eHealth Documentation

Java - Objected Oriented Programming

Johannes Jobst
Marin-Petru Hincu
Pascal Kautzmann
Hania Anjum Chatha

February 11, 2021

Contents

List of Figures

Chapter 1

Introduction

(Johannes Jobst)
This documentation is about our results of creating a smart eHealth consulting system. We give an overview how our application is structured, how we proceeded the project, how the task were be distributed in the team and much more.

Chapter 2

Project motivation

Chapter 3

Project description

(Johannes Jobst)
Our eHealth application is designed that a user doesn't anymore need to call a doctor via the telephone if he or she has a health problem.
Instead it provides a whole new experience for people to creating an appointment. They now are able to create an account in our eHealth application. With this account they can login into the system and specify their health problem. The smart application then displays based on the symptoms you have and the distance a doctor should be, all suggested doctors for you. In the next step the user is able to make an appointment at one of these doctors. Furthermore our eHealth application displays for logged in users all upcoming appointments they have. Adding to that the user has the ability to edit or delete a certain appointment.
Because our user storing is implemented in a database our eHealth application has an admin account which is available with special login data. In this admin view the admin is able to manage the users like edit personal information of them but also deleting them.

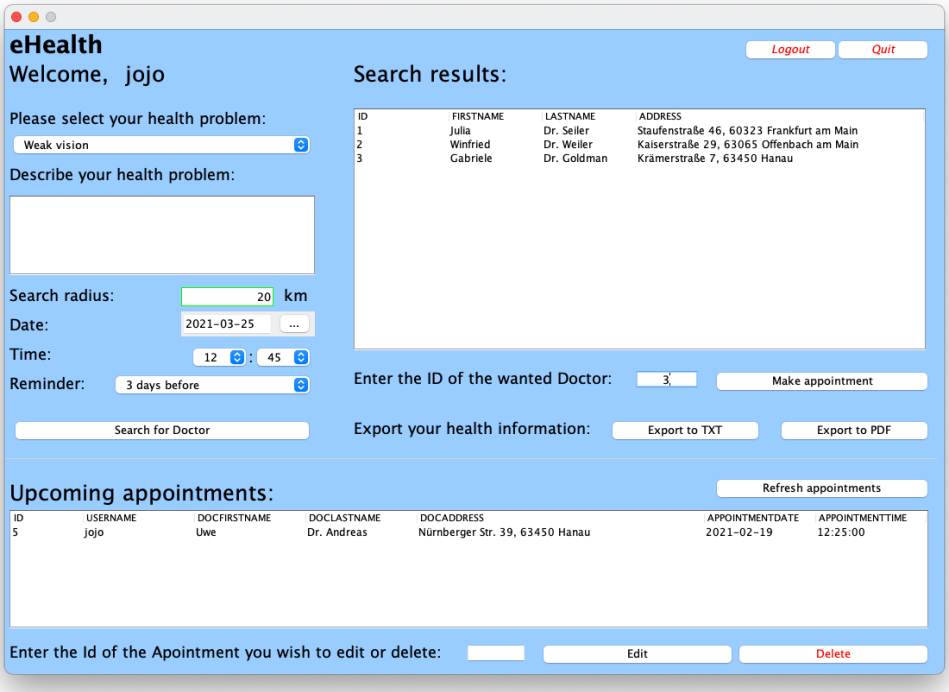

Figure 3.1: eHealth main window under MacOS

Chapter 4

Team organization

4.1 Meetings

(Johannes Jobst)
At the beginning of the project we first of all thought it would be very useful to have a fixed day in the week to have a meeting about our project.
In our weekly meetings we discussed how to do and implement certain tasks in the project. So always at the end of our meetings we came to a task distribution and defined who is responsible for which task.

4.2 Task distribution

Johannes Jobst was responsible for everything concerning the Graphical User Interface. He also implemented the most parts of ERROR handling. For example the check whether the given input by the user was correct. But also checks that appointments could not be made in the past.

Marin-Petru Hincu designed the database, created the necessary tables and embedded the database into the project. He provided methods for connecting to the database, manipulating, and retrieving data from the database. Marin did also implement the user class and its functionality.

