Rok akademicki 2014/2015

Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych
Instytut Informatyki

PRACA DYPLOMOWA INŻYNIERSKA

Martyna Wiącek

Mobile application for international exchange students at the Warsaw University of Technology

Opiekun pracy
dr inż. Robert Bembenik

Ocena: .....

.....

Podpis Przewodniczącego
Komisji Egzaminu Dyplomowego

Kierunek: Informatyka

Specjalność: Inżynieria Systemów Informatycznych

Data urodzenia: 1992.09.08

Data rozpoczęcia studiów: 2012.02.20

Życiorys

Urodziłam się 8 września 1992 r. Po ukończeniu Szkoły Podstawowej nr 143 w Warszawie, podjęłam naukę w Gimnazjum nr 33 im. Stefana Batorego w Warszawie w klasie o profilu dwujęzycznym, którą zakończyłam w 2009 roku. W 2011 roku ukończyłam II Liceum Ogólnokształcące im. Stefana Batorego w Warszawie w klasie o profilu dwujęzycznym z rozszerzonym programem matematyki i geografii, zdając maturę oraz egzamin certyfikujący z języka angielskiego Certificate of Proficiency, wydany przez British Council. W lutym 2012 roku rozpoczęłam studia na kierunku Informatyka na Wydziale Elektroniki i Technik Informacyjnych na Politechnice Warszawskiej. W 2014 roku odbyłam dwumiesięczne praktyki w londyńskiej siedzibie banku Goldman Sachs na stanowisku Summer Analyst. Podczas moich studiów byłam wolontariuszem Erasmus Student Network, pomagając studentom, uczestniczącym w programie wymian międzynarodowych na Politechnice Warszawskiej oraz Uniwersytecie Warszawskim. W 2012 roku pracowałam jako wolontariusz, wspierając organizację konkursu w programowaniu grupowym ACM ICPC na Uniwersytecie Warszawskim.

.....
Podpis studenta

EGZAMIN DYPLOMOWY

Złożył egzamin dyplomowy w dniu ..... 20__ r

z wynikiem .....

Ogólny wynik studiów: .....

Dodatkowe wnioski i uwagi Komisji: .....

.....

.....

SUMMARY

Summary in English.

The thesis presents the process of creating a mobile application for Android system, basing on a specific example, which is a system for foreign students (called ESN PW), taking part in the international student exchange at the Warsaw University of Technology. The idea of the project derives from the market gap as well as the author's personal experience. The main scope of the project is facilitating students' stay in Warsaw and providing them with relevant information about the city of Warsaw. The process of creating the application assumes existence of all the stages of application development, starting with comparison of the existing solutions, market evaluation, the domain analysis and architecture schema and finishing with implementation and testing. Whilst describing the implementation, the major attention was drawn to the best programming practices and modular division to help the application be easily modifiable and upgradeable in the future.

Keywords: mobile application, Android system, international exchange, Erasmus Student Network, Warsaw University of Technology

---

Mobilna aplikacja dla studentów wymian międzynarodowych studiujących na Politechnice Warszawskiej

Streszczenie pracy w języku polskim.

Głównym tematem pracy inżynierskiej jest przedstawienie sposobu tworzenia aplikacji mobilnej dla systemu Android, na przykładzie aplikacji ESN PW dla zagranicznych studentów, uczestniczących w programie wymian międzynarodowych na Politechnice Warszawskiej. Idea tego projektu wynika z potrzeby rynku oraz z osobistych doświadczeń autorki. Celem projektu jest ułatwienie studentom pobytu w Warszawie przez zapewnienie dostępu do jak największej ilości informacji dotyczących studiów na Politechnice Warszawskiej a także samego miasta. Proces tworzenia aplikacji zakłada istnienie wszystkich etapów, od porównania istniejących rozwiązań, badania rynku, analizy dziedziny i projektu architektury systemu po implementację i testowanie. Podczas opisywania implementacji, szczególną uwagę zwrócono na wykorzystanie najlepszych praktyk tworzenia aplikacji mobilnej oraz na modułowy podział kodu tak, aby mógł być on w późniejszym czasie łatwo modyfikowalny.

Słowa kluczowe: aplikacja mobilna, system Android, wymiana międzynarodowa, ErasmusStudent Network, Politechnika Warszawska

Table of Contents

1. Introduction

In our modern, incredibly fast moving world, it is almost unthinkable not to take advantage of great opportunities, given to us thanks to the continuing growth of technology as a whole. It is extremely hard to imagine our lives without computers, Internet and constant access to all sorts of information.

Being a student of Engineering is exciting and stimulating, particularly because along with hundreds of young people we are taught how to create and introduce sharp ideas to wider public using new technologies. Unfortunately, learning about it can at times mean being caught up in reading thousands of code lines in order to perform the task given at the highest level. Because we are still learning programmers it is not always effortless and simple. That being said, it is possible to lose track of the reality, spending more time in front of the computer rather than socializing and discovering the world.

I decided I did not want to become one of those programmers.

I have always loved meeting new people, learning about new cultures and being open to all the opportunities we can get in the lifetime. Speaking two foreign languages fluently, I decided that in my free time I wanted to do something which both could help me expand my abilities even more and hopefully simultaneously help other people. Therefore, I became a member of ESN Community that is Erasmus Student Network.

ESN is Europe-wide student organization, hugely supporting the international exchange programs “Erasmus” and “Erasmus Plus”. It has its branches all over Europe, particularly in the most popular destinations for student exchanges. It composes of around 13,500 people in 452 local sections in 37 countries and the main goal of the community is to make exchange students feel welcome in a foreign country, provide them with appropriate help from the community and relevant information, improve the social and organizational integration of students, contribute to students’ mobility and finally encourage students to study abroad, learning about themselves but also about other cultures. Around 180,000 students participate annually in this type of exchanges [4].

In Poland there are 33 sections in various Polish cities trying to facilitate the stay for all the exchange students. Only in Warsaw are there 11 sections, each belonging to a different university and taking care of about 10,000 exchange students [5]. Altogether, they create a network which enables Polish students to collaborate easily in the preparation for various events such as orientation week or trips to famous Polish sightseeing centres. What is really noteworthy is the fact that Polish ESN is one of the biggest networks of this kind in Europe, after Turkey and Italy [12].

ESN values volunteering and active citizenship therefore it is possible to get involved and help foreign students really easily. I decided to become a mentor - a person who is attributed to an exchange student. The mentors have to take care of the students, including answering their questions about the studies, leading them around in the city of their stay and helping them in resolving problems, should any occur. Having lived in Warsaw since my birth, I have never thought about the problems that can be encountered here, mostly because of lacking knowledge in the local language. How to get to the university? Where is the closest shop? How to buy transportation tickets? What documents do I need to complete my exchange? These are the questions the mentor is supposed to know the answer for.

After several conversations with my Erasmus friends I realized what ESN was missing – there was no mobile application, which could help students in getting around, providing

them with basic information helpful throughout their exchange. Therefore, I decided it could be a valuable idea to create such an application, not necessarily eliminating the mentors from the picture, but being an addition to their role, and what is more important, helping the Erasmus students as well, when the mentors may be temporarily unavailable.

The other motivation for my work was the fact, that none of Polish universities has developed a dedicated application to solve this problem, therefore my application could fill the existing gap in Polish market and hopefully present decent solutions, compared to other widely available applications of this type.

1.1 The needs

It is extremely difficult to name unambiguously all the functionalities that a satisfactory Erasmus mobile application should have. To my mind, the ones which should be included are the official information vital for completing the studies properly, ways of getting in touch with corresponding Erasmus mentor, all the data about recent activities and upcoming events, likely connected with Facebook event page, as currently this is the main place of Erasmus students' communication.

Undoubtedly, this system has to have the necessary means of security such as logging option, to prevent random people from having access to the system. I believe, it is vital for the application to be connected also with the Google Maps, thus being possible for Erasmus students and mentors to mark on the map the most interesting places to visit.

The system should be easy to use and very intuitive, because the application ought to be the very first one obligatorily downloaded by the Erasmus students. Not everyone has a decent knowledge of new technologies so this factor also has to be taken into consideration.

1.2 The main goal

The main goal of this project id, as I mentioned before, to create an easily accessible mobile application called ESN PW to facilitate the work of mentor and provide the Erasmus students with additional and more elaborate information regarding the rules of the exchange and all other information, relevant to the stay.

The application is designed for the devices run on Android OS. It consists of:

- A mobile application for Erasmus students
- A server with the data base (containing all of the information)

The application gives the students access to the information, stored in the database as well as will try to contain all the important news and details such as news feeds from social media in just one application. The main idea is instead of having several applications installed in the mobile phone, the users will only need one application, which links or forwards them to appropriate websites or other applications.

1.3 Thesis structure

The thesis consists of nine chapters. The first presents the introduction to the whole project itself, containing basic information about the application and the idea, from which it derives.

The second chapter shows the comparison of other applications, created at different universities, mentioning their functionalities along with their advantages and disadvantages. This makes it easier to understand the market the application is supposed to be introduced to.

The third chapter focuses on the functional and non-functional requirements. It contains description of all the functionalities contained in my work.

The fourth chapter presents technology stack used for the development of my project. It also mentions the environment in which my application was developed.

The fifth chapter presents details of the implementation of all the elements of the system. It describes non-functional as well as the functional requirements, examples of the code of the most interesting features in the application, both regarding the backend itself and the user interface.

The sixth chapter is focused on the description of testing with the explanation of the results.

The seventh chapter presents the outline of my application in the final version, describing various screens, picturing the solutions.

The eighth chapter is a summary of the thesis that describes the positive and negative outcomes of the work and outlines whether there are any chances of improving or extending the work in the future or introducing the application to ESN.

2. Overview of mobile applications for international exchange students

In recent years the Erasmus program has become increasingly popular. At some universities various mobile applications were created in order to facilitate the stay for the international student. However, I believe that some of the applications lack useful functionalities, while the others may have been born from a great idea but not finished in a satisfying way.

In this chapter I present a quick overview of some of the existing applications and explain, in my opinion, which functionalities are valuable and what could be improved. The existing applications I present in this chapter can be divided into two general groups. The first group is exemplary applications, which introduces valid solutions worth imitation. Applications in this group include ESN Sea Battle, My ESN Events, ESN Messina, ESN Cordoba and ESN Roma ASE. The second group – unsatisfactory applications, gathers those which introduce solutions that should be avoided – as in Erasmus Phone, ESN Aveiro, ESN Lund and SPQE Erasmus Roma.

2.1 Exemplary applications

2.1.1 ESN Sea Battle

The ESN Sea Battle is a student cruise between Stockholm and Tallinn. It organized twice a year, usually annually in November and April and gathers over 2100 students from ESN Denmark, ESN Estonia, ESN Latvia, ESN Lithuania, ESN Norway and ESN Sweden. The event is arranged on the M/S Baltic Queen by the National Board of ESN Sweden.

The official application, created for this specific event, consists of two parts, focusing the user's attention on the program of the event (*see Fig. 3*) and the menu (*see Fig. 1*). The menu consists of: the descriptions of destinations, Instagram competition section with the photos from the recent editions, Frequently Asked Questions and a list of partners. Moreover, Destinations menu contains a bunch of information (*see Fig. 2*) regarding the biggest ports this event will take place in, including the maps and FAQ about the cities and transportation.

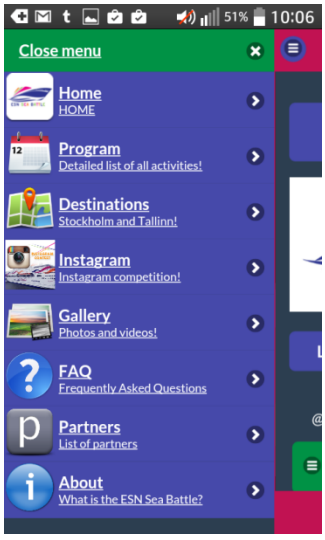

Figure 3 ESN Sea Battle Menu

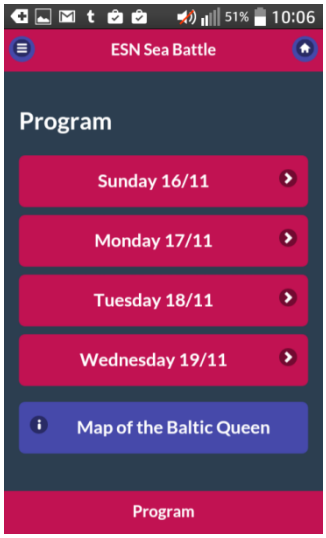

Figure 2 ESN Sea Battle information

Figure 1 Sea Battle program

The modules in the application are partially built-in, as the module for Instagram competition or the modules regarding the detailed program of the cruise. On the other hand, there are also links, forwarding the user to the external applications or websites such as Google Maps for viewing the maps of the cities or Flickr or Vimeo, for browsing the collection of the photos and videos [23].

