Architecture Document
Tuberculosis Treatment Application

Iteration 2 (April 5, 2018)

Client: E.M. Koops
e.m.koops@student.rug.nl

Teaching Assistants:

Charles Randolph
Frank te Nijenhuis

Team Members:

Teodor Ionut Oanca
Giacomo Casoni
Rutger Berghuis
Andrei Scurtu
Sten Sipma
Julius van Dijk
Noam Drong
Hidde Folkertsma
Marco Lu
Pieter Jan Eilers
Sytze Tempel
Roel Brandenburg
Robert Riesebos
Niek de Vries

Contents

1 Introduction

Tuberculosis is a dangerous disease that over the years has become a very rare disease in first world countries. One reason for this is the treatment that is possible. The treatment of tuberculosis entails months of taking different kinds of antibiotics. It is very important that the patient will take all his medication and finish the treatment. This is important for the health of the patient but also because otherwise the bacteria can become resistant against the antibiotics. For a lot of patients it is hard to remember to always take their medicine, this is where the tuberculosis treatment app comes to help reminding them when and which medication to take. Also the app provides them with information about tuberculosis and will include achievements to help motivate the patient to finish their treatment.

We decided to split our team into three sub-teams: the webapp team, the ios team and the android team. The android and ios app are the front end of the application for patients and allow users to check their medication schedule, read information and watch videos about tuberculosis, answer quizzes, ask questions to their physician and other useful features. In terms of design we keep it as native as possible and take notion of other apps such as samsung health to create a similar slick design according to the customer. The webapp has a front end that a physician can use to add their patients and create a treatment plan for them. The android, ios and webapp all push and retrieve information to and from a database through API's. The database stores the accounts of the patients and physicians with their corresponding treatment plan and other necessary data.

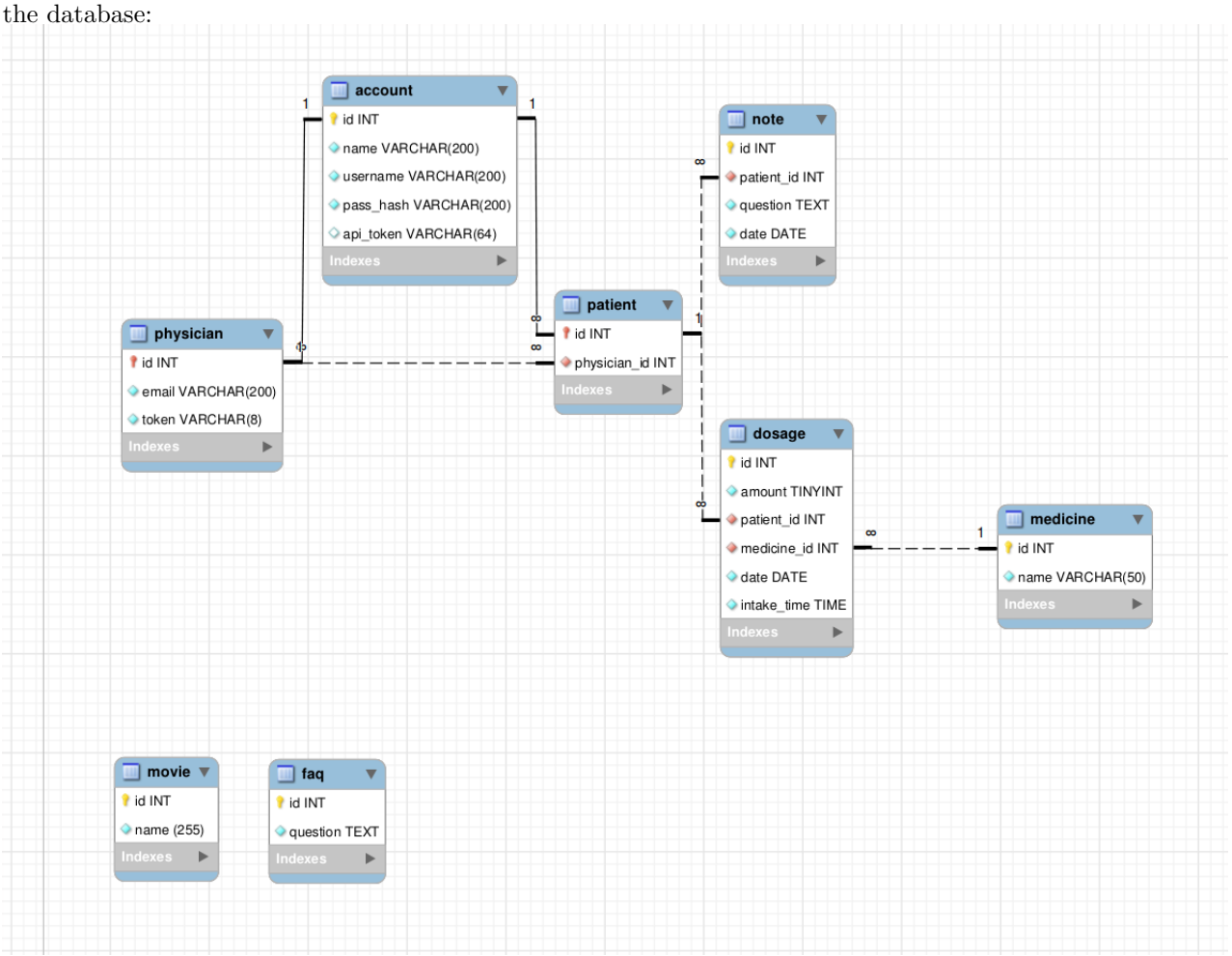
2 Architectural Overview

What components are there and what are their functions? How are components connected/communicating? If a more general principle or a paradigm is applied that is worth mentioning, mention it.

