OULU UNIVERSITY OF
APPLIED SCIENCES

Heikki Laulajainen

ELECTRONIC GUIDANCE AND MONITORING APPLICATION FOR MAINTENANCE PERSONNEL

ELECTRONIC GUIDANCE AND MONITORING APPLICATION FOR MAINTENANCE PERSONNEL

Heikki Laulajainen
Bachelor’s Thesis
Spring 2016
Information Technology
Oulu University of Applied Sciences

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, Langattomat laitteet

Tekijä: Heikki Laulajainen
Opinnäytetyön nimi: Sähköinen huoltotyön ohjaus- ja valvontatyökalu
Työn ohjaajat: Kari Jyrkkä, Jouni Mäkinen, Tero Kulmala
Työn valmistumislukukausi: Kevät 2016, sivumäärä 23

Opinnäytetyön tilaajana toimi Oy L M Ericsson AB, ja työn tarkoituksena oli tehdä Android-sovellus demojärjestelmään ylläpidon kenttätyöntekijää ja hänen esimiestään helpottavasta järjestelmästä.

Tavoitteena oli tehdä demo, jolla voidaan näyttää asiakkaalle, että tällainen järjestelmä on mahdollista toteuttaa. Tässä opinnäytetyössä pääpaino oli Android sovelluksen puolella.

Työssä käytettävät laitteet ja työkalut olivat Android ohjelmistokehitykseen erityisesti sopivia. Ohjelmistokehitykseen käytettävä paketti oli Googlen toteuttama ja laitteet Googlen suunnittelemia.

Projektin alusta asti oli tiedossa, että työtä ei ehdi saada täysin valmiiksi annetussa ajassa. Tilanteesta, johon tämän opinnäytetyön aikana päästiin, on hyvä jatkaa kehitystä eteenpäin, sillä tarvittavien ominaisuuksien suunnittelu ja koodin perustan luonti saatiin toteutettua.

Asiasanat: Android, Java, ohjelmisto

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Wireless Devices

Author: Heikki Laulajainen
Title of the bachelor’s thesis: Electronic Guidance and Monitoring Application for Maintenance Personnel
Supervisors: Kari Jyrkkä, Jouni Mäkinen, Tero Kulmala
Term and year of completion: Spring 2016, number of pages 23

This Bachelor’s thesis was commissioned by Oy L M Ericsson Ab. The aim of this theses was to create an Android application into a demo system to ease the maintenance workforce and their managers in their daily tasks.

The aim was to create a demo that is capable of showing a customer that it is possible to create this kind of system. In this thesis, the main focus was on the Android application.

Tools and devices, which were used, were suitable for Android software development. The development environment and devices were designed by Google.

From the beginning of this project it was known that the work will not be fully completed within the given time. The project reached the point from where it will be good to proceed with the development in future because research, planning and base of code have been done.

Keywords: Android, Java, software

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

RECA	Region Northern Europe and Central Asia
POI	Point of interest
API	Application programming interface
OS	Operating system
ASCII	American Standard Code for Information Interchange
PDF417	Portable Data File (4 bars and 17 units per pattern)
HTTP	Hypertext Transfer Protocol
IDE	Integrated development environment
SDK	Software development kit
NDK	Native development kit
APK	Android application package
AVD	Android virtual device

1 INTRODUCTION

This Bachelor’s thesis work was commissioned by RECA (Region Northern Europe and Central Asia) unit at Oy L M Ericsson Ab. Ericsson was founded in 1876 and it used to be a small 2-man business in Stockholm. It has since expanded their business to international levels. (1) Nowadays in 2017, Ericsson can be considered to exist all around the globe because Ericsson does business in every continent and in almost every country. Ericsson used to be mainly known for their cellular phones in the past but there is a lot more than that in their product and service catalogue. Most likely everyone in the world who uses networked mobile devices has used Ericsson technology without being aware of it.