2.1.2 Erasmus ESN Events

Erasmus ESN Events mobile application imitates a news feed regarding some of the biggest European universities among ESN sections. The first screen (*see Fig. 5*) shows the list of all possible sections to follow and there are around 100+ of them. After selecting the interesting ones, we get to another screen, where all the events are listed, attributed to previously selected sections. There exists also a small menu which enables user to refresh the feed page and select a few options regarding the app.

On the main screen, after clicking on an interesting event, the application takes us to another screen, showing a detailed description (*see Fig. 4*) and a link to event's page on Facebook or to the official website of a given university. There's also a possibility of adding a new event to the personalized calendar on user's mobile phone (*see Fig. 6*).

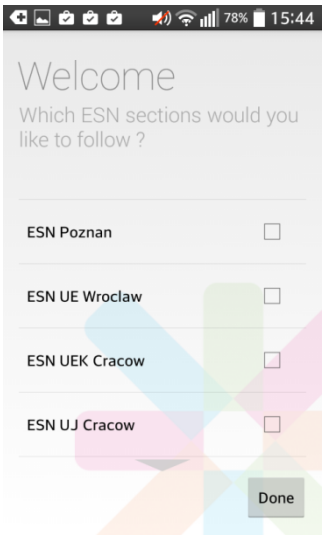

Figure 6 ESN Events map screen

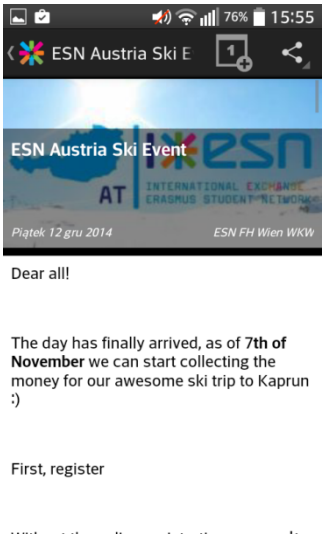

Figure 5 ESN Events event detail

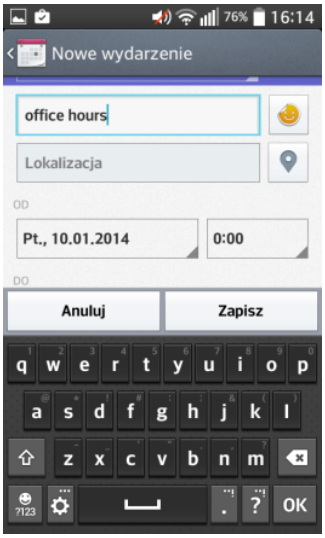

Figure 4 ESN Events adding event to calendar

The application is very simple and seems quite useful, in particular the possibility of adding the event to our own calendar. The main drawback is that on some of devices the option menu is not working properly and instead of being shown on the screen, the application throws an Exception, which, what is more interesting, from my observation does not occur on tablets [18].

2.1.3 ESN Messina

ESN Messina is a basic application with all the most important information regarding Erasmus. There can be found news about the university (*see Fig. 8*), maps explaining transportation and the location of the most important university sites in the city. There is

also a screen with Administrative Contacts, Frequently Asked Questions (*see Fig. 7*) and services.

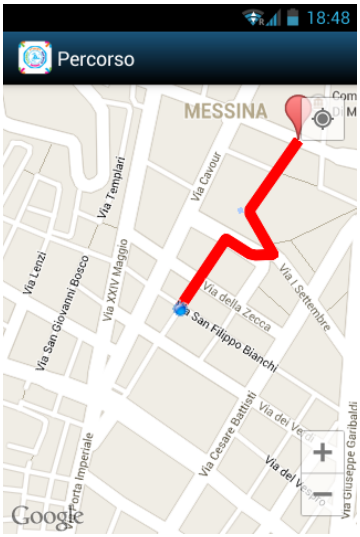

Figure 8 ESN Messina route to campus

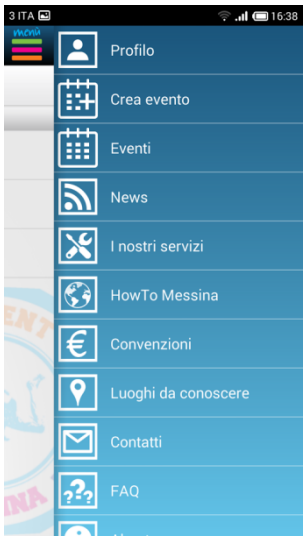

Figure 9 ESN Messina menu

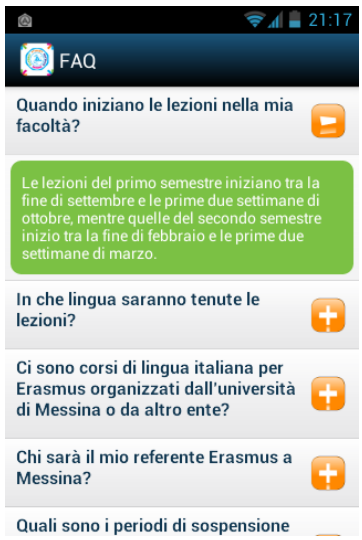

Figure 7 ESN Messina Q&A screen

An internal contact network is found within the application. After authentication, the users are able to add friends to their list and chat with them. There is information about the events and a functionality, which enables users to create their own events.

Overall, the application is attractive and very intuitive, as can be seen on Figure 9. However, I think it may lack user's interaction as for example adding their interesting places to the list [21].

2.1.4 ESN Cordoba

The application of the University of Cordoba is again a simple system for the international students, helping them in gaining the relevant information about the local environment.

Menu is divided into a few sections (*see Fig. 11*) such as Parties, Trips, Events, Partners and Office (*e.g. Fig. 12*). The majority of them seems to contain a simple screen with static information, while the screens describing the Events are probably a news feed from another website, what can be seen on Figure 10. On the top of the screen there are also many icons, connecting users with ESN profiles on the major social media websites.

The application requires an account in the system therefore I was not able to login. Comparing to other previously mentioned applications it seems to have fairly secure authorization system [26].

Figure 10 ESN Cordoba main menu

Figure 12 ESN Cordoba campus map

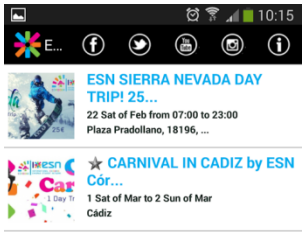

Figure 11 ESN Cordoba events

2.1.5 ESN Roma ASE

This application fairly resembles the ESN Cordoba, although the GUI does not impress as much, as it does in the case of the first one.

Users are required to register and sign in (*e.g. Fig. 15*) in order to be able to take advantage of all the functionalities. On the top of the main screen there is visible a simple menu: News, Guest list, Info and Contact Us (*see Fig. 13*). After clicking on one of them the user gains access to more elaborate information.

The information tab has a lot of options to select from, including Rome Erasmus Guide, information about the offers for the students, maps and information about cyclical events taking place (*e.g. Fig. 14*) [22].

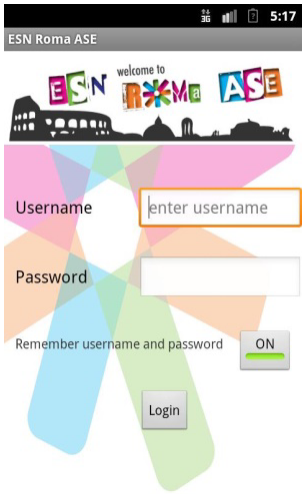

Figure 15 ESN Roma login page

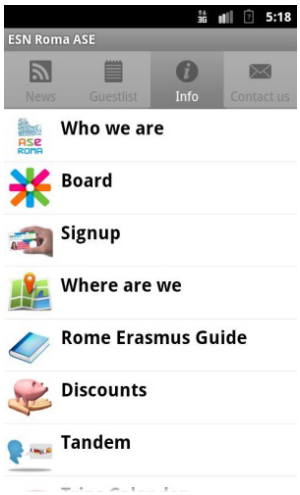

Figure 14 ESN Roma main menu

Figure 13 ESN Roma news screen

2.2 Unsatisfactory applications

2.2.1 Erasmus Phone

This application is based mainly on the concept of simplifying the action of calling interlocutors from abroad using their mobile or landline phone number not necessarily knowing the country code. The application imitates the standard phone call application embedded in the Android OS (*as seen on Fig. 17*) with a slight difference – there is a helpful list box with the country codes for all of the European countries, therefore if a user is not entirely convinced which prefix should be used before the number, this quick fix can be very handy (see Fig. 18). After inserting the full number the application forwards the number to the standard Phone application on Android and attempts to establish a connection.

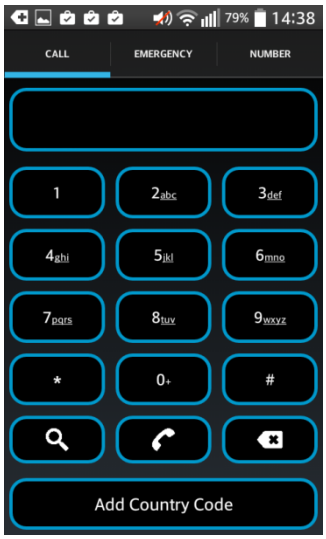

Figure 17 Erasmus Phone main menu

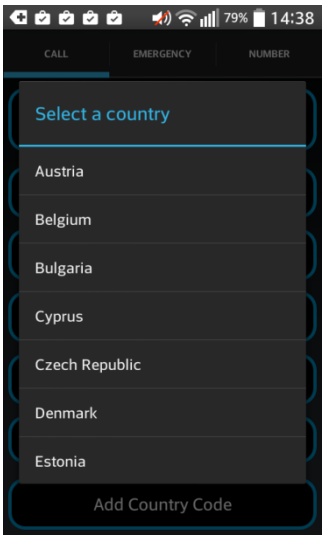

Figure 18 Erasmus Phone country selection

Figure 16 Erasmus Phone emergency data

Erasmus Phone has got two additional screens. One of them is a practical tool for all of the foreigners, as it contains the emergency numbers for all the European countries (see Fig. 16). The user has to simply type the name of the country into a search box and on the screen there will be shown the emergency numbers available to call in a chosen country.

Another featured screen could be helpful only in theory. In case of sharing a mobile number with another person, the user is asked to type the caller's name and caller's destination number and after that their message is forwarded to a standard Message application to be sent manually by the user.

As overall, I can see the point of first two functionalities because they spare the user's time in searching for the critical information, particularly when they are in a hazardous situation, but I fail to see the usability of the third functionality, which, essentially, only generates a simple message that could be typed just as quickly as using the build-in messaging system [7].

2.2.2 ESN Aveiro

This application is the main project of the ESN at the University of Aveiro. In theory it looks very simple, it is divided into three sections: News, Events and Partners so should provide an exchange student with the information regarding news at the university, events taking place and discounts, which are available for students (*see Fig. 20*).