Pascal Kautzmann was responsible for the e-mail, the password hashing, the reminder function and the usability of the OpensStreetMaps-API

4.3 Development timeline

(Johannes Jobst)
At the beginning we started to create a GitHub repository for our project. We then constructed a basic GUI construct for the login window. With his basic construct we already were able to implement our database in the project.
After this we were already basically been able to create new accounts and add them to the database and login with them or with already existing accounts to our system.
In the next step we added password encryption to the login data.
We then thought that a convert of our project to a Maven project would be very handy, because of ...
After that we build the admin window and add step by step the full functionality to the GUI of the admin. We created a table to display all users in the database. Made buttons to edit a user via the "editUserWindow" or delete a user.
After finishing with the admin window we focused more on the main part of the application. Therefore it was required to implement a 2 factor authentication. We implemented that with a new window.
As a next step we started to develop all functionalities for the main window. We added all necessary GUI elements and connected them to the database. We implemented a "editAppointment" window to shift upcoming appointments. Concomitant with that we finally implemented the reminder functionality and also the health information export function.
So finally our program is finished. At the end we had only to fix some bugs where something was not probably working.

4.4 Tools

(Johannes Jobst)
For our project the whole team used the Eclipse IDE, because of some small differences in other IDEs. In Eclipse itself we used the WindowBuilder to design our GUI.
Our main tool for working on the project as a team was GitHub. There we have a central repository for storing all project data like the complete source code but also the project documentation. It makes the working on the project very comfortable and easy because of the ability to work simultaneously on the code.

Chapter 5

Technical description

5.1 Requirements

5.1.1 Overview

The following figure represents all components needed for the project and their correlation.

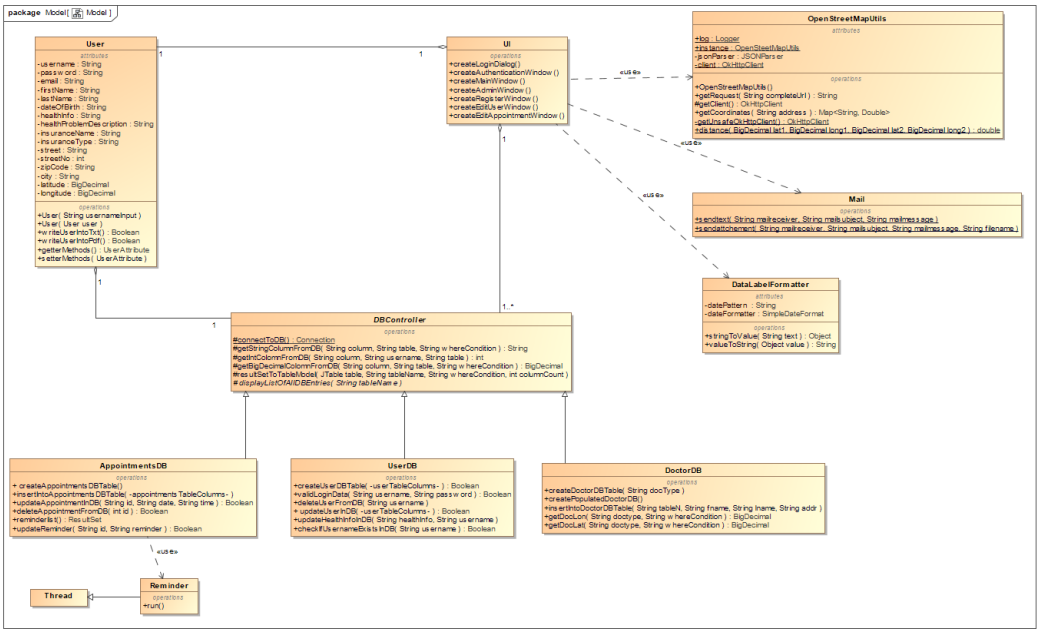

Figure 5.1: Class diagram for the eHealth project

The user class represents the user of the eHealth application. Every piece of data needed for the user is stored as an attribute. Some methods allow for