The aim of this thesis was to create a mobile application for field engineer personnel to help their daily tasks during the maintenance on the site. Field engineer personnel/maintenance personnel will be referred as a user in this document after introduction. The idea was to eliminate the need of laptops and paper manuals while working on site. Then application brings an easy access to this information with their handheld mobile devices. On the background the application communicates with a backend server, which holds the database and logics for queries made by the application.

The application was one part of the project and starts almost from zero and only preliminary studies were made. It was a good starting point for the participants because they could be part of the planning process straight from the beginning and it gave a more throughout vision of the project. A field engineer is not the only person that benefits from this project. On the management side it is much easier to follow which tasks are done.

2 PROJECT TEAM BRIEFLY

The solution consists from a backend server and an Android application and six participants were involved in this project. The first unit had two members, the second unit had only one and the rest were working with both as mentors. The first team was responsible for the implementation of backend server and its manager web interface. The role of the backend server is to handle the communication and forwarding of the fetched data to the Android application. All the data is not stored into the backend server and the server needs to know where to find it. The manager web interface also runs on this backend server and it is the view that interacts with the manager. The web frontend shows the issues to the manager and they can then be assigned to correct people as tasks.

The second unit was responsible for developing the Android application layout design and programming it. This is the part that this thesis focuses on and it will be explained more thorough.

Communication was an urgent part of the project to overcome its troubles. The whole team could not gather around locally and meetings had to be kept as conference call meetings. At the beginning of the project meetings were held on the first working day of the week with the whole team. Short follow up meetings with backend and application units were also kept every day and this was not a big issue because of the size of the actual working team. Later on, follow up meetings were kept only every second day because there was really no need to keep them every day.

3 REQUIRED FEATURES

This application benefits the users and their managers. Managers can assign tasks into the user profile at the web interface and the user can review those tasks assigned to their profile from the application. (Figure 1.) The application throws a notification to the user when new tasks are assigned to them.

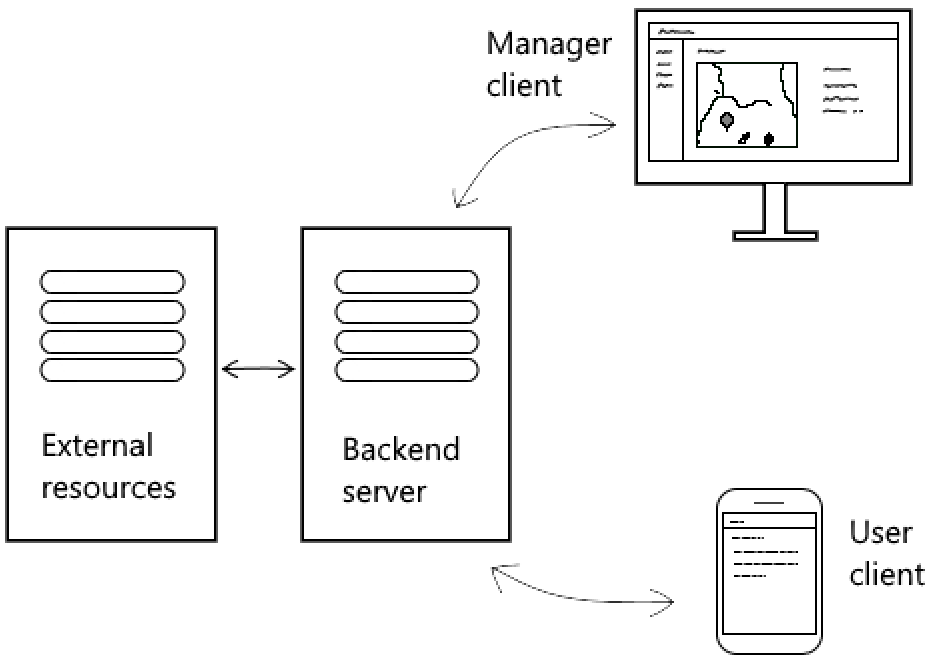

FIGURE 1. Architecture of the solution

3.1 Task description

The task description contains information on the destination site and work tasks. This contains all the data needed to access a site that needs maintenance or a regular check-up.