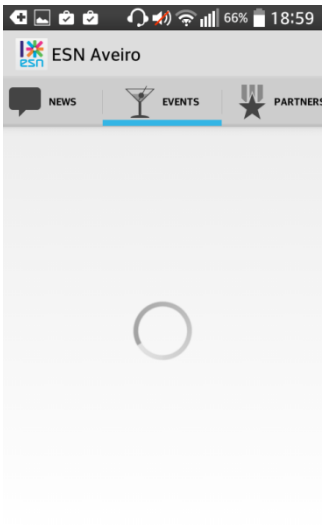

Figure 19 ESN Aviero connecting to the server

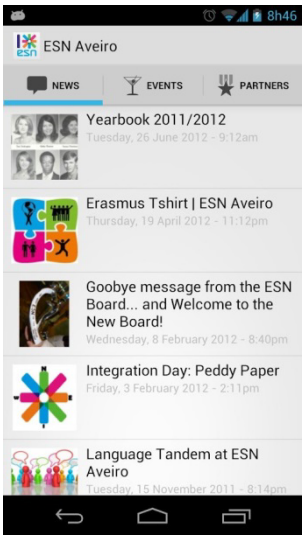

Figure 20 How ESN Aviero should look like

Unfortunately, I was unable to see the application functioning properly, because it could not connect to the server (*see Fig. 19*). There was shown a loading page and regrettably I was not able to proceed any further [19].

2.2.3 ESN Lund

This application is attributed to Swedish university in Lund. Although my expectations and hopes were very high, after seeing its snapshots on the Google Play Store, alas it turned out to be faulty in many ways.

The first screen is a simple page with a text field to insert a password in order to be able to use the application properly. However, I noticed that there was an elementary programming error included as well. After clicking on the "back" button in my mobile phone I expected application to close. Instead, I saw the main menu screen for a split second. After a few trials it was possible to open every page in the menu, and although the GUI of the application itself is really impressive, there exist basic errors in the authorization and authentication of the users.

The menu is very intuitive (*see Fig. 23*), enabling users to read about the news in a news feed, there are links to Facebook news feed and Twitter news feed as well (*see Fig. 22*), which can be useful. Another positive aspect of the application is the calendar with the events and ESN Office information page. Discount and Social Wall options in the menu seemed to be a great idea, however their execution leaves much to be desired.

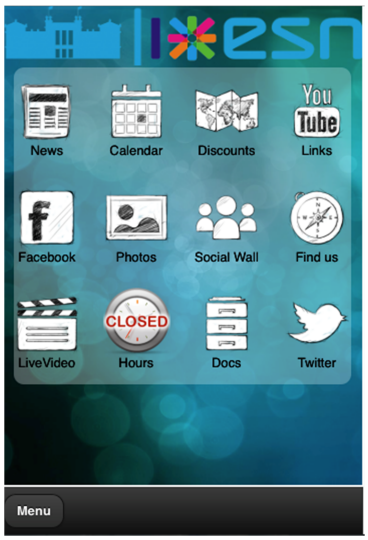

Figure 23 ESN Lund main menu

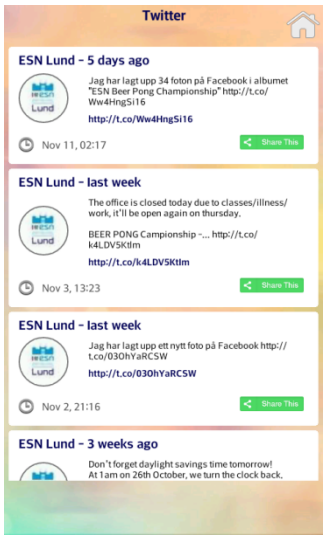

Figure 22 ESN Lund news feed

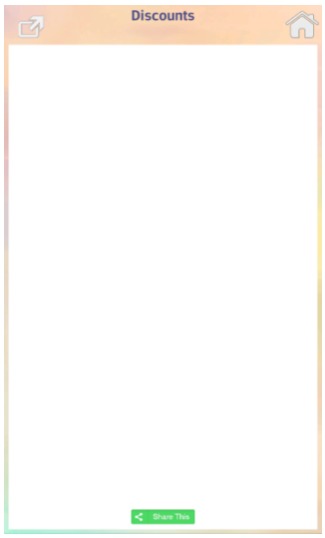

Figure 21 ESN Lund offers map

Discounts should be a separate screen with a Google map showing all the places offering discounts, unfortunately instead of this the screen changed its colour into white and it was possible to be forwarded to Google Maps only by clicking a small button, located on the upper left side of the screen (*see Fig. 21*). Social Wall is a basic chat, resembling Facebook wall, but it certainly lacks functionalities. Although the user is able to "like" someone's post, they are unable to "dislike" them as well and delete their own posts [20].

2.2.4 SPQE Erasmus Roma

This application is, as the one mentioned before, attributed to ESN Rome and potentially helpful for all the students experiencing their Erasmus in Rome. Unfortunately, the application is not "bug free".

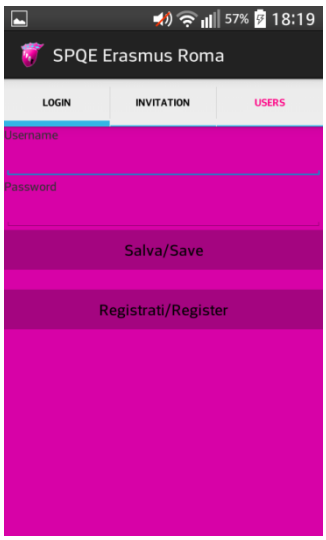

Figure 26 SPQE login screen

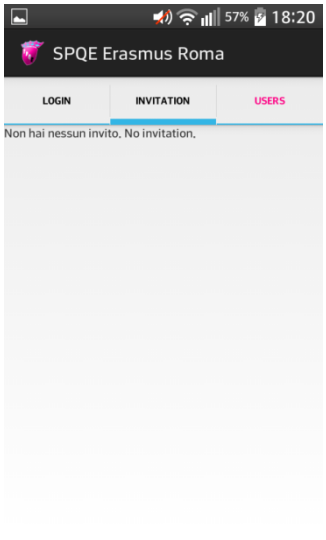

Figure 25 SPQE not working screen

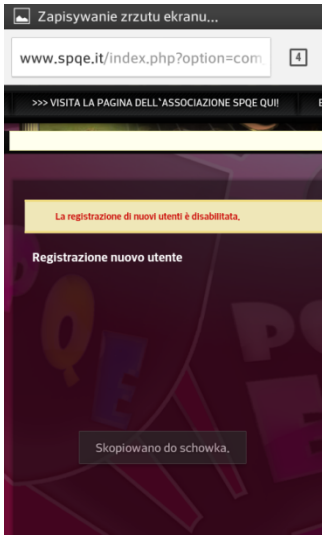

Figure 24 SPQE website with no registration option

Firstly, there is not a possibility of logging in or registering (*see Fig. 25 and Fig. 26*), yet the authorization is required to see the content of the application. Opening the application and clicking the registration link forwards the user to the website of the community, where there is only short information in Italian, notifying the user about the inability to login or register.

The other links in the menu, which supposedly should forward the user to the events or to the list of the other users [1], do not work (*see Fig. 24*), obviously because of the lack of prior authorization. Instead of blocking these options in the menu or create a nicely looking error screen, the application responds with black screen and error message written in black on the output.

2.3 Summary

To summarize all the results of the research there can be found both exemplary applications, which could be inspiring as well as completely unsatisfactory ones, which should be undoubtedly avoided.

Starting with positive examples, I found authentication to be certainly unomissible part of my application. Firstly, it enables only a closed group of people to get access to the application, which is reasonable, as it can contain the information not that necessary or relevant for non-Erasmus students. Secondly, authentication helps unequivocally identify the users, which can help in personalizing data shown in the application or introduce such modules as communication between users.

The other relevant functionality is certainly the use of Google maps and location detection. There are a lot of applications using this technology, some of them simply displaying pre-calculated points on the map and some offering more complex solutions as calculating the shortest route to a certain point. I feel that this functionality is one of the most important, as foreign students very often have difficulties in moving around the city.

The other interesting functionality is giving the users information about the events organized by ESN, which is vital for exchange students in the process of getting to know new people and creating networks. Almost all the already existing applications are equipped with this function in a various ways, mostly via a news feed coming from Facebook or the official ESN website. Another good idea is to include offers and discounts, available to students thanks to partnership programs. I see it as potentially helpful as it is extremely difficult for the exchange students to get to know about the concessions by themselves, usually not knowing the language of the foreign country they are staying in.

The next interesting module in the application is the contact with the ESN Office, the mentors and national emergency units. It certainly could help the exchange students to get in touch with ESN authorities or emergency units quicker, in case of emergency or any urgent situation.

Another interesting feature used in a lot of application is Frequently Asked Questions section. I think this particular idea is very inspiring. It contains a lot of valuable information in one place, useful to the students mainly at the beginning of their stay. Moreover, this form of presentation is more appealing to the young people.

Focusing on negative aspects, the existing applications show the lack of basic securities solutions. There are either systems, which have a very simple authorization, not protecting the data at all, or systems having the option for authentication but no option for user

registration, which makes the application barely usable. Another problem is the non-functioning connection to the server with data. It may be the case of the wrong established connection on the side of the application or non-responding server – in both cases the application is not functioning properly.

Further negative examples are functionalities, which do not make sense from a practical point of view, therefore those overriding an already existing solutions embedded in phone's system. There are also examples of poorly programmed functionalities which potentially could be elaborated but in its current state are too simple. Another issue is that some of the applications work only on a selected version of Android system. In general, the applications should not be intended for one specific type of device and constrained to it, as in the case of My ESN Event application, working properly only on a tablet.

To sum up, after browsing through the already existing applications, I was able to notice both their positive and negative aspects and decide which errors should be avoided in my project and which functionalities were worth reusing and improving in some way.

3. Functional and non-functional requirements

In order to develop an application, which would be most useful to the users, it is essential to take into consideration their needs and expectations. Regarding this particular problem, it can be noticed that the vital part of the application should be its simplicity and intuitive navigation. We have to remember, that although the users will be taking part in the exchange at technological university, not all of them have to be fluent in new technology solutions. The goal is to develop an informative application, available to use out of the box without any prior knowledge regarding mobile applications.

Additionally, it should be noted that the application will be used by a limited and finite group of users. Each semester there is a different set of students visiting the university, besides the students, who stay there for a yearlong exchange. It means that the system will be less likely to fail because of the overload of the traffic and it will be aimed at a selected group of people, around 100 – 300 people.

The application requires authentication of the users and is a closed system basing on invitation, so there is no possibility of registering a new account unless the user obtains the password from the system administrator. In this way, it can be ensured that the system functionalities will not be used by unauthorized users.

3.1 Basic requirements and customer's needs research

In order to provide customers with some basic requirements, based on my knowledge gained during my work as a mentor, I created such functionalities, which would give them access to the most important data after their arrival to Warsaw. I incorporated information regarding the Warsaw University of Technology, as well as the city itself.

After inventing some solutions and having an overall idea of how the application should be developed, I surveyed the customers themselves. I asked 70 of my friends who I met whilst working as a mentor in last two years to complete a form, created by Google Docs platform. The survey consisted of 10 closed questions, asking customers to give their opinion on the basic functionalities proposed. Additionally, there was an open question as well, asking potential customers to share their ideas regarding the application and request any changes or additions.

The general response to the survey was satisfactory. It was completed by 36 respondents out of mentioned 70. The split of votes regarding the proposed solutions was mainly in favour of the functionalities. However, some of the functionalities were more popular and some were thoroughly criticised in the comments section. The results of the survey can be seen on the graph below. As it can be noticed, the majority of proposed solutions got an enthusiastic acceptance, as there was only one functionality out of ten in which the difference between the positive and negative response was minimal. The mentioned functionality, therefore the built-in chat, was not proceeded with, as there are many already existing and widely used applications with similar functionalities such as Facebook, Skype, Twitter or Whatsapp. During the process of finding the requirements it is important to take into consideration already existing solutions, as it is unnecessary to create ones that are simply redundant. There can also be seen a keen interest in social-centered requirements, featured in the application. It means that the students, visiting our university find this information more crucial and less available. The more detailed results of the survey can be found on Figure 27.

A far more important part of the survey was surely the open suggestions, which were a remarkable way to share opinions and ideas with the customers. There were many of them, which I incorporated into my application, such as a division into social-based and university-based functionalities feature in the same menu, so as to make the navigation much simpler, however still drawing the line between those two groups. Another interesting suggestion touched the problem of the availability of basic transportation, accommodation and study information, vital after arrival and possible to obtain only in Polish. The next recommendation was to ensure the customers would be provided with a map of campus, with all the offices potentially helpful for an exchange student to find help. Focusing on social-based requirements, surveyed suggested as well a planner, useful to create notifications regarding meetings and trips, which could be shared among a chosen group of friends, who are using the application.