modifications and obtaining information about the user. Methods for writing user information into an txt or pdf are also available.
DBController contains important methods for connecting to the database modifying its contents and retrieving data. The classes AppointmentsDB, UserDB and DoctorDB are derived from the DBController and are used for creating and querying the respective database tables. The AppointmentsDB also uses the class reminder which allows threading the appointments table. The class named UI represents all User Interfaces used in the project. Some graphical user interfaces make use of the class OpenStreetMapUtils, which provides useful functionality for determining the location of a user or a doctor and calculating the distance between both. The class Mail is also used for sending out E-mails whenever this is necessary in the project. Finally, the class DateLabelFormatter is used by the UI to ensure correctly formatted dates for the appointments.
As illustrated in Figure 5.1, each user contains one DBController and the UI contains one User and multiple instances of the DBController. The reason being that in some cases the UI needs to have access to all three database tables.
The following pages will go in more detail about the components of the UI and their interactions.

5.1.2 Activity overview

(Johannes Jobst)
In Figure 5.2 you can see how the activity flow of our program works.
First of all our program displays an login window where you have the choice to create a new account or to login with your account into our eHealth system. If you type in your username and your password correctly the system will check the authentication via 2 factor authentication, so it sends you a mail with a random code which you must type in the dialog. After validation the system checks whether you are an admin or a user.
In the admin view the admin has the ability to show up all registered users and can edit their personal data or delete them. In the user view the user has the ability to make appointments after selecting his health problem and providing some more data. The system then searches for matching doctors and displays them. Now the user can choose his doctor out of the results and make an appointment.
Additionally the user can display the upcoming appointments and make changes of the date and the time or even delete the appointment. The last main functionality he has is that he can export his health information either in pdf or txt format.