The information that the task description contains is as follows:

- An address or coordinates of destination.
- A key pick up location.
- A to-do work list.

- Possible special tools and spare parts that are needed.
- A notification if there is something that needs attention, for example if you need a car with a high ground clearance is needed.

3.2 Map feature

The task description contains a POI (point of interest) object. The application uses Google Maps API (Application programming interface) for the navigation and to show map locations. The benefit of this functionality is that the user does not need to search for the location from the Google Maps application manually.

3.3 User location

The user’s location is sent to the backend server during working times. This helps the manager to see which user is nearest to the site and they can then assign the nearest suitable personnel to go to the destination. The user sets the working hours into the application settings and this function is active only during working hours. The user also has an option to disable this feature.

3.4 File uploading

Uploading files to the server is done via the application. The user can capture a photo with their camera or select a file from a gallery and then send it to the server. This is a handy tool when there is a need to create documentations or reports after the work has been done.

The backend server needs to accept the image and document files. The file size or extension type is not restricted. File types are described in the header of an HTTP message and the backend server decides whether it accepts them or not.

3.5 File downloading

The application can download files from the backend server into the storage of the used device. These files can be images or user and installation guides.

3.6 User profile and authentication

Managers assign tasks to user profiles. Each maintenance employee in the field has their own personal account and by authentication they can access tasks assigned to them.

Each profile contains information on the user’s name, certificates and skills. This enables managers to assign tasks to a suitable user.

3.7 Barcode reading

This is one of the main functions of this application. It helps the user to find a correct and up-to-date installation and maintenance guide for the device. The user reads the barcode by using the camera in their mobile device. After a successful barcode reading, it converts the message to a more readable format and the application delivers the code to the backend server, which replies by sending a correct instruction file as a return.

4 APPLICATION CONCEPT IN DEPTH

This application is designed to work under Android devices with an API level 17 or higher. In the future, the roadmap of the application might contain developing a support for another mobile platform, such as Apple iOS, but for now the application is developed to work natively on Android mobile devices.

4.1 Android in general and why API level 17

Android is an open source mobile OS (Operating system), which is currently developed and maintained by Google. It is based on the Linux kernel, which makes Android OS versatile. It can be deployed to multiple different platforms, which usually are based on x86 or ARM. The devices that can currently be used with Android OS are smartphones, televisions, wearables, cars, notebooks, gaming consoles and cameras.

Android is well known across the world is the most sold mobile OS and the usage of it in smartphones is the most common. It has the largest set of different applications and setups. Operators and smartphone manufacturers can customize the OS as they wish within the limits set by Google.

Developing for Android is easy and there is a lot of help available in different developer and programming communities. Google provides and maintains Android Studio for Android developing purposes and it contains almost everything that is needed for basic and advanced development.

Android has API levels and versions to specify which functions are available or which updates are applied. When a larger update is applied, then the API level number and version number increase and if it is a minor update, then only the version number increases. Higher API levels support or have compatibility for lower API levels.

The API level 17 which is used in this project, is enough to cover the needed functionality and most of the used devices on the market. Before starting an Android project, it is a good thing to plan which activities and functions the application should have and which of them are necessary. Then the earliest suitable version should be chosen. When choosing the earliest possible compatible level, the application will support older devices that users may possibly have. If the level is set too high it also makes the catalogue of supported devices more restricted. If the API level is set too low and the functionality the developer wants to use is not supported, then the developer is going to have some challenges bringing that functionality into the application.