All the featured ideas were extremely helpful and vital in understanding a foreign student's point of view, which differs greatly from the point of view of the organizers who are mainly Polish students and have not experienced the same issues as students coming from abroad.

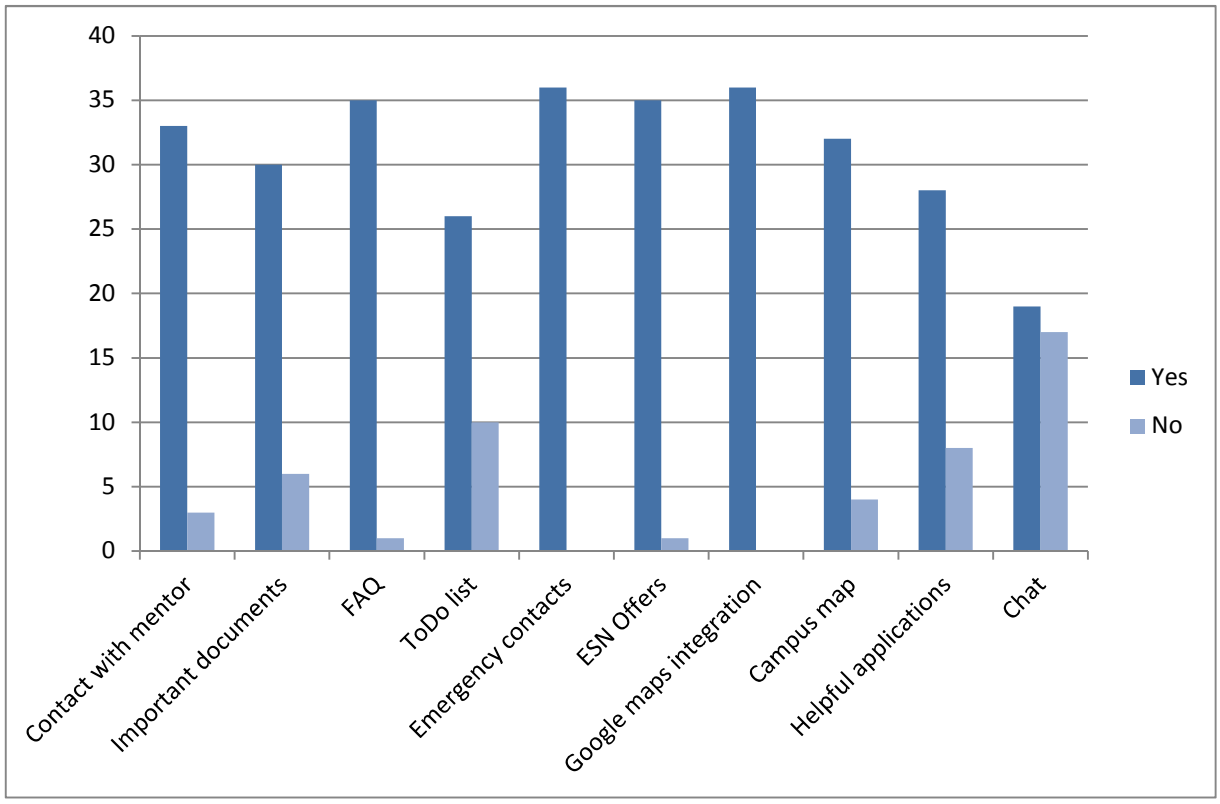

Figure 27 Results of my survey

3.2 Functional requirements

The ESN PW application has only two actors, the **registered logged user** and **non-registered user**. The non-registered user has access to one functionality which is logging in to the application and becoming a registered logged user. Registered logged user has the access to

all of the functionalities in the application. Based on the research there have been created functional requirements, introduced in the application. The list of the functional requirements is contained in Table 1.

Code	Use Case	Primary Actor	Brief	Flow
ESN-BUDDY-1	User's mentor contact list	Registered logged user	Presenting a clickable contact list of the user's mentor	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. On the screen information regarding user's mentor is shown – their first name, last name and a list of contacts to the most popular applications such as Facebook, Skype etc.
ESN-BUDDY-2	Contacting user's mentor	Registered logged user	Application performs an action according to chosen data to inject to external application	**Basic flow:** 1. User clicks the appropriate position in the emergency list 2. The application transfers given contact to the application, operating certain request and protocol i.e. protocol *skype:* will be transferred to Skype application 3. The application starts request, trying to match the application which would be the best to process the query 4. If the chosen application is Facebook, the application processes another query to Facebook API in order to get mentor's identification number **Alternative flow 1:** 1. – 2. The same as in the basic flow 3. If mentor has not provided the data regarding a particular application, on the list is shown information „Not available" 4. After clicking „Not available" the application shows the user a notification, informing of the inability to process the request **Alternative flow 2:** 1. – 3. The same as in the basic flow 4. If application is not installed, the notification is shown on the screen, informing user of the necessity of installing a particular application to complete the request
ESN-CAMPUS-MAP-1	Showing campus map of Warsaw University of Technology	Registered logged user	Application shows a Google Map of Warsaw University of Technology campus	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. A screen is opened containing Google map with all the faculties marked on it. In yellow is marked the user's main faculty, in red are marked exchange offices, and in blue – all other faculties **Alternative flow:** 1. The same as in the basic flow 2. If there is no connection to the Internet, user is informed about it with a notification on the screen

ESN-CAMPUS-MAP-2	Showing details for Campus Map	Registered logged user	Application shows details for each marker, featured on the Campus Map	**Basic flow:** 1. User clicks the marker on the map 2. A new prompt info window is opened a with detail information about the building, including dean's and student's offices located there **Alternative flow:** 1. User clicks anywhere on the map, excluding the markers 2. Recently opened prompt window is closed
ESN-INFO-DOCS-PAGE-1	Showing information regarding transportation	Registered logged user	Application shows information regarding important documents to be taken for the exchange	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. The data from a text file is loaded and presented on the screen
ESN-EMERG-1	Emergency numbers list	Registered logged user	Presenting a clickable emergency number list to call	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. On the screen there is shown list of contacts available to choose from
ESN-EMERG-2	Calling emergency number	Registered logged user	Application performs a call to the emergency number chosen by user from a list	**Basic flow:** 1. User clicks the appropriate position in the emergency list 2. The application transfers given number to the process, operating Android phone call machine 3. The phone call process starts automatically, calling a given number
ESN-EVENTS-1	Showing events to attend	Registered logged user	Application shows a clickable list of events to attend by the user	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. A query to the server is made 3. Server asks a few group profiles on Facebook via Facebook Api, collecting the data 4. Server returns data as JSON The data is presented on the screen **Alternative flow:** 1. – 2. The same as in the basic flow 3. If the response is negative, user is informed about it on the screen via notification
ESN-EVENTS-2	Redirecting user to event	Registered logged user	Application redirects user to Facebook application, showing the proper event	**Basic flow:** 1. User clicks the appropriate position in the event's list 2. The application redirects a query to Facebook application, showing the details for a certain event, basing on its id number **Alternative flow:** 1. The same as in the basic flow 2. If Facebook application is not installed, the event page is shown in the WWW browser on mobile device
ESN-FACULTY-1	Showing official ESN PW website	Registered logged user	Application opens an embedded WWW browser, showing official ESN PW website	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. Basing on user's faculty name stored in the Shared Preferences

				upon user's login, there is shown a list of clickable links to main website of the faculty, virtual dean's office and e-mail server 3. The list of possible websites to visit is embedded in the application's code 4. On clicking a chosen link, user is forwarded to the WWW browser application on the mobile device in order to complete request **Alternative flow:** 1. The same as in the basic flow 2. In case of any errors, the corresponding notification is shown on the screen
ESN-FAQ-1	Showing FAQ list	Registered logged user	Presenting on the screen Frequently Asked Questions list	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. The application performs a query to PHP server 3. As response it gets a data in JSON format 4. The application builds objects basing on obtained JSON and creates a list, shown on screen 5. Upon selecting an item from a list, the list row is expanded and there is shown response to the question, posted in a list's row **Alternative flow:** 1. – 2. The same as in the basic flow 3. If the response is negative, on the screen there is shown a notification with the error message
ESN-FUNMAP-1	Showing Fun Map	Registered logged user	Application shows Google map with marked places, recommended by users to other users	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. A new screen is opened with map, showing all of the places added to the database by users **Alternative flow:** 1. The same as in the basic flow 2. If there is no internet connection, user I notified about this by a prompt on the screen
ESN-FUNMAP-2	Filtering Fun Map	Registered logged user	Application filters marked places on Google Map by categories which they belong to	**Basic flow:** 1. User chooses a category from a roll-down list 2. The view is refreshed, showing only the places, belonging to a chosen category
ESN-FUNMAP-3	Add new place	Registered logged user	Application enables user to add a new place to the database	**Basic flow:** 1. User clicks "Your places" button 2. There is shown a notification on the screen, explaining how to add a new place to the database 3. User clicks "OK" button 4. On the map there are shown all the markers added by user, they are coloured yellow 5. After long clicking on map there is created a new draggable marker, coloured red

				6. If user drags marker it is possible to drop it in the other place of the map, changing its coordinates 7. On clicking the marker a new dialog is opened and it is possible to change name, description and category of a place 8. After changing data user clicks "Save" 9. The data is added to database and on the screen appears a notification, informing user about the success of this operation 10. The markers on map are refreshed and updated **Alternative flow 1:** 1. – 6. The same as in the basic flow 7. User i.e. chooses a category from the map 8. Points on the map are updated 9. The newly added marker on the map is no longer available, because it was not added to the database **Alternative flow 2:** 1. – 7. The same as in the basic flow 8. User withdraws the changes and clicks "Cancel" 9. The point is not added to the database 10. The view is refreshed
ESN-FUNMAP-4	Edit existing place	Registered logged user	Application enables user to edit a place	**Basic flow:** 1. User clicks "Your places" button 2. There is shown a notification on the screen, explaining how to add a new place to the database 3. User clicks "OK" button 4. On the map there are shown all the markers added by user, they are coloured yellow 5. On clicking the marker a new dialog is opened and the name, description and category of a place fields are filled with the data pulled from the database 6. It is possible to change the data 7. After changing data user clicks "Save" 8. The data is updated in the database and on the screen appears a notification, informing user about the success of this operation 9. The markers on map are refreshed and updated **Alternative flow:** 1. – 6. The same as in the basic flow 7. User withdraws the changes and clicks "Cancel" 8. The point is not added to the database 9. The view is refreshed
ESN-FUNMAP-5	Deleting point from Fun Map	Registered logged user	Application deletes a point from the database, previously added by user	**Basic flow:** 1. User clicks "Your places" button 2. There is shown a notification on the screen, explaining how to add a new place to the database 3. User clicks "OK" button