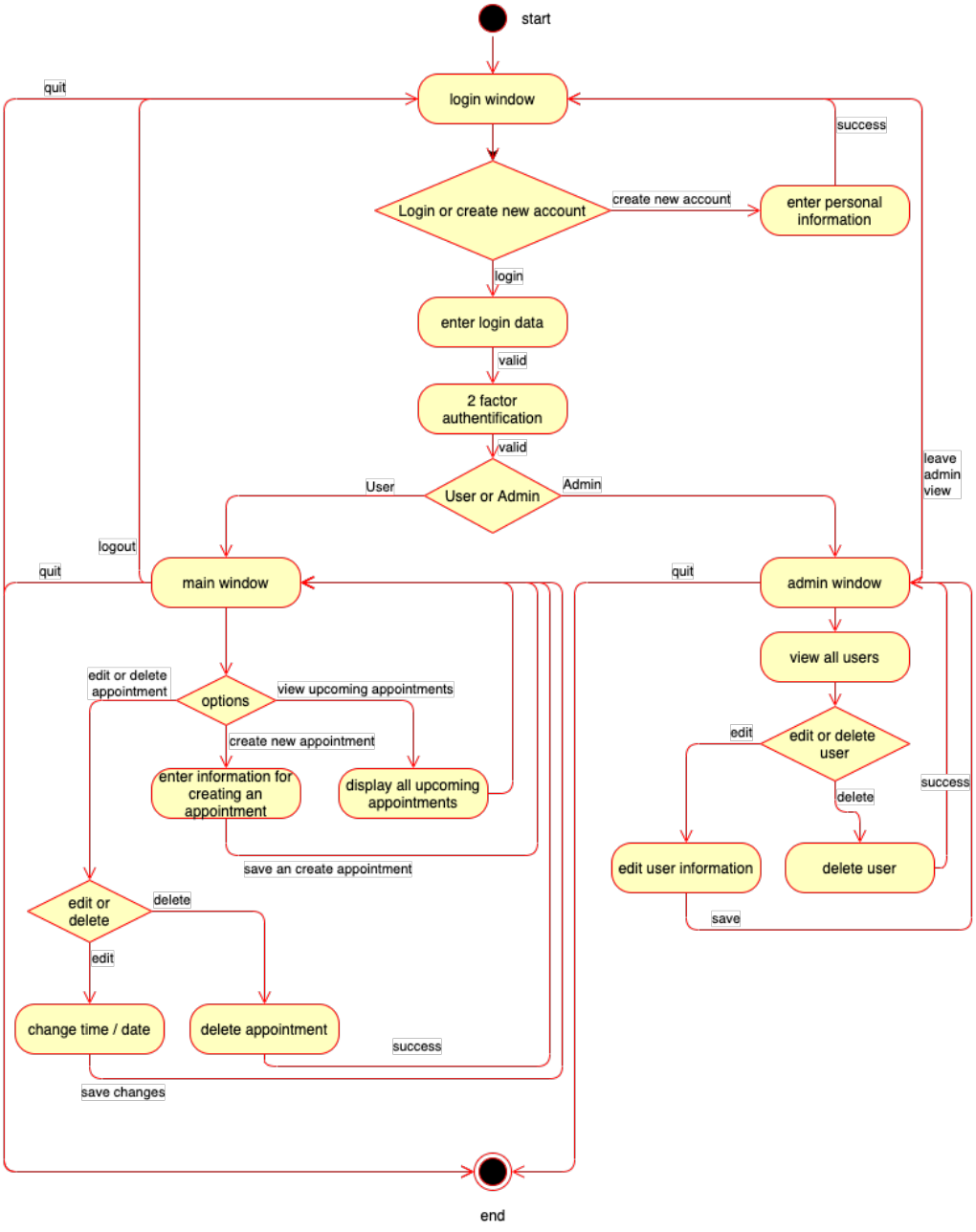

Figure 5.2: Activity diagram for the eHealth project

5.1.3 Account creation

(Johannes Jobst)
The first main requirement of our program is the account creation.
If the User uses the application for the first time he has the ability to create a new eHealth account. Therefore he has to enter his firstname, lastname, email address, address information, date of birth, insurance type and his insurance name. If the user confirms, the entered data is checked for correctness. The Validation of the email address happens by checking if the email has the right format (eg. max@mustermann.de). For validating the address we use the API of OpenStreetMaps and check if the given address exists. The username must be unique in our database, so the system checks whether its already existing. For the insurance type one of the given types must be selected. For the rest of the Fields the system checks that they aren't empty and not longer than the capacity of the corresponding field in the database. With this error handling we ensure that ours system doesn't run into database errors.
If every information is typed in correctly the account is now created.

5.1.4 Login

(Johannes Jobst)
If the user already has an account he is able to login with his username and password.

5.1.5 Password

(Pascal Kautzmann)
The password is saved as a hash, because it is really difficult and time consuming to recreate a hashed string. The password is hashed with the from javax.crypto provided functions. It uses a SHA512 algorithm to hash, the password is also is"Salted", so that if two useres have the same password, the hashed password is still different.

5.1.6 E-Mail

(Pascal Kautzmann)
To send e-mails the javax.mail classes are used. An e-mail adress to send e-mail is predefined because mail services, like g-mail, don't allow unverified apps as long as less secure app access is deactivated.

5.1.7 Admin view

(Johannes Jobst)

If the user enters admin login data into login dialog the system will open an admin view. The admin there has the ability to view all users. This is implemented via an SQL query out of the user table in the database. The admin has also the option to edit the personal informations of users. Therefore a looking almost like the register window pops up, where he is able to make changes. With the same error handling like in the account creation the system is prevented from any errors.

5.1.8 Searching for a doctor

(Pascal Kautzmann)

In the main window the user has the possibility to search for a doctor, for that the user has to provide some data.