4.2 Barcode reading

There are multiple applications made for barcode reading, which can be found in Google Play Store. These scanner programs are able to read the barcode or QR code using the camera of the phone. Then they convert the barcode to a human readable format. The application in this project needs to be able to read barcodes in Code 128 and PDF417 formats because the target products, which have barcodes in them, use both formats and contain different information.

4.2.1 Code 128

Code 128 is a common and simple encoding method for converting an ASCII (American Standard Code for Information Interchange) character set to a linear one dimensional barcode. (2) (Figure 2.)

android-is-cool

FIGURE 2. “android-is-cool” message encoded to Code 128 (3)

Code 128 barcode consists of two quiet zones in the beginning and in the end of barcode, a start character, encoded data, a check character and a stop character.

Quiet zones are before the start character and after the stop character and their width is 10 times of the minimum width of a module (bar). The start character indicates when the message begins and the stop character indicates when the message ends. Code 128 contains three set types, A, B and C, and a type value is set in the start character. (2)

Between the start and stop character there is encoded data and the check character, also known as a checksum. Encoded data contains the message that is written in the barcode. The checksum is a sum of start code value and values multiplied with their positions in encoded message. The checksum value (table 1) is modulo 103 reminder of the sum of values. (2)

The current published revision of Code 128 bar code specification is ISO/IEC 15417:2007.

4.2.2 PDF417

PDF417 (Portable Data File) is a stacked two-dimensional linear barcode format and it consists of 4 bars and spaces per pattern. (Figure 3.) Each pattern is 17 units long. The barcode is 3 to 90 rows long and each row has a quiet zone, a start/stop pattern, a left and right row indicator and 1-30 data code words.

FIGURE 3 "android-is-cool" message in PDF417 format (4)

PDF417 can theoretically hold up to 1,850 characters, 2,710 digits or 1,108 bytes (5). The current published revision of the PDF417 standard specification is ISO/IEC 15438:2015

4.3 File transfers

Files are transferred by using simple HTTP (Hypertext Transfer Protocol) GET and POST methods. Later this may be changed with a more secure implementation but this will suit the needs of this application prototype.

HTTP is a way for a host and a client to communicate between each other. In this case, the backend server is the host and the Android device with the application is the client. Basically, HTTP transactions consist of requests and responses.

The client is the one who sends the requests to a server and the server sends a response to the client. The message contains a request method, a URI and a protocol version.

4.4 Security

Security needs to be taken into consideration to protect the data. As this project is a demo, security was not the first concern. Specialists will later ensure that data protection will meet the standards of the company.

5 MAKING OF ANDROID APPLICATION

5.1 Design planning

The planning of application was executed with a design first method. Since the application is supposed to help the user, thus user experience is a top priority in the layout design.

Two simple designs were made and from them the team had to choose one. Both of them followed Google's Material Design with their own colour palettes and images and commonly used navigation styles. Material Design keeps the application simple and it is visually simple and attractive.

First design was based on the TabLayout navigation with fixed tabs. It is a simple navigation style where the user swipes their way through a set of views one view at a time.