				4. On the map there are shown all the markers added by user, they are coloured yellow 5. On clicking the marker a new dialog is opened and the name, description and category of a place fields are filled with the data pulled from the database 6. It is possible to change the data 7. After changing data user clicks "Delete" 8. The confirmation prompt is shown 9. User clicks "OK" button 10. The data is updated in the database and the point is deleted from it 11. The markers on map are refreshed and updated **Alternative flow:** 1. – 9. The same as in the basic flow 10. User withdraws the changes and clicks "Cancel" 11. The view is refreshed
ESN-GET-IN-TOUCH-1	Showing Get-In-Touch list	Registered logged user	Application shows list of all the users, who agreed to be listed in the application	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. A new screen is opened with a list of all of the users, currently using the application **Alternative flow:** 1. The same as in the basic flow 2. If there is no internet connection, there is shown a notification on the screen, informing user about the errors
ESN-GET-IN-TOUCH-2	Filtering Get-In-Touch list	Registered logged user	Application filters users list by name, last name or faculty	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. User inserts name, last name or faculty, they are looking for 3. The list is updated in the view
ESN-GET-IN-TOUCH-3	Showing Get-In-Touch details	Registered logged user	Application shows details of the user, selected from Get-In-Touch list	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. User selects person from Get-In-Touch list 3. A new screen is opened with selected user's profile information, including a picture and contact information in form of a clickable list 4. Upon clicking on list user can forward the flow to an external application, which will be able to process this query **Alternative flow:** 1. – 3. Same as in the basic flow 4. If selected user has not provided us with some data, the data will be marked as "Unavailable" 5. After clicking on this list row user will be prompted with the notification
ESN-HELP-1	Contacting ESN office	Registered logged	User contact ESN	**Basic flow:**

		user	office member via e-mail	1. User clicks appropriate position in the menu 2. On the shown screen user clicks email icon 3. User is redirected to mail application by an Intent
ESN-INFO-RENT-1	Showing information regarding accommodation	Registered logged user	Application shows information regarding accommodation and ways of renting an apartments and links to the websites, useful for the user in the research	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. The data from a text file is loaded and presented on the screen 3. A special adapter is loaded, which enables user to swipe between two tabs, one of which is only a plain text and the other – a list of clickable links
ESN-INFO-RENT-2	Showing details of clickable link pages	Registered logged user	Application redirects user to a website, described by the link, contained in the links list in one of the tabs	**Basic flow:** 1. User clicks the appropriate position in the main menu to open Rent A House info position 2. User swiped left in order to be able to see the content of the second tab 3. In the tab there is presented a list of clickable links, leading to websites 4. On clicking the link, the action is redirected to Android web browser, showing the content of the link
ESN-INFO-TRANSP-1	Showing information regarding transportation	Registered logged user	Application shows information regarding transportation and links to the websites, useful for the user	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. The data from a text file is loaded and presented on the screen 3. A special adapter is loaded, which enables user to swipe between two tabs, one of which is only a plain text and the other – a list of clickable links
ESN-INFO-TRANSP-2	Showing details of clickable link pages	Registered logged user	Application redirects user to a website, described by the link, contained in the links list in one of the tabs	**Basic flow:** 1. User clicks the appropriate position in the main menu to open Transportation info 2. User swiped left in order to be able to see the content of the second tab 3. In the tab there is presented a list of clickable links, leading to websites 4. On clicking the link, the action is redirected to Android web browser, showing the content of the link
ESN-LOGIN-1	Logging user into application	Non registered user	Users log into their profile and get access to public and private data.	**Basic flow:** 1. Non registered user enters the username and password and clicks a button, which logs them into application 2. Application performs a call via HTTP to PHP server and queries for the verification of the data 3. Server responds positively 4. Application performs a call to get the vital data to be stored in the internal database of the mobile device 5. The data is returned from server in

				JSON format 6. The returned data is written in a database created on the mobile device 7. The session parameters such as UserId are stored in the Shared Preferences on the mobile device 8. The application creates a notification on the screen informing user about the success of this operation 9. The main screen of the application is shown, featuring links to social media websites where ESN PW can be found **Alternative flow:** 1. – 2. The same as in the basic flow 3. Server responds negatively 4. Application creates a notification on the screen informing user about the failure of the authorization 5. The logging screen is refreshed
ESN-LOGOUT-1	Logging out of the application	Registered logged user	User is logged off the application and the application is closed	**Basic flow:** 1. User clicks the button located in the bar in the upper part of the screen 2. User's session is destroyed and swiped off the phone memory 3. The data stored in the mobile device's database regarding ToDo functionality is sent to the external server 4. External server accepts data and stores it in the external database
ESN-OFFERS-1	Showing offers list	Registered logged user	Presenting on the screen Offers list	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. The application performs a query to PHP server 3. As response it gets a data in JSON format 4. The application builds objects basing on obtained JSON and creates a list, shown on screen 5. Upton selecting an item from list, the list row is expanded and there are shown details of the offer **Alternative flow:** 1. – 2. The same as in the basic flow 3. If the response is negative, on the screen there is shown a notification with the error message
ESN-OFFICE-1	ESN Office hours	Registered logged user	Showing data of ESN office hours	**Basic flow:** 1. User clicks appropriate position in menu 2. There is shown a screen with information data
ESN-PICTURE-1	Displaying profile picture	Registered logged user	Application screen for displaying user's profile picture	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. There is opened a new screen with user's profile picture and three button, one used for uploading a picture from mobile device's memory, one for taking new picture and finally one for uploading it on

				server **Alternative flow:** 1. The same as in the basic flow 2. If there is no previously uploaded picture, there is shown a default picture
ESN-PICTURE-2	Uploading picture from mobile device's memory	Registered logged user	User uploads picture from mobile device's memory	**Basic flow:** 1. User clicks "Choose picture" button on the Change picture screen 2. There is shown a gallery of pictures, currently stored on the phone 3. Upon choosing one of them the picture is shown on the main activity screen
ESN-PICTURE-3	Taking a picture using mobile device's camera	Registered logged user	User uploads picture taken by mobile device's camera	**Basic flow:** 1. User clicks "Take picture" button on the Change picture screen 2. The request is redirected to camera application, letting user use standard mobile device's camera activity to take a photo 3. After the picture is taken user has to click "Ok" button to accept the choice 4. On accepting the choice the picture is shown on the main activity screen **Alternative flow:** 1. – 2. The same as in the basic flow 3. If the "Cancel" button is clicked the device returns to the main application screen
ESN-PICTURE-4	Uploading picture on server	Registered logged user	Uploading picture, chosen by user, to the server	**Basic flow:** 1. User clicks the "Upload" button 2. Previously selected picture is changed to bit array 3. The bit array is uploaded to the server 4. The field in the external database is changed so it can point to the file's path 5. The file is updated in the mobile application view **Alternative flow:** 1. The same as in the basic flow 2. If there was no picture uploaded or chosen, user is informed about this error by a notification on the screen
ESN-PWONLINE-1	Accessing online the Warsaw University of Technology information	Registered logged user	Accessing online the Warsaw University of Technology information	**Basic flow:** 1. User clicks appropriate position in menu 2. There is shown screen with social media application icons 3. Upon clicking on an icon user gets redirected to a certain profile via chosen application if it is installed, else via web browser
ESN-SETTINGS-1	Showing the settings list	Registered logged user	Application shows a list of settings available to change	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. User is presented with a list of the options to choose and edit from

ESN-SETTINGS-2	Change password	Registered logged user	Application opens an embedded WWW browser, showing official ESN PW website	**Basic flow:** 1. User clicks the appropriate item in Settings list 2. There is opened a new dialog on the screen which shows three empty fields to fill in 3. The user has to insert their old password, the new password and the confirmation of new password 4. User clicks "OK" button 5. The information is send to server 6. Server receives the data and sends response to the mobile device 7. On the screen there is shown notification informing user about the success of the operation **Alternative flow 1:** 1. – 3. The same as in the basic flow 4. User click "Cancel" button and the operation is withdrawn **Alternative flow 2:** 1. – 2. The same as in the basic flow 3. If the old password does not match the one stored on server or the new inserted passwords are different, user is notified by the message on the screen
ESN-SETTINGS-3	Change personal information or visibility	Registered logged user	Application opens a dialog, which enables user to change information about their account or visibility of their profile data.	**Basic flow:** 1. User clicks the appropriate item in Settings list 2. There is opened a new dialog on the screen which shows user the already existing data 3. If there is no data, the dialog will be empty 4. User changes the information and clicks "OK" button 5. The information is send to server 6. Server receives the data and sends response to the mobile device 7. On the screen there is shown notification informing user about the success of the operation **Alternative flow:** 1. – 3. The same as in the basic flow 4. User clicks "Cancel" button and the operation is withdrawn
ESN-TODOS-1	Showing the list of „ToDo" tasks	Registered logged user	Application shows current state of ToDo tasks, saved in the database on the mobile device	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. There is opened a new screen with tabbed view, including tasks ToDo after student's arrival and before their departure' 3. It is possible to switch between two tabs, marking the tasks regarding the after arrival and before departure activity
ESN-TODOS-2	Changing the state of ToDo task	Registered logged user	Changing the state of a ToDo task in the local database	**Basic flow:** 1. User clicks the checkbox next to the position on the list 2. Clicking checkbox changes the state of a given ToDo task for a logged user in the local database 3. The screen with the tasks list is

				updated 4. The value of a given task in the database on a mobile device is updated
ESN-USEFUL-APPS-1	Useful applications to install on device	Registered logged user	Application opens an embedded WWW browser, showing official of the Warsaw University of Technology	**Basic flow:** 1. User clicks appropriate position in menu 2. There are shown applications icon 3. Upon clicking the icon, user is redirected to Google Play Store to the page of the chosen application
ESN-WWW-ESN-1	Showing official ESN PW website	Registered logged user	Application opens an embedded WWW browser, showing official ESN PW website	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. There is opened a new screen with embedded WWW browser, showing the main page of ESN PW website **Alternative flow:** 1. – 2. The same as in the basic flow 3. If the user uses *BACK* button of the mobile devices, the application takes him back to the previous page of the browser, not to the previous screen of the application
ESN-WWW-PW-1	Showing official website of Warsaw University of Technology	Registered logged user	Application opens an embedded WWW browser, showing official website of Warsaw University of Technology	**Basic flow:** 1. User clicks the appropriate position in the main menu 2. There is opened a new screen with embedded WWW browser, showing the main page of Warsaw University of Technology **Alternative flow:** 1. – 2. The same as in the basic flow 3. If the user uses *BACK* button of the mobile devices, the application takes them back to the previous page of the browser, not to the previous screen of the application

Table 1 Functional requirements

3.3 Non-functional requirements

Apart from creating the list of functional requirements, there exist also non-functional requirements which need to be fulfilled in order for the application to work properly. These requirements are included in the Table 2 presented below.

Code	Description
NFR-1	Application gives user ability to change data on the server via the mobile application.
NFR-2	Some of the application functions do not require a constant connection to the Internet.
NRF-3	Application should guarantee a responsive reaction to user's actions.
NRF-4	The process of obtaining the data should take as little time as it is possible

NRF-5	Application should be able to maintain and process data for around 100 – 300 students.
NRF-6	There should be maintained a continuity between the data changed on the mobile device and the data stored on the server side.
NRF-7	The application should work properly on all type of mobile devices, from Android 4.0 to Android 5.1.

Table 2 Non-functional requirements

4. Technologies and programming environment used

In this chapter I to introduce some of the technologies and programming environment I used in my project, explaining briefly the way of their functioning. The first part of the chapter focuses on technology stack while the latter will thoroughly describe the environment I am using.

4.1 Technologies

4.1.1 Android Software Development Kit and Java

Android Software Development Kit provides developer with a lot of libraries and solutions created specifically for creating mobile applications. Basic Android applications are written usually in Java-like language, which uses a similar syntax as Java. The main difference is that Android does not use Java Virtual Machine but either Dalvik or Android Runtime environment.

Java bytecode cannot be executed on Andrid platform, but Java classes are compiled to a special format bytecode and run on Dalvik, so a virtual machine for Android. Dalvik generally uses less space and its interpreter has been simplified. Android however still uses java.lang package, so it is possible to query some of Java characteristic functions [3].

4.1.2 MySQL and SQLite

For this solution I chose simple relational database to store my data. I picked MySQL because it is widely available. It has a useful administrator panel. For the internal storage I chose default SQLite database because of its widely available documentation and simplicity. In the future database could be changed into a NoSQL database, but for now this solution is absolutely sufficient.

4.1.3 PHP

I am using PHP for my server side for several reasons. Firstly, it is open software and all its components are free to use. It is also backed excellently by a huge community developing and updating it constantly. It is very easy to seek for help in case of any difficulties.

PHP can handle a lot of traffic. It is used by a lot of big companies as their framework. It is also a language mainly used to code server side of project. PHP is not as aggravating as other solutions could be. For example, using Java server with Hybernate could be too much to handle for a simple mobile application.

PHP is easy to understand and it is easy to learn. It is as well platform independent. PHP is equipped with excellent mechanisms to support all major databases, therefore it is easy to process database operations. It is also characterized by built-in mechanisms supporting HTTP querying, making communication between server and client much easier. [29]

4.1.4 ButterKnife

ButterKnife is Android library used for injecting views to classes using them such as activity, fragment or dialog. Thanks to this class developer is not required to map a layout object to its equivalent in activity / fragment class programmatically. It also simplifies code, by restraining programmer from creating anonymous listeners for objects such as buttons. Thus, code is legible, clear and error proof [28].

