Bachelor's Thesis

Geolocalization and routing in complex multi-floor hospital environments

UC Odisee

Department of Engineering Technology - Electronics and Information Technology,

Specialisation Information Technology

Joachim Cardoen

2018

Thanksssssss

Abstract

Contents

Acronyms

API application programming interface. 4, 8–10, 19

EU European Union. 19

PoC proof of concept. 8, 10, 17

SDK software development kit. 8, 9

UI user interface. 17

UML unified modelling language. 9, 19

UX user experience. 17

Chapter 1

Summary

Chapter 2

Context

Chapter 3

Project Specification

3.1 Project Description

The emphasis of the PoC is on developing it in such a way that it should be easy to re-implement the application elsewhere. The PoC is developed in the two current formats for mobile development: iOS and Android. This bachelor's thesis will cover the implementation of the Android architecture. Firstly the existing application is reworked from using the Ionic framework to a native mobile application (Swift for iOS and Kotlin for Android). In addition to this part, geolocalization is implemented in the native mobile app using the MapWize service [15] and the IndoorLocation framework [11], both service provide working software development kit (SDK) for iOS and Android. Finally the application is revised by the team of interns and the developers at IBM and uploaded onto the Apple Store and the Google Play Store.

3.1.1 Technical Design Specs

The communication with the hospital happens with a server provided by IBM and the hospital's application programming interface (API). This means that the mobile device interacts with a intermediary server from IBM which in its turn communicates with the API of the hospital. This model is an example of a highly reusable architecture. If another hospital needs to be attached to the IBM server, only a small 'translator' for the endpoints of the additional hospital's API needs to be created whilst the structure of the IBM server remains the same.

3.1.2 Features

The main features of the project are specified below [10]:

1. Login with hospital provided credentials;
2. Synchronization of appointments with the hospital;
3. Ability to set reminders for an appointment;
4. See the hospital's location (and venues) as well as contact details;
5. Allow geolocalization inside the hospital;
6. Provide feedback after an appointment;
7. Localization in French, English and Dutch;

8. Available on both iOS and Android
9. Distributed in the Apple Store and Google Play Store;

3.1.3 Detailed View of Features

Login and Registration Process

Firstly a patient should create an account to login into the application at a hospital site (mandatory to provide some details). Afterwards the patient will be able to login to the application and has the option to execute the following actions:

- Change userId, password (using the IBM BlueMix API)
- Set an additional authentication method: pincode, face id or fingerprint when launching the app

To register upon arrival at the hospital, two of the following options can be provided: unique cipher code or via scanning a QR-code (displayed at the reception of the hospital).

Appointments

Overview of appointments:

1. View the upcoming or past appointments
2. Go to the notification screen and view notifications
3. Search appointments
4. The first upcoming appointment of the patient

Available user actions possible in the overview:

- View the upcoming or past appointments
- Go to the notification screen and view notifications
- Search appointments

Individual view of an appointment:

1. Link to the specific doctor that will handle the appointment
2. Location
3. Preparatory notes
4. Description
5. Status

The following user actions should be available for the end user:

- Register upon arrival at the specified location
- Request a cancellation of the appointment

- Call the hospital site
- After registration, meaning the user is at the hospital site, the option to show the indoor route to the place of appointment
- Information and link to the hospital site

To create an appointment the user will have to fill in a specific form and is given the option to select a hospital site, service that he/she requires and a specific doctor. Upon submitting the form the hospital will receive a request for an appointment and can notify the user of his or her confirmation.

General Information

Users of the application should be available to view any information available on doctors, services and other important facets of the hospital (for instance: a news feed featuring health tips, recipes and important news on the hospital). This includes possible contact options such are telephone and email possibility. For the doctors, this means the following information needs to be displayed accurately:

- List of all available doctors
- Possibility to search and filter doctors by name, function and service
- Display the languages a doctor is able to converse in
- A timetable that indicates availability based on time and place
- Contact information if available

Other than the doctor's information, the services that the hospital provides should be able to be consulted. This includes the following information:

- List of all available services
- Possibility to search and filter on the hospital site that offers this service and a concrete body part that situates the service.
- Ability to contact the hospital site providing this service, either via email or phone.
- Important times of day: office hours and visiting hours

For the sites this is:

- List of all sites of a specific hospital
- Map view (Google map) that shows a location using a marker
- Ability to contact the hospital site via email or phone.
- Link to the specific website of this site, if it is available.

Geolocalization

The indoor map of the hospital displays general points of interest, such are: restaurant, toilets, elevators, staircases and reception. The route from the reception to the location of the appointment will be displayed when the user requests it inside the detailed view of an appointment (after registration process).

User Settings

The following settings should be displayed inside the application:

- Notification settings
- FAQ
- Feedback on the application
- Information about the application: disclaimer, version, developer(s), privacy agreement
- Level of mobility
- Authentication method
- Profile: email, password, userId, phone
- Doctors of the user (where he or she has or will have an appointment)
- Ability to sign out of the application

Ideas

Ideas for the application that can be implemented in future releases:

1. Symptom checker;
2. Chatbot to schedule and handle appointments
3. Prescriptions for medicines that can be used in apothecaries
4. 3D scan of patient's limb so a doctor can detect possible fractures or other superficial problems
5. Emergency button that will display the nearest emergency exit
6. Data integration of smart devices to determine possible health risks
7. Set reminder for medicine intake
8. Home Assistant integration
9. Details of the medical records of a patient

3.1.4 Technologies to research

Throughout the development of the PoC, several technologies are used, such are: Android SDK, authentication, RoomDB for offline storage, IBM BlueMix API, unified modelling language (UML), dependency injection, MapWize, IndoorLocation and Cisco CMX.