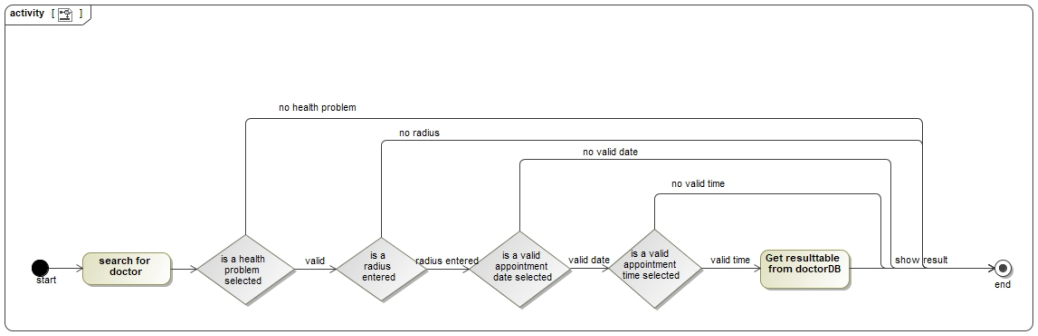

Figure 5.3: Activity diagram for searching a doctor

First the user needs to select their health problem with the help of a drop-down list, where we predefined some health problems . Then the user has the possibility to better describe their problem nad give some more information. Next they have to select the search radius, so it is possible only search for a doctor in the given radius around thier address. Now the user can select the date and time when they want to have an appointment. The last information needed is whether they want to receive a reminder and how long before the appointment. If the user presses the button "Search for Doctor" the programm first checks if all necessary data is given and in a correct format. Then it gest the list of possible doctors from the Doctor-Database and and displays them so the user can select one.

5.1.9 Editing an Appointment

In the main window the user has the possibility to view the appointments he made. He can also edit or delete these appointments.

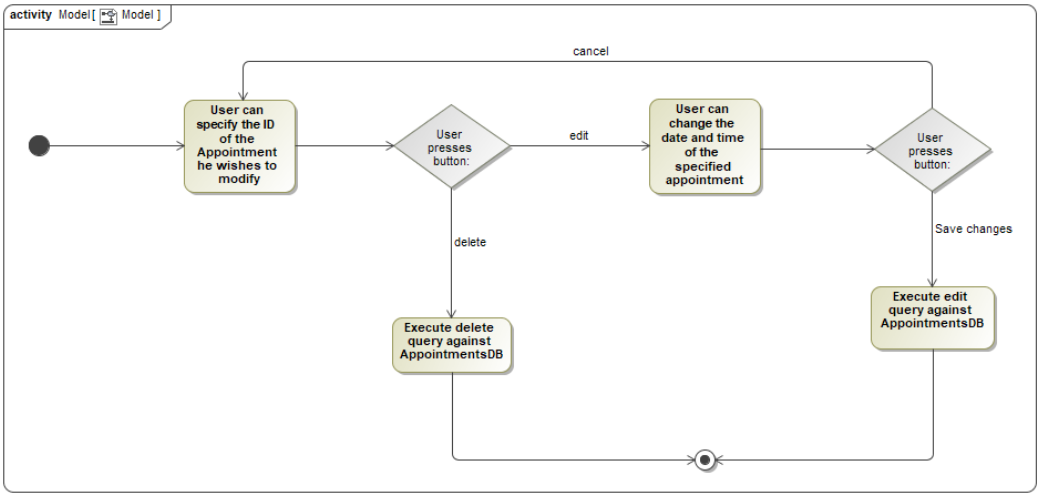

Figure 5.4: Activity diagram for deleting and editing an appointment

In the appointments table displayed in the main window, the user has to select the id of the appointment he wishes to edit or delete. After the user introduced the id into the respective text field, he can either press the button "delete" or "edit". If the user presses "delete", he must first confirm the action and then the selected appointment is deleted from the table. If the user presses "edit", he gets redirected to a new window, where he can then change the date and time of the appointment. The user must then press "save" for the changes to be saved in the appointments table, and he returns to the main window. However, if the user presses "cancel", he returns to the main window without modifying anything.

5.1.10 Reminder

(Pascal Kautzmann)
The reminder runs in a separate thread, so it dosn't interfere with the rest of the program. The reminder will check every minute whether there is an entry in the Appointment-database that requires a reminder. Then it sends to the user who wants the reminder an e-mail with the appointment details.

5.2 Problem facing

Chapter 6

Conclusion

Chapter 7

Sources