Each mapping from layout object to its corresponding object in activity / fragment is done by creating an annotation at the beginning of the file. It binds layout object with its equivalent until the activity / fragment is ended. The view in ButterKnife is injected with command *ButterKnife.inject()*, called in a method, responsible for inflating the view for a certain activity. Injections can be also marked as optional and withdrawn by calling *ButterKnife.reset()*. The difference between the declaration in ButterKnife and standard mapping is shown in the Listing 1 snippet.

```
@InjectView(R.id.profilePic) ImageView imgIcon;
 @InjectView(R.id.name) TextView txtTitle;

VS

ImageView imgIcon = view.findViewById(R.id.profilePic);
TextView txtTitle = view.findViewById(R.id.name);
```

Listing 1 ButterKnife vs regular view mapping

4.1.5 Picasso

Picasso is an image library for Android. It is mostly used for image loading and processing. It simplifies the process of displaying image on the screen, mainly coming from an external location. It is so compact that sometimes it requires only few code lines to pull an image from URL, as it is shown in Listing 2.

As it was mentioned, it is especially helpful in displaying remote images, for example from the server. The library handles every stage of HTTP request until caching the image on the device. Obviously, image caching can be turned on and off, depending on user's needs. This library can be quite helpful, considering how much code would have to be developed in order to achieve the same goal [17].

```
Picasso.with(getApplicationContext()).load(ServerUrl.BASE_URL +
        session.getValueOfProfileImage()).memoryPolicy(MemoryPolicy.NO_CACHE,
                MemoryPolicy.NO_STORE).into(profilePic);
```

Listing 2 Picasso image loading snippet

4.1.6 Volley

Volley is a HTTP library that makes networking for Android much easier and faster than the default way, in which it is done. Volley offers some of very interesting benefits such as concurrent network connections, support for requests prioritization, scheduling of requests, and cancellation of requests.

Volley is most helpful at downloading data for populating UI components like ListView. It integrates with protocols and comes out of the box with supporting strings, images and JSON. However, Volley is not suitable for large download or data streaming, because it needs a lot of memory to hold data for parsing [10].

Volley has several request Objects ready to use just after installation, such as StringRequest, JSONObjectRequest and JSONArrayRequest. All of these constructors get several parameters including HTTP method to process the request, URL of request and response listener interface, implementing *onResponse* and *onErrorResponse* methods. This makes querying a lot easier, as it is possible to forget about the process and focus on parsing the response. An example of Volley library used in my project can be seen in Listing 3 snippet.

```
public class GetDirectFbIdTask {

    private final Context mContext;
    private String ID;
    private final Activity activity;
    private final String url;

    public GetDirectFbIdTask (Context context, Activity activity, String url) {
        mContext = context;
        this.activity = activity;
        this.url = url;
    }

    public void runVolley() {
        RequestQueue queue = Volley.newRequestQueue(mContext);
        StringRequest sr = new StringRequest(Request.Method.GET, "http://graph.facebook.com/" + url, new
Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                try {
                    JSONObject jsonObject = new JSONObject(response);
                    ID = jsonObject.getString("id");
                    try {
                        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("fb://messaging/" + ID));
                        activity.startActivity(intent);
                    } catch(Exception e) {
                        Toast.makeText(mContext, "Facebook is not installed. Please install Facebook!",
                                Toast.LENGTH_SHORT).show(); }
                } catch (Exception e) {
                    Toast.makeText(mContext, "Facebook is not installed. Please install Facebook!",
                            Toast.LENGTH_SHORT).show();
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Toast.makeText(mContext, "Facebook is not installed. Please install Facebook!",
                        Toast.LENGTH_SHORT).show();
            }
        });
        queue.add(sr);
    }
}
```

Listing 3 GetDirectFbIdTask

4.1.7 TextJustify

TextJustify is Android library used to justify text, shown on the screen via custom view – DocumentView - featured in this library. There are two layouts available for documents: DocumentView.PLAINTEXT and DocumentView.FORMATTED_TEXT. The first comes out of the box, while the second one is more customization friendly.

TextJustify gives also ability to customize text using HTML syntax. In this way it is possible to highlight some text fragments and make them more appealing. There is also a possibility of adding quotation style to text as well as caption under text, for example to provide users with author and date data [27].

4.1.8 Google Maps and Facebook API

There are two APIs which were needed for completion of my work. Both of them require a special access key, generated by respectively Google and Facebook. The application has to be registered in the companies' developer's system in order to be able to obtain such a key.

Google Maps API is very useful for working with Google Maps application. It lets us embed map in our activity using layout object *MapView*. There is also a possibility of adding markers to the map, showing places of interest. Google Map is highly customizable, so developer can invent all kinds of various listeners for different behaviors, modifying map for own purposes.

What is really helpful is that Google Map is distributed with automatic geolocation function as well as route finding and redirecting to Google Map application options. These features come completely out of the box and are added to the MapView after running our application [2][15].

Facebook API, on the other hand, is vital for querying the Facebook repository in search for data. Facebook has developed a website called graph.facebook.com, which enables querying the dataset via various endpoints and parameters. There is an option to query a certain page's endpoint as well as the currently logged user.

However, there are various levels of API authorization and on the basic developer one we get access to the majority of publicly available information, but we cannot access other user's profiles, which is restricted in order to maintain data protection. To sum up, Facebook API is extremely good solution for populating our application with Facebook data, but we have to take into consideration that not all the data is easily available [6].

4.2 Programming environment

4.2.1 Android Studio

Android Studio is a new integrated development environment for developing applications for Android devices. It was announced in 2013 and its beta version released in June 2014. The first stable build has been available since December 2014.

Android Studio is based on IntelliJ software, replacing Eclipse Android Development Tools as Google's primary IDE for native Android application development. Android Studio features a lot of functionalities, helping programmers in creating well organized and error free code [9]. The version I used comprised:

- WYSIWYG layout editor, enabling developer to just drag and drop objects onto sample Android screen, creating a layout

- Gradle based build
- Android-based code refactoring
- Analyzer of errors and redundancies
- Built-in template wizards to create Android components
- Integration with version control (Github and SVN)
- Android Device Monitor, used to control data and memory resources on the device

4.2.2 Gradle

Gradle is a project automation tool, highly supported by Android Studio. It contains information regarding SDK version used for the project, project's version, the build dependencies and libraries. It is based on domain-specific language instead of XML. Gradle was designed for multi-project builds which can grow to be quite large, and supports incremental builds by intelligently determining which parts of the build tree are up-to-date, so that any task dependent upon those parts will not need to be re-executed.

In Android Studio based project, the main directory features a subdirectory called Gradle Scripts. This directory contains all of *build.gradle* files, used for defining rules for a proper build. There is one general build file for a whole project as well as build files for all the modules used in project e.g. libraries and app module, which is equivalent of main project directory (/src).

Apart from the build.gradle files (see Listing 4), Gradle Scripts directory contains rules.gradle, settings.gradle (see Listing 5), gradle.properties and local.properties files. The file settings.gradle stores paths for project libraries and it is important to update them case of adding a new one to the project. Rules.gradle contains rules to perform, when unknown domain object is referenced [15].

```
buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile project(':library')
    compile project(':volley')
    compile 'com.jakewharton:butterknife:6.1.0'
    compile 'com.android.support:appcompat-v7:21.0.3'
    compile 'com.android.support:support-v4:21.0.3'
    compile 'com.google.android.gms:play-services:6.5.87'
    compile 'com.squareup.picasso:picasso:2.5.2'
}
```

Listing 4 build.gradle for app module

```
include ':app', ':library', ':volley'
```

Listing 5 settings.gradle

4.2.3 AVD / Genymotion

AVD as well as Genymotion are Android simulators, letting developers create virtual replicas of Android system and run it on computer. These tools emulate a large number of Android devices, from phones to tables and other gadgets. They can precisely simulate a lot of built-in sensors like battery, GPS, camera or accelerometer. This makes Android testing much easier to perform.

AVD is a built-in simulator featured in Android Studio. It is easy to use because of its availability and integration with Android IDE. However, it tends to be very slow while processing some more complicate applications. Fortunately, in the newer version features a x86 processor acceleration, which helps a little bit.

Genymotion, on the other hand, is a little bit harder to install and runs as a separate program using some virtual machine software as VirtualBox. However, the functionalities of Genymotion are considerably better than in the AVD. It runs extremely fast, installation of any additional applications or packages (for example Google Maps) is effortless – they have to be dragged and dropped on virtual device's screen. Additionally, Genymotion maps screen position changes much better than AVD, which helps in verifying whether the application acts the same regardless of being in horizontal or vertical position [8].

4.2.4 WAMP

I run my application on a local Apache server, distributed with WAMP software bundle, giving user out of the box access to database, server and PHP.

4.2.5 GIMP

For creating all the graphical elements I used GIMP. It is freeware software for creating and modifying raster graphics. It is a substitute for Adobe Photoshop with same functionalities.

5. Architecture and technical aspects of the application

The following chapter presents the ways in which application has been modelled and implemented. There are presented the decisions regarding operating system, programming solutions and modelling of the objects and relations between them.

5.1 Selected mobile system

Currently, on the market there are four key operating system providers, that is Google, Apple, Windows and BlackBerry. Although, in this year Apple company, with the launch of their newest smartphone iPhone 6 and other devices as Apple Watch, has hugely increased sales of their products comparing to the previous years, still, the product of Google - Android platform - remains the most popular and widely used operating system (*see Fig. 28 and Fig. 29*). It may be because of its much customizable functionalities, it may be also because Android does not vary much from iOS (or Windows Phone). The fact is that the devices operating on it are much more affordable for an average user than e.g. mentioned earlier iPad. This customer's preference is maintained in Europe, where Apple products are fairly popular but not as quite desired as in the United States of America [11].

Period	Android	iOS	Windows Phone	BlackBerry OS	Others
Q1 2015	78.0%	18.3%	2.7%	0.3%	0.7%
Q1 2014	81.2%	15.2%	2.5%	0.5%	0.7%
Q1 2013	75.5%	16.9%	3.2%	2.9%	1.5%
Q1 2012	59.2%	22.9%	2.0%	6.3%	9.5%

Figure 28 OS market share table [11]

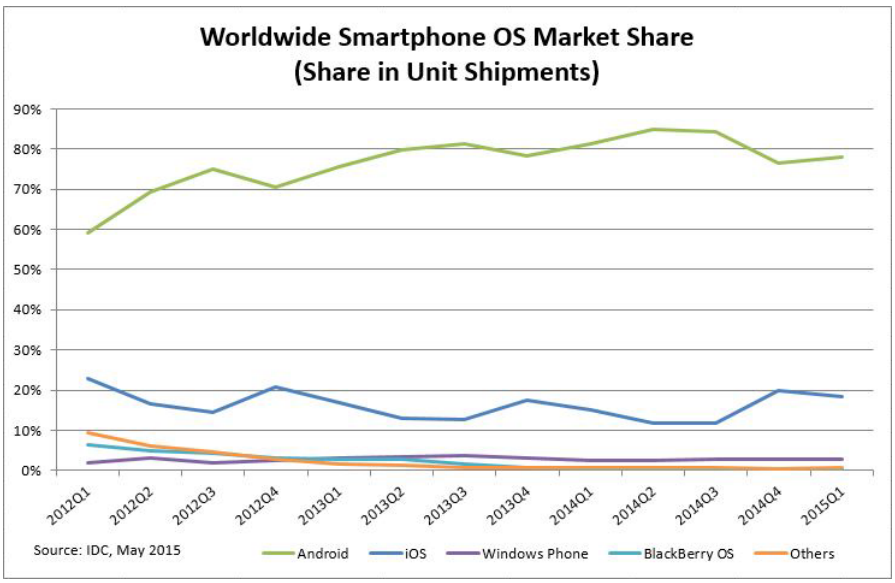

Figure 29 OS market share graph [11]

I would like my application to be as widely available to the users as possible therefore I found it natural to start off with an application for Android platform. In case of a positive

response from users, it would be beneficial to create the corresponding applications for the other systems - mentioned earlier iOS and Windows Phone.

5.2 Project structure

5.2.1 Directories structure

The project structure in my solution is a standard structure used in Android applications. The project consists of few folders, containing custom source files as well as generated files [15].