2.1 Webapp

2.1.1 database

The database stores all the data from the patient and physician accounts. This is the layout of the database:



An **account** consist of a ID integer linked to the account, a name, a username, a password hash and an API token linked to the account. The account can be either a physician or patient account.
A **physician** has stored an ID int so it is linked to an account, a email and a token. This token is used to make a patient account.
A **patient** has stored a ID int so it is linked to an account and the ID of the physician the patient is linked to.
A **note** also has a an ID int and another ID int from the patient the note is linked too. The note also includes the question text and the date.
A **dosage** has its own ID int and also stores the ID int of the patient it is linked to and the medicine that it is linked to. It stores a tiny integer for the amount of pills the patient has to take and it stores the date and time the patient has to take the dosage.
A **medicine** has its own ID int and stores the name of the medicine.
The **movie and faq** we are not sure about yet if we will include those in the database or in the app itself. For the movie will always be storing the url to the movie. The benefit of storing the movie URL and faq in the database is that it would be easier to add URL's and questions later on after the app is done. The downside is that it would be slower to retrieve them from the database and that it would cost us more time to include them.

2.1.2 API's

pushPatient: this is the API function that is used to create a new patient account. The API call expects a JSON file containing the information about the patient and a url encoded physician token. The function checks the token and returns an error when the token is not valid. If the token is valid the function will save the patient account in the database. The password is hashed before it is stored in the database.

pushPhysician: this is the API function that is used to create a new physician account. The API call expects a JSON file containing the information about the physician. The function creates a token for the physician and hashes the password before storing all the information in the database.

deletePatient: this is the API function that is used to delete a patient. The function expects a patient ID integer. It will search the database for the patient account with the corresponding ID and delete the patient with the corresponding account.

deletePhysician: this is the API function that is used to delete a physician. The function expects a physician ID integer. It will search the database for the physician account with the corresponding ID and delete the physician with the corresponding account.

modifyPatient: this is the API function that is used to modify a patient. The function expects a JSON file containing the modified information about the patient. It then replaces the changeable data in the account from the patient in the database. Only the name, username and password are changeable.

modifyPhysician: this is the API function that is used to modify a physician. The function expects a JSON file containing the modified information about the physician. It then replaces the changeable data in the account from the physician in the database. Only the name, username and password are changeable.

getDosages: this is the API function that is used to retrieve a dosage from the database. The function expects an ID integer from a patient and the time slot from which you want the dosages. Then it searches the database and returns the dosages within the time slot with their corresponding medicines.

pushDosages: this is the API function that is used to store a dosage in the database. The function expects an ID integer from a patient and a JSON file with the dosage and the corresponding medicines. It then inserts the data from the dosage into the database.

getNotes: this is the API function that is used to retrieve a note from the database. The function expects a ID integer from a patient. It searches the database for the notes linked to the patient and returns them.

addNote: this is the API function that is used to add a note. The function expects an ID integer from a patient and a JSON file containing the note. The note is then inserted into the database.

2.1.3 front end webapp

The front end of the webapp is yet to be created. We do however have an idea how we want this front end to work. The webapp front end is used by the physicians to easily add patients and create treatment plans for them. The webapp will have a log in screen where physician can create an account or log in to their existing account. Then they will reach a screen that show them the token that can be used by a patient to make an account in the android/ios app and where they can search their already existing patients. They can select one of the patients to create a treatment

plan for them and to view how the patient is doing in his treatment plan. Also they can see of the patients send them any notes.

2.2 Front end

The front end uses multiple views to achieve the needs of the customer, allowing us to display a lot of information divided into multiple pieces. The app tries to retrieve data from the web app when the schedule/medication/video view is selected. If this is not possible they will use their current data. The app can also send data to the web app regarding questions of the user to their doctor.

2.3 (Overall communication ??)

The front end app tries to retrieve data when the schedule/medication/video tab is selected from the web app. If this is not possible they will use their current data. The app can also send data to the web app regarding questions of the user to their doctor.

3 Technology Stack

What programming languages are being used? What technologies are being used (Frameworks, libraries, platform, peripherals)? If different components have different technologies, present them individually.

3.1 Webapp

3.2 Android

For android we use android studio as it is a rather simple and capable of performing to our needs for the app. It also provide a lot of basic libraries that are used in our app itself. It also supports virtual devices for android which we use to test it on multiple devices as well as using your own device.

3.3 iOS

4 Team Organization

What teams are there and what are their responsibilities? Are the team responsibilities focused on different components?

4.1 Webapp

Members:
Teodor Ionut Oanca, Giacomo Casoni, Rutger Berghuis, Andrei Scurtu, Sten Sipma, Sytze Tempel.

Responsibilities:
Database, API, Pysician Webapp.

4.2 Android

Members:
Marco Lu, Pieter Jan Eilers, Roel Brandenburg, Robert Riesebos, Niek de Vries.

Responsibilities:
Android Application, User Interface Layout.

4.3 iOS

Members:
Julius van Dijk, Noam Drong, Hidde Folkertsma.

Responsibilities:
iOS Application.

5 Change log

Date	Contributor(s)	Section(s)	Description
22/03	Sten Sipma	All	Created first basic layout for document + added team members & responsibilities
22/03	Marco	All	created basic introduction and filled in most of the android stuff
30/03	Rutger Berghuis	All	Updated the introduction and wrote the part on the webapp