3.2 Development Guidelines

To attain uniformity in the codebase of iOS and Android a 'Development Guidelines' document is written, this document can be found as an appendix.

Chapter 4

Indoor Positioning Systems

4.1 Indoor Positioning

Chapter 5

Cisco CMX

5.1 Cisco CMX

Chapter 6

Integration using MapWize and IndoorLocation

6.1 IMplementation

Chapter 7

Conclusion

Appendix A

Development Guidelines

Bibliography

[1] Android Developer. *Transformations — Android Developers*. 2019. URL: https://developer.android.com/reference/android/arch/lifecycle/Transformations (visited on 03/06/2019).

[2] Android Developers. *Download Android Studio and SDK tools*. 2019. URL: https://developer.android.com/studio (visited on 03/23/2019).

[3] Android Developers. *Guide to app architecture — Android Developers*. 2019. URL: https://developer.android.com/jetpack/docs/guide (visited on 03/08/2019).

[4] Bhavya Karia. *A quick intro to Dependency Injection: what it is, and when to use it*. 2018. URL: https://medium.freecodecamp.org/a-quick-intro-to-dependency-injection-what-it-is-and-when-to-use-it-7578c84fa88f (visited on 03/09/2019).

[5] Daniele Faraglia. *Welcome to Faker's documentation! — Faker 1.0.2 documentation*. 2014. URL: https://faker.readthedocs.io/en/master/ (visited on 03/08/2019).

[6] Eslam Hussein. *Dominate Remote/local data with (RX+Retrofit+ROOM+MVP)*. 2018. URL: https://medium.com/@eslam.hussein/dominate-remote-local-data-with-rx-retrofit-room-mvp-f2b13a0ac27b (visited on 03/17/2019).

[7] EU GDPR Portal: by Trunomi. *Key Changes with the General Data Protection Regulation – EUGDPR*. 2019. URL: https://eugdpr.org/the-regulation/ (visited on 03/23/2019).

[8] Express. *Express - Node.js web application framework*. 2019. URL: https://expressjs.com/ (visited on 03/08/2019).

[9] GoogleDeveloperTraining. *14.1A: Room, LiveData, ViewModel · Advanced Android Development Course- Practicals*. 2018. URL: https://google-developer-training.gitbooks.io/android-developer-advanced-course-practicals/content/unit-6-working-with-architecture-components/lesson-14-room,-livedata,-viewmodel/14-1-a-room-livedata-viewmodel/14-1-a-room-livedata-viewmodel.html (visited on 03/17/2019).

[10] IBM. *Med App Project Specification*. 2018.

[11] IndoorLocation.io. *Indoor Location - The open-source framework for indoor mapping.* 2019. URL: https://www.indoorlocation.io/ (visited on 03/09/2019).

[12] Jaewoong Eum. *Android MVVM Architecture Components using The Movie Database API.* 2018. URL: https://medium.com/@skydoves/android-mvvm-architecture-components-using-the-movie-database-api-8fbab128d7 (visited on 03/06/2019).

[13] James Halpern. *Android – SavedInstanceState Bundle FAQ*. 2012. URL: https://content.pivotal.io/blog/android-savedinstancestate-bundle-faq (visited on 03/06/2019).

[14] MapBox. *Mobile — Mapbox*. 2019. URL: https://www.mapbox.com/mobile/ (visited on 03/23/2019).

[15] MapWize.io. *Indoor mapping & Wayfinding for Smart Buildings*. 2019. URL: https://www.mapwize.io/ (visited on 03/09/2019).

[16] MapWize.io. *Indoor mapping for developers*. 2019. URL: https://www.mapwize.io/developers (visited on 03/09/2019).

[17] Nahidul Hasan. *SOLID Principles-simple and easy explanation – Hacker Noon*. 2018. URL: https://hackernoon.com/solid-principles-simple-and-easy-explanation-f57d86c47a7f (visited on 03/23/2019).

[18] Per-Erik Bergman. *Repository Design Pattern – Per-Erik Bergman – Medium*. 2017. URL: https://medium.com/@pererikbergman/repository-design-pattern-e28c0f3e4a30 (visited on 03/23/2019).

[19] Retrofit. *Retrofit*. URL: https://square.github.io/retrofit/ (visited on 03/24/2019).

[20] Mark Seemann. *Dependency Injection in .NET*. 2011, p. 16.

[21] SwaggerHub. *Basic Authentication — Swagger*. 2019. URL: https://swagger.io/docs/specification/authentication/basic-authentication/ (visited on 04/07/2019).

[22] THE EUROPEAN PARLIAMENT AND THE COUNCIL OF and THE EUROPEAN UNION. *DIRECTIVE 95/46/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL*. Tech. rep. European Parliament and of the Council, 1995. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:31995L0046%7B%5C&%7Dfrom=en.

[23] Theo Jungeblut. *Clean Code II - Dependency Injection*. 2015. URL: https://www.slideshare.net/theojungeblut/clean-code-part-ii-dependency-injection (visited on 03/17/2019).

[24] TutorialsPoint. *SQLite Tutorial*. 2019. URL: https://www.tutorialspoint.com/sqlite/ (visited on 03/06/2019).