- *src*: This folder contains the Java source files, created by the user
- *main/gen*: Generated Java library, this library serves for Android internal use only
- *main/res*: This folder stores all the data, connected with view programming, such as pictures, XML files for defining layouts, and so forth
- *drawable*: Folder contained within Res folder. In here there are located various graphic files. We can see three types of drawable folders. This is because there are many Android devices with different screen resolutions. By default, there are several versions of this folder such as: *Drawable-mdpi*, *drawable-hdpi* etc. This is required in order to adapt to different screen resolutions
- *layout*: This is the place for XML layout files. Layout files are XML files which define how various Android objects and widgets (such as textboxes, buttons, etc.) are organized on the screen
- *values*: XML files which store various string values (titles, labels, resolutions etc.)

5.2.2 Design pattern

General rule for communication between components in Android is that the Activity classes (described further in a more detailed way) combine domain objects, created by the user, with view injected into them. A part of activity is a variable of type *View,* which is used by Android to present data on the screen. The layout is created within XML layout files, where developers can establish dependencies between objects, visible on the screen [16].

In Activity class, upon creating an instance, the first method called is overridden *onCreate* method. This method bounds activity with a given layout file, being called with a given pattern *R.layout.layout_name*. After binding, there is a possibility of accessing objects shown in the view layout programmatically. At first, in the activity there have to be created objects of the same type as objects we want to map to, defined in layout file. Each object in layout file should be provided with a unique id (at least unique within one file) in order to be manageable in a proper way. From Activity class, we can call method *findViewById*() which binds layout object with their mapping in activity. Then, the objects can be modified programmatically and every change done to them will be shown on the screen.

R class, which was featured in the explanation above, is a generated class, responsible for creating unique identifiers for all the layout objects: not only widgets but also pictures, xml files, string and other resources. This class contains only public static final declarations of variables and is updated upon adding a new object which binds variable with the code.

As it was presented, this pattern lays somewhere between MVVM and MVC pattern, but is not quite any of them, because it has neither explicitly declared Controller nor ViewModel.

5.2.3 Android Manifest

AndroidManifest.xml file is one of the most important elements enabling developer to define application features and its behavior regarding communication between the program and the system. AndroidManifest should be located in project's main folder, because it contains information essential for the application to be even run. AndroidManifest used in my project can be seen in Listing 6 snippet [30].

Manifest has several duties regarding the application, some of which are:

- Naming Java packages for the application, as it serves as a unique identifier for the application
- Describes the components of the application – activities, services and content provider that application is composed of. These declarations let Android system know which components can be launched under what conditions
- Determining which processes will host application components
- Declaring permissions for the application to user protected API parts e.g. internet access, camera access, geolocation access etc.
- Declaring permissions other components are required to have in order to communicate with our application
- It declares the minimum level of Android API that the application requires to run on a mobile device
- It lists the libraries which application has to be linked to in order to run properly
- Marks one activity as a launcher, so the application will start by running the chosen activity

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.wiacek.martyna.esnpwapp" >

   <uses-permission android:name="android.permission.INTERNET" />
   <uses-permission android:name="android.permission.CALL_PHONE" />
   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
   <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
   <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

   <application
      android:allowBackup="true"
      android:icon="@drawable/logo_6"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <activity
         android:name=".Activities.Login"
         android:label="@string/app_name" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
```

```
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".Activities.NavigationDrawer"
            android:label="@string/title_activity_entirely_new_drawer" >
        </activity>

        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="@string/google_maps_key" />
    </application>
```

Listing 6 AndroidManifest.xml

5.3 Architecture of the solution

The architecture of the solution consists of the mobile application, web server and the database and its outline is presented on the Figure 30. The communication between components is realized as follows. The mobile application performs queries to web server via HTTP requests. The arguments of the queries are either in a simple key-value form or in JSON format.

After that, the server receives the query, parses its arguments and connects to the external database via class *DB_Connection*. This class opens a new connection to the database and tries to manage it, closing it after performing an operation. Afterwards, the database is called to perform SQL query returning data to the server, which is then sorted or parsed, according to needs.

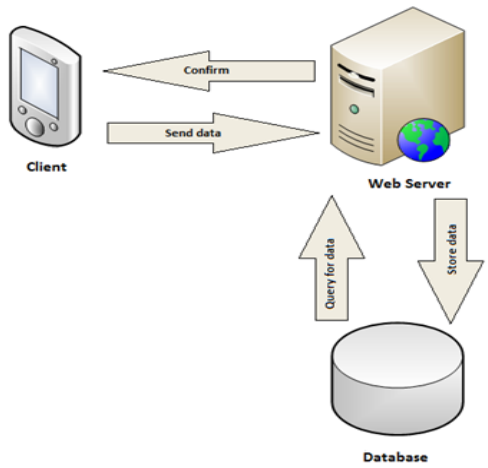

Figure 30 Solution architecture

Server returns data to the mobile application in JSON format. After that, the application gets the data as a response from server, then parsing it on a client's side and mapping into Java objects.

Communication between application and PHP server using JSON object is very simple and handy, because they both are provided with great tools for parsing and performing these types of queries.

5.4 Modular division

The project is divided into modules, in order to separate all the functionalities, creating APIs to easily connect all of them. I wanted to maintain separation in order to make code more scalable and editable.

My *src/* directory contains following directories:

- Manifests
- Java

Manifest folder contains AndroidManifest.xml file, described in paragraph 4.2.3.

Java folder contains all the source code for my project and the structure is presented on the Figure 31.

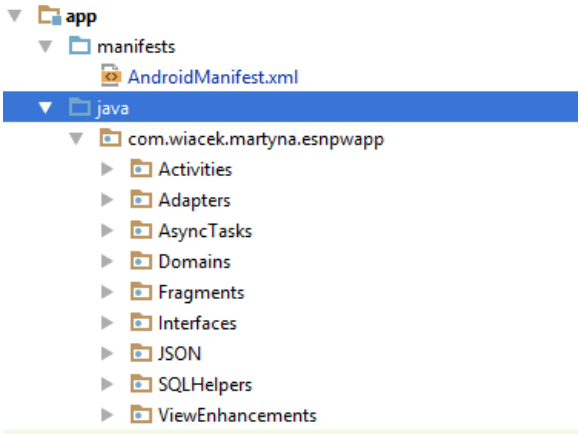

Figure 31 Modular project structure

Activities – this directory contains code of two of my Activities: one is the *Login* Activity, which is the main Activity from with the project is started. The second one is NavigationDrawer, which is responsible for managing the NavigationDrawer scheme in project.

Adapters – this directory contains all the custom Adapters, created for the project, in order to improve the view of presented screens.

AsyncTasks – this directory contains code asynchronous tasks and for HTTP queries. Some of them are performed using AsyncTasks while the others are done with Volley library, which uses a mechanism similar to AsyncTask. I decided to put all of them in the same directory, because they are essentially working in a similar way, therefore performing long operating tasks.

Domains – this folder contains all the domain models, necessary for mapping of the object to database. Here is also located a class called *ServerUrl*, which is a static class, containing URL to server. It contains also *SessionManager* class, which manages user's session, using SharedPreferences.

Fragments – in this directory are located all the files containing Fragments, displayed by NavigationDrawer class. All the fragments are completely separate, therefore changing one on them will have no impact on the others.

Interfaces – this directory includes interfaces, used for performing callbacks from AsyncTasks back to the Fragment it was called from

JSON – this directory contains class called *JSONFunction*, which encapsulates JSONObjects into objects, used in the application.

SQLHelpers – the directory contains class responsible for creating and managing internal SQLite database. The class performs inserts, updates and deletes on given rows in the local database, trying to maintain data continuity.

ViewEnhancements – this directory contains a few classes, used for enhancing the view of some application screen. There are classes operating on Layout libraries as well as classes, responsible for presenting popup dialogs on the screen.

5.5 Authentication

In creating the process of the user authentication in my application I based it on real methods of authentication. Database stores username, password hash and salt, where salt is a random data used as an additional input to function which hashes password.

The process of hashing password is quite simple: we generate salt by hashing a random number with sha1 method. Then, we perform some arithmetic logic on this operation to obtain final salt. The next step is to encode password with *base64_encode* method. The argument given to the function is sha1 of concatenation of password and salt.

To obtain results there similar work has to be done, only in the opposite order. Username and password are passed to the function, where there is performed a query to the database. Then the data which was inserted is encoded in the same way described before, using salt taken from database. If the newly created hash equals hash stored in the database, the authentication can be assumed as positive and authorization is given to the user.

In case of changing the password by the user, *changeUserPassword* function is called, which returns a hash created with newly generated salt and inserted password and then updates database.

5.6 Session

The application is capable of storing its login session, in order to perform correctly any HTTP requests and be able to manage the data in local database, regarding a particular user.

The user's session is created upon logging in using the Android build in *SharedPreferences* mechanism with appropriate tag to easily distinguish session. SharedPreferences is a class which lets developers store a small amount of data. The premise is that this mechanism stores information about account, application, interface settings etc. The main restriction is the fact, that SharedPreferences can only store simple type of data, so there is no possibility of storing custom objects. The main advantage is that thanks to this approach it is possible to share data between all the components of application. If data is not deleted, it can be stored also after rerunning our application [15].

The data which is stored as session is user's username id on server and other information as contact details or faculty name. All this information can be represented with simple types. This data is essential for performing a call to server, as majority of calls require providing the user's id.

When the application is resumed, the session is simply restored, ready for the next use. Upon logging out, the session is destroyed, that is all the data belonging to a certain tag is cleared and committed.

Because the tag for distinguishing the session remains always the same, it is impossible to store more than one session on one device. This mechanism ensures as well no overwriting of the session.

5.7 UI Scheme

5.7.1 Navigation Drawer

In the past few years there has been observed an increased popularity of applications, using a swipeable menu bar, appearing from one side of the screen. This kind of UI scheme is called Navigation Drawer and it enables the user to have a quick access to the main menu from every part of the application, so there is no need to go back to a certain screen. This approach also lets architects divide functionalities into categories and list them under a certain headline, creating more organized and easily navigable application [14].

There are several big companies who have incorporated this scheme into their applications i.e. Google, LinkedIn, Facebook, Twitch. In fact, Google can be called a big supporter of this solution as on the Android Developer website it is mentioned as one of the best available ones.

In addition to easy navigation, creating applications using Navigation Drawer has another positive aspect, this time for the programmers. Using technologies usually linked with web development like JavaScript and its frameworks it is possible to create an application, which could work on all of the most popular mobile operating systems. This solution is presented by Phone Gap, which is a framework, letting developers create applications using standardized API. Of course, it still requires backend changes regarding different mechanisms used in each operating system, but it lets developer create a common GUI once, and then use it for all the devices. A common GUI makes applications more consistent [16].

The other reason is trying to merge mobile applications with web applications, therefore get users accustomed to same patterns of navigation in order to be able to reuse same GUI for different solutions. In this way, users will find applications created with a similar pattern more intuitive.

In my application I implemented my own Navigation Drawer, based on fragments rather than activities.

5.7.2 Activity

An activity is an application component, equated with one application screen, so everything which is seen on the device's screen (windows) belongs to the activity. Usually, the application consists of several activities, loosely bounded to each other. One of them is marked as a main activity in A*ndroidManifest,* containing all the settings and dependencies regarding activities. One activity then can call another one, stopping itself and passing control to the new one. When Activity is not active on the screen, it is stopped and put on activity stack (*"back stack")*. In case of other activity being finished, the system will pop the recently added Activity from the stack. Each Activity has its own lifecycle so depending on the actions

taken it can perform a callback to a certain application and be resumed, killed or started once again [15].

However, as in Navigation Drawer scheme there is always a need to open the swipeable menu, it would be highly inefficient to create a new activity each time the menu is called to be shown. Therefore, it would be beneficial to completely connect the activity with the menu and change only a part of the screen, showing views for a chosen position from the menu. Therefore *fragments* are response for this issue.

5.7.3 Advantages of fragments

A fragment represents a behavior in an activity. Fragments can be combined together, creating a multi-pane interface for activities. Essentially, they are modular sections of activity, having their own separate lifecycle, being able to be reused or removed. The lifecycle of the fragment is bound to the host activity lifecycle, therefore if the activity is paused, so are the associated fragments. Each activity has its own *Fragment Manager,* enabling it to control the fragments, delete them or add to fragments stack.

As it can be seen, it is a very useful solution, enabling developer to manage fragments easily and spare application's resources as it is easier to create a new fragment rather than create a new activity [15].

In the Navigation Drawer implementation, upon clicking on a position in the menu, last used fragment is replaced by a new one in the Fragment Manager and is shown on the screen.

The implementation of the main function selecting fragments to be inserted into Fragment Manager can be seen in the snippet Listing 7.