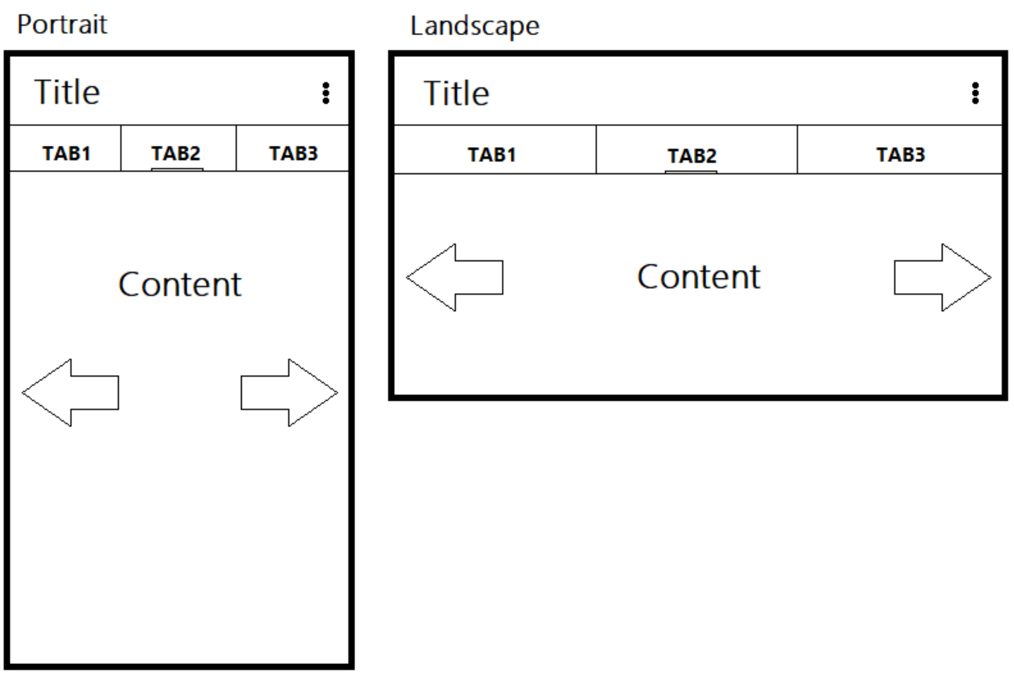

FIGURE 4. Draft of TabLayout navigation style

What is good in this layout is that when the used features and settings can be designed to be used sequentially and or the user interface only needs a few views. The bad side is that when the application has multiple views it can be painful to search through. Even though the user can just tap on the wanted tab and it will switch its view, the tab carousel will get bloated easily when there are lots of tabs.

The second design was based on Drawer Navigation. The navigation menu is in a separate view and it can be swiped as an overlay on top of the current view.

