

Future improvements could include a notification system for the messages. This would actually require the migration from an SQL database to the Firebase service.

Also, the application is lacking a registration section. This is intended, in the case of medics, for the sake of security. For patients, on the other hand, it is not necessarily the case, therefore a registration screen would prove useful to make the administrators’ life easier.

There’s also the need for an automatic appointment cleanup system to remove appointments that are past their date. Currently, the medic must manually remove the appointments from their agenda.

# Bibliography

<http://square.github.io/retrofit/> - Android API for creating HTTP requests

<https://stackoverflow.com/> - The answers for coding and life questions

<http://spring.io/> - Spring Framework website

<https://www.youtube.com/> - Tutorials for anything, also cat videos.