SYMPTOMS
Final Assessement Document

Maciej Szarliński

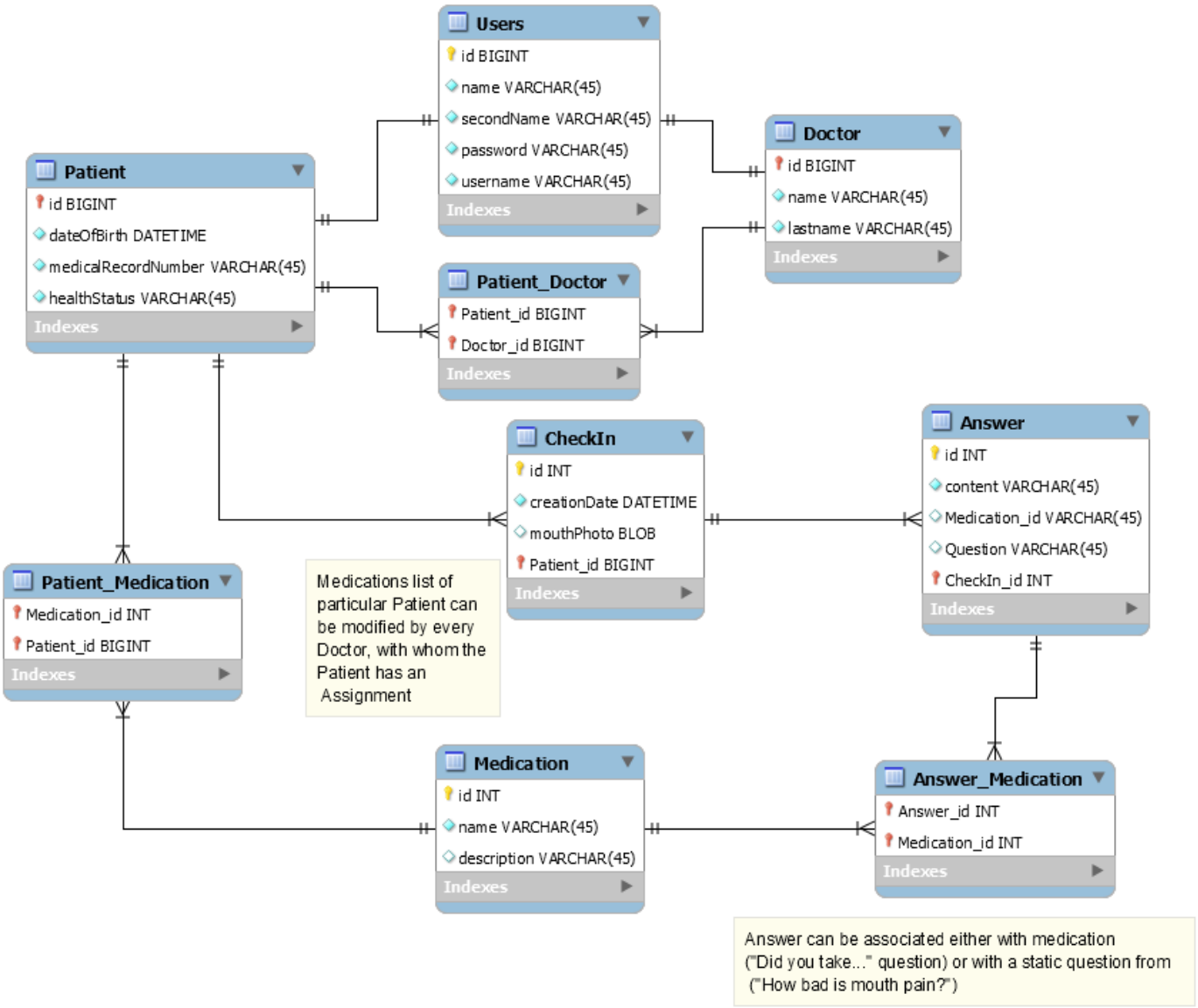
1. Maven project

Symptoms application is fully “mavenized” as it can be easily built and run with a couple of maven commands. System has been divided in a few maven modules:

- symptoms-server - the server side,
- symptoms-rest - common part for server and android clients,
- symptoms-android - contains 2 client android applications.
- third-party - “mavenized” AndroidBootstrap library providing nice look and feel for android applications.

For more information about building and running each part of the system, please watch attached screencasts.

2. Domain model (database)



`Users` table stores credentials for both Patients and Doctors used by a security mechanism - see:

- `UserRepository`,
- `OAuth2SecurityConfiguration`.

Application is configured to run on `MySQL` database. It can be easily changed to use embedded in-memory database - see LocalConfiguration. Application can run also in cloud - it requires to configure `CloudConfiguration` properly. Warning - server url is hardcoded on clients - see `Config` class in `android-client-commons` module.