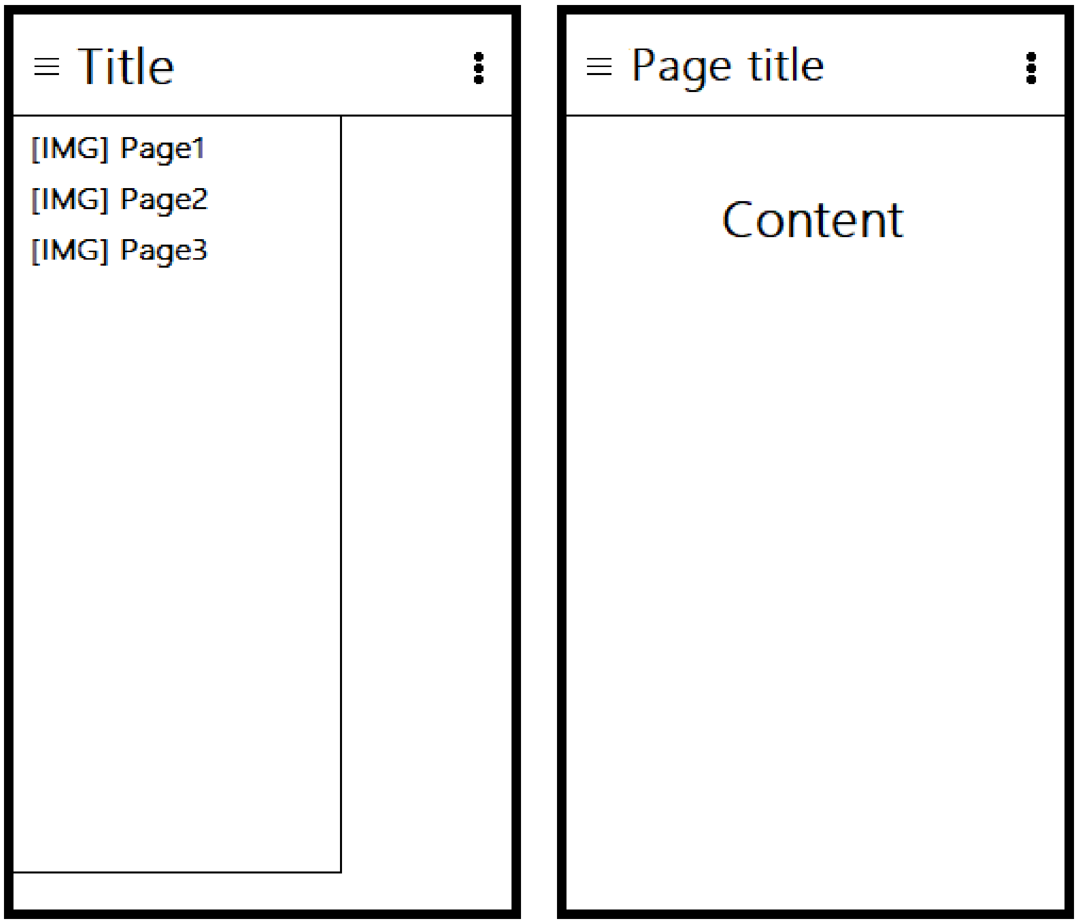

FIGURE 5. Draft of drawer navigation style

With this design, there is more height for the content compared to the TabLayout and navigating through views is easy. It is also possible to make submenus to have different content per view and to insert buttons and switches into the view of drawer to serve a function.

The team chose the latter style because it is easier to move around with different views compared to the TabLayout navigation style. Then next step is to consider how to assemble the information in a view to keep it readable and clean. There must be enough space between every object and lines of text. At the same time creating a long wall of information, which needs to be scrolled all the time, should be avoided. This can be solved by using descriptive icons for functions and by only showing the relevant information for the user.

5.2 External libraries

As everything does not have to start from zero, few assets were used. Google Maps API helped with the locating feature. Because barcode reading had already been done in another project, there was in-house code available, which was also used in this application.

To use Google Maps API in the application, the developer needs to have a key that is provided by Google. Each key can be used on only one application and it is linked to a Google account. The key is put into google_maps_api.xml file in project files.

5.3 Preparations and the basics

The language used was Java, which is a preferred native language for Android development. The chosen development environment for coding was Android Studio IDE (integrated development environment). Android Studio has SDK (software development kit), NDK (native development kit), a layout editor and testing and platform tools bundled which help and save time when configuring the environment is configured.

An App Manifest “AndroidManifest.xml” is an essential file that defines the application. (6) It must be located in a root directory of the application to work. It holds the following information:

- an application and package name
- permissions, an SDK version, compatibilities and configuration
- activities and providers

A resources directory “res/” is used to keep configurations, images and values. It can contain bitmap and vector images, animations, string values, colour values and layout XML files. (6)

A few good to know things are listed below:

- Every application must have an App Manifest.
- Landscape and portrait layouts have their own layout resource.
- Strings and images should always be used from resources.
- Some controls, such as lists, need adapter classes to create their item collection.
- Each view needs to have an activity class and a layout resource. One activity can have multiple layouts.

5.4 Programming process

As the planning process was UI (user interface) first, so was the programming part too. The basis of the application, which holds navigation and frames, was made by creating the layout first and programming activities after it. There are a few good to know things. All the fields that contained a text or an image were configured to show their content through resources. In this way language localisations can be created and fixing misspells is easy. Used images were found from Ericsson’s own resources and colouring was done by the Ericsson’s guide instructing how the colour theming should be.

The basic activity used FrameLayouts. Each view was showed in a container in the main activity and it was changed through the navigation menu. The first feature to be made was a locating service and it was done as a service. Services in Android applications are components that perform long-running operations in the background of the application. (6) The service polled the data through a locating provider of Android OS and delivered this information to the backend server.