3. Frameworks and libraries used

I used following frameworks and libraries to get the project done:

Server side

a. Hibernate/JPA - http://hibernate.org/orm
b. Spring Data JPA - http://projects.spring.io/spring-data-jpa/
c. Google Guava - https://github.com/google/guava
d. Spring MVC, Security, OAuth 2.0 - http://projects.spring.io/spring-security-oauth/

Android applications

a. Butter Knife - http://jakewharton.github.io/butterknife/
b. AndroidBootstrap - https://github.com/Bearded-Hen/Android-Bootstrap
c. Spring for Android - http://spring.io/guides/gs/consuming-rest-android/

4. Data security and authentication

All requests are encrypted as server is configured to use SSL. Client can authenticate with token using Spring Security + OAuth 2.0 (see `OAuth2SecurityConfiguration` for details).

5. Data synchronization mechanisms

a) **patient's app** - user can manually adjust notification frequency, thus the requirement is implemented using Android AlarmManager and BroadcastReceiver executing REST call to a server (`GetCheckInTemplateAndNotifyTask`)
b) **doctor's app** - looking for dangerous check-in answers is implemented using Android AlarmManager, alarm is launched every hour (`CheckPatientsHealthStatusAndNotifyTask`)

Google Cloud Messaging API is not used, but it is a good feature to be introduced in further release.

5. Meeting the requirements

Basic requirements

- *App supports multiple users via individual user accounts* - maltiple Patients and Doctors can register to get access to their accounts,
- *App contains at least one user facing function available only to authenticated users* - Patients and Doctors have access to protected resources,
- *App comprises at least 1 instance of each of at least 2 of the following 4 fundamental Android components* - `Activity`, `BroadcastReceiver`,
- *App interacts with at least one remotely-hosted Java Spring-based service* - interacts with multiple RESTful services accessible via `PatientController` and `DoctorController`,
- *App interacts over the network via HTTP* - interacts via HTTP with SSL (port 8443),

- *App allows users to navigate between 3 or more user interface screens at runtime* - there are 15 screens in total,
- *App uses at least one advanced capability or API from the following list (covered in the MoCCA Specialization): multimedia capture, multimedia playback, touch gestures, sensors, animation.*** - Patient client application uses camera,
- *App supports at least one operation that is performed off the UI Thread in one or more background Threads of Thread pool* - every request to the server is being processed asynchronously by dedicated `AsyncTask` - see `*.async` packages in both Patient and Doctor application.

Functional requirements

- *App identifies a Patient as a user with first name, last name, date of birth, a (unique) medical record number, and possibly other identifying information). A patient can login to their account.* - see database schema and `Patient` entity
- *App defines a Reminder as an alarm or notification which can be set to patient-adjustable times (at least four times per day)* - see `NotificationManager` in Patient application
- *A Reminder triggers a Check-In, which is defined by the app as a unit of data associated with a Patient, a date, a time, and that patient's responses to various questions (items 4-8) at that date and time.* - see database schema and `CheckIn` entity
- *Check-In includes the question, "How bad is your mouth pain/sore throat?" to which a patient can respond, "well-controlled," "moderate," or "severe.* - see `HowBadIsMouthPainActivity` and `DoesPainStopFromEatingActivity`
- *Check-In includes the question, "Did you take your pain medication?" to which a Patient can respond "yes" or "no"* - see `DidYouTakeMedicationsActivity`
- *A Check-In for a patient taking more than one type of pain medication includes a separate question for each medication (e.g., "Did you take your Lortab?" followed by "Did you take your OxyContin?"). The patient can respond to these questions with "yes" or "no."* - see `DidYouTakeMedicationsActivity`
- During a Check-In, if a patient indicates he or she has taken a pain medication, the patient will be prompted to enter the time and date he or she took the specified medicine. - see `MedicationListAdapter` in Patient application
- *During a Check-In, the patient is asked "Does your pain stop you from eating/drinking?" To this, the patient can respond, "no," "some," or "I can't eat.* - see `DoesPainStopFromEatingActivity`
- *App defines a Doctor as a different type of user with a unit of data including identifying information (at least first name, last name, and a unique doctor ID) and an associated list of Patients that the doctor can view a list of. A doctor can login.* - see database schema and `Doctor` entity
- *App allows a patient's Doctor to monitor Check-Ins, with data displayed graphically. The data is updated at some appropriate interval (perhaps when a Check-In is completed)* - patients's check-ins data is updated by scheduled task once every hour
- *A doctor can search for a given Patient's Check-In data by the patient's name (an exact text search hosted server-side).* - no, but Patient can search for Doctor using server-side search
- *A doctor can update a list of pain medications associated with a Patient. This data updates the tailored questions regarding pain medications listed above in (6).* - see `PatientMedicationsActivity`

- *A doctor is alerted if a patient experiences 12 of "severe pain," 16 or more hours of "moderate" or "severe pain" or 12 hours of "I can't eat."* - see `CheckInService`
- *A patient's data should only be accessed by his/her doctor(s) over HTTPS* - see `SSLRestTemplate` from `android-client-commons`