With the barcode reading the first challenge was to bring the camera functionality into the application. It was solved in the way that the application launches an action intent of an external camera application and the user chooses which one to be used. Intents can be used, for example, to launch a camera application or to dial to a phone number. The application received output data of a captured image and converted and saved it into the storage memory as a bitmap file. Soon after this functionality, it was time to show a proof of concept to a customer and developing a file transferring began. The development of barcode reading was not completely fulfilled and it ended up as a lacking functionality.

The file transfer feature communicates with the backend server. It is able to send and receive files. The user can choose a file from their storage and then send it to the backend server via an HTTP POST action. At this point, this was restricted to images and documents. The application can receive files by sending a GET command with a string to the server, and then receive the file as a reply and save it into a storage.

Since there was a few features ready, it was time to create a settings activity to control them. From the settings, the user can disable locating or setting up the active working hours, and the application automatically saves it into a configuration file to control the locating service. The IP address of the server was configurable by the user at this point for easy access.

6 TESTING OF APPLICATION

No automated test features were implemented to this application since it is meant as a demo and not for the production. The program was tested and debugged locally on a physical device and no AVDs (Android virtual device) were used. The reason to test the application on a device was to save time because launching the AVD takes more time than pushing and installing the APK (Android application package) straight on to the device. Another benefit of testing the application locally is that the developer can monitor the performance of the application and fix possible slowdowns.

For testing the Motorola Moto 2nd gen, the Nexus 5X and Nexus 7 (2013) tablet with an LTE connectivity were used. Nexus devices are designed by Google therefore they are a perfect companion to be used as a debugging devices, although they are not very common among target users and the devices that target users may have can use different ways of application handling, for example the Samsung file management. Using devices with different configurations helped to indicate issues in the layout scaling and performance. The application performed well and stable overall on these devices without having any critical issues in it. Views were made to be light so that the navigation was fast and loading times were not noticeable.

The application required an Internet connection through a public network and it brought some challenges between the communication with an Android client and the backend server. The server was running in an unexposed internal network. Accessing it through a public network had some restrictions and there was no approved way of connecting the mobile device to an internal network. After this issue had been solved, realistic testing was possible.

One live test with a possible customer gave more input to what features the application should have. It also revealed some good points about the functionality. This input was processed and applied to the application and it received a positive feedback.

7 CONCLUSION

This project gave a lot of experience of how to work efficiently in a small group. It also showed that communication is a key element. Distances and own protocols of the company brought some challenges but everything necessary was solved. Every day we improved our team work, programming skills and the ways of working.

What comes to all the planned features, we are satisfied with everything that was done in given time. The application has now a specification, a locating service, an image capturing and a processing capability without a working barcode reading, an implemented basic Google Maps activity without the POI functionality, a file handling and the base code. All the features were tested when a new functionality was implemented to ensure that they will work together.

For the rest of the features that were not implemented, we had made a pretty good research on how to implement them to the application and server. It was already known that this whole project most likely will not get fully done but the most important thing was that now there is a base for it.

REFERENCES

1. Wikipedia, Ericsson. Date of retrieval 1.8.2016.
   https://en.wikipedia.org/wiki/Ericsson

2. IDAutomation, Code-128 Bar Code FAQ & Tutorial. Date of retrieval 1.8.2016.
   http://www.idautomation.com/barcode-faq/code-128/

3. RACO Industries, RACO Resources Code 128 barcode generator.
   https://racoindustries.com/barcodegenerator/1d/code-128/

4. RACO Industries, RACO Resource PDF417 barcode generator.
   http://racoindustries.com/barcodegenerator/2d/pdf417/

5. TEC-IT, PDF417 (2D Barcode). Date of retrieval 1.8.2016
   https://www.tec-it.com/en/support/knowbase/symbologies/pdf417/Default.aspx

6. Android Developers, 2017. Date of retrieval 25.3.2017.
   https://developer.android.com/studio/intro/index.html