```
    if (savedInstanceState == null) {
        SelectItem(1);
    }

void SelectItem(int position) {

    Fragment fragment = null;
    boolean useNativeFragment = false;
    Bundle args = new Bundle();
    int positionInDataSet = position - 1;
    switch (position) {

      case 1:
        fragment = new HomeFragment();
        args.putString(HomeFragment.ITEM_NAME, dataList.get(positionInDataSet)
            .getItemName());
        args.putInt(HomeFragment.IMAGE_RESOURCE_ID, dataList
            .get(positionInDataSet).getImgResID());
        break;
       ....
      case 25:
        fragment = new HelpFragment();
        args.putString(HelpFragment.ITEM_NAME, dataList.get(positionInDataSet)
            .getItemName());
        args.putInt(HelpFragment.IMAGE_RESOURCE_ID, dataList.get(positionInDataSet)
            .getImgResID());
```

```
        if (savedInstanceState == null) {
            SelectItem(1);
        }

    void SelectItem(int position) {

        Fragment fragment = null;
        boolean useNativeFragment = false;
        Bundle args = new Bundle();
        int positionInDataSet = position - 1;
        switch (position) {

            case 1:
                fragment = new HomeFragment();
                args.putString(HomeFragment.ITEM_NAME, dataList.get(positionInDataSet)
                        .getItemName());
                args.putInt(HomeFragment.IMAGE_RESOURCE_ID, dataList
                        .get(positionInDataSet).getImgResID());
                break;
              ....
            case 25:
                fragment = new HelpFragment();
                args.putString(HelpFragment.ITEM_NAME, dataList.get(positionInDataSet)
                        .getItemName());
                args.putInt(HelpFragment.IMAGE_RESOURCE_ID, dataList.get(positionInDataSet)
                        .getImgResID());
                break;
            default:
                break;
        }
```

Listing 7 SelectItem in NavigationDraver.java

5.7.4 Action Bar

Another UI design pattern is *ActionBar* - an option bar, located in the upper part of the screen. The ActionBar can contain its own icon, linking to functionalities in application, but more importantly it has a changeable title and is linked with Navigation Drawer in order to show it properly on the screen.

In case of my application, the title of the ActionBar depends on the currently chosen Fragment and it contains icons picturing Settings and Logout functionality. In the left part it contains icon, opening and closing the Navigation Bar.

5.8 Data storage

In my application storage of the data is divided into two sections – storing data in an external database and in the internal database, located on the mobile device. In this way we can ensure that we will be calling external database only to get the most recent data, while we will be using personalized data in our local database. The data is synchronized upon user's logout from the application.

5.8.1 External database

The external relational database MySQL is located on a server, which we are querying for the requests. The database contains the most vital data, therefore the data about the system's

users, recommended places, ToDo tasks etc. This data has to be pulled usually upon user's login and then updated on the local devices. Data, which changes more often, is returned upon every query to the database, triggered by e.g. clicking the button in the application.

In the external database the following tables are featured:

- *Buddies* – data of the assigned buddies to foreign students, including their contact details
- *Funmap_categories* – categories in which the FunMap places are divided into
- *Funmap_places* – description of places, featured in the application
- *Funmap_places_categories* – joined table, which can enable one place to belong to more than one category
- *Students* - table describing registered users details, contact details and profile pictures
- *Todo_public_list* – table containing ToDo tasks, which need to be done by a student. The tasks are default and inserted into table by the administrator.
- *Todo_users_data* – table containing data about ToDo tasks, regarding a particular student
- *Users* – data needed for user's authorization, login and password etc.

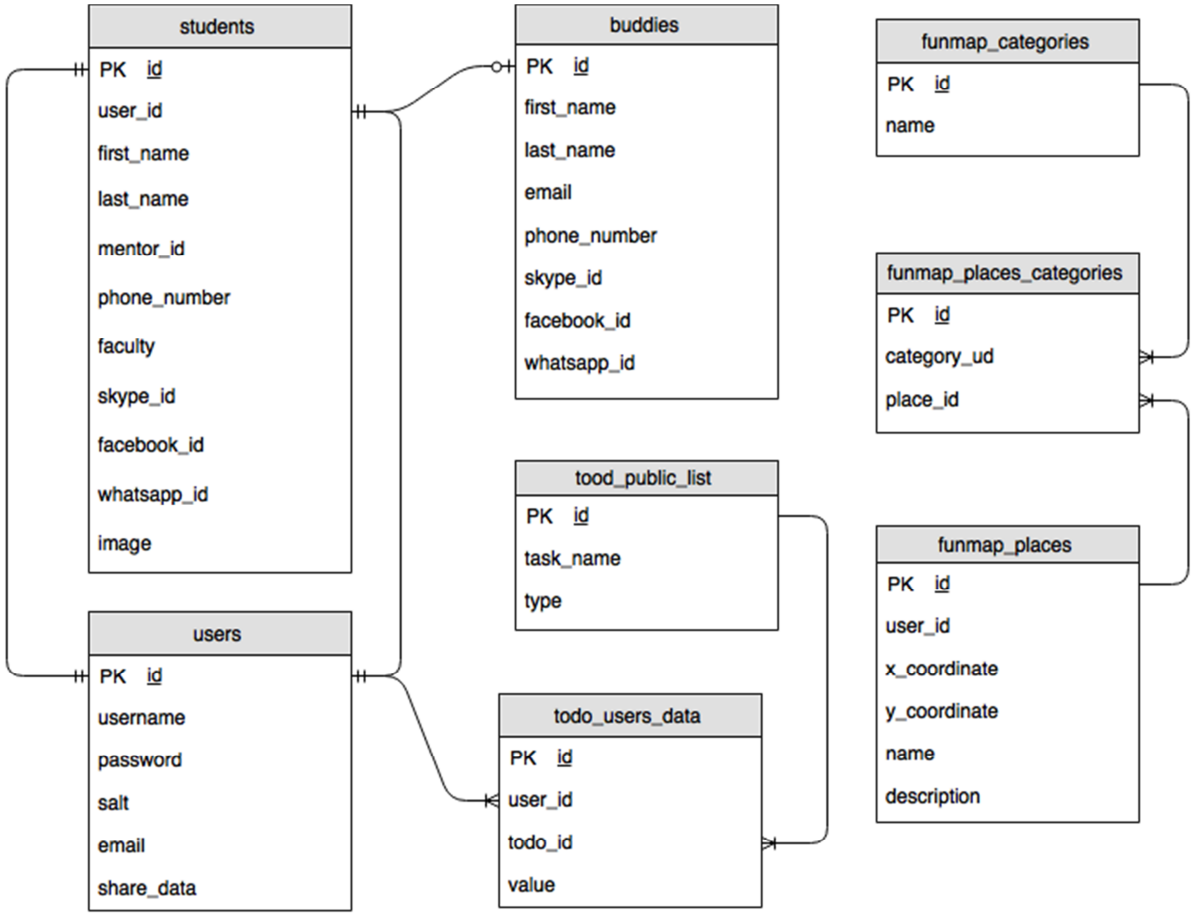

Figure 32 External database ER diagram

The E-R diagram is presented on the Figure 32. The entities identifiers are marked with PK symbol and are underlined. The other data in the entity are either identifiers used for creating relations between entities (they are located at the end of the arrows, connecting entities) or plain data, being a part of entities declaration.

5.8.2 Internal database

The internal database is located on a mobile device using the application and is created upon user's login. The database is based on a SQLite library, therefore it features relational model, just like the external database.

Essentially, the internal database should enable multiple users to login on the same device. It is at the same time storing the data, which is beneficial for user who does not have the internet connection.

There were created two main tables, *Buddy* (*see Table 3*) and ToDoTasks (*see Table 4*). These tables are used to store information about user's mentor contact details as well as about ToDo tasks, modified locally. It is beneficial to avoid sending a request to the server on every change in ToDos table, therefore rather than calling the server very often, we call only from time to time, to synchronize both databases.

As it was mentioned earlier, after user's login, application pulls the data out of the external database and stores it locally, in order to be able to work on a selected dataset. Local data is synchronized with external data by performing a call upon logout, which updates ToDo's values in external database. Thus, the data flow and consistency can be preserved and after logging back, user will be provided with up-to-date data.

Field name	Field type
id	INTEGER
user_id	INTEGER
first_name	TEXT
last_name	TEXT
email	TEXT
phone	TEXT
facebook	TEXT
skype	TEXT
whatsapp	TEXT

Table 3 Buddy table

Field name	Field type
id	INTEGER
user_id	INTEGER
todo_id	INTEGER
todo_name	TEXT
todo_type	INTEGER
todo_value	INTEGER

Table 4 ToDo table

5.8.3 Problems with online and offline data

As it could be noticed, I combined two approaches, used in the data storage scheme - offline data and online data. However, both of them are characterized by some advantages and disadvantages. The solution which automatically comes to mind is cloud storage, but for such a small application it would be completely unnecessary.

Both online and offline representation and storage of the data can have their advantages and disadvantages [13].

Online approach advantages:

- The data is always up-to-date
- There is no need to store data on the mobile device, therefore our application is lighter

- It can help measuring the popularity of application features
- Supports maintaining data integrity
- Simpler transition to cloud computing

Online approach disadvantages:

- Server has to be called a numerous times, therefore
- Server can be flooded with too many requests from too many users
- User has to have a connection with internet

Offline approach advantages:

- The data is always available
- It minimizes probability of server overloading
- Data requests for a local database are quicker than those for the external
- Application has control over caching data

Offline approach disadvantages:

- The data can be outdated
- The data has to be stored locally, so the application size grows
- The administrators have no control over user's data and obtain no information about the data's popularity
- There has to be introduced a good synchronization system

As it can be easily seen, there is no easy way to decide which one of this approaches should better considering my application. I decided to divide the functionalities basing on the validity of the data. If the data is extremely necessary for the user - it is stored locally. If data is updated often – it is stored remotely.

5.8.4 JSON files

As project features a lot of descriptive data common for all of the users, I decided to store it in text files, where is located in JSON format.

JSON is an acronym for JavaScript Object Notation. It is very light, so it is incomparably simpler to read and store this data rather than storing the text data in database. This notation is common for all of the most popular programming languages, so it can be a good way to share data between two main languages I used for my project - that is Java and PHP.

Two files, *partners_data.json* and *qanda_data.json* as their name suggest respectively contain data regarding ESN partners - the possible discount offers for the exchange students and a set of Frequently Asked Questions.

This particular data is widely available and rather well-know, therefore I decided not to introduce any specific means of security in this case.

5.8.5 Text files

In my application, data which is changed rarely is stored in text files, embedded in the file structure of my project. The cases, in which this solution is used, are those regarding transportation, documents or accommodation. The information is fairly consistent and static, therefore it can be stored in a simple text file. On the other hand, these reports are absolutely vital for a new coming student, therefore they have to be stored offline. The files are characterized by flat structure and some of them contain some HTML formatting, in order to be presented better on the screen.

In case of any change in the domains presented, there will be needed an update for an application. We can tolerate this solution, as these changes would be in fact very occasional.

5.8.6 RESTful application

Representational State Transfer (*REST)* is an architectural style, which can be described with six constraints: Uniform Interface, Stateless, Cacheable, Client-Server, Layered System, Code on Demand [15].

- *Uniform Interface* – this constraint defines the interface between clients and servers. It simplifies and decouples the architecture, which enables each part to evolve independently. An important feature is the fact that resources are identified by URIs and are conceptually separate of the representation, returned to client. Instead of sending the database queries results to the client, JSON data is sent, representing records coming from database. Client should also hold information, enabling him to manage resources on server.
- *Stateless* – this constraint explains, that the necessary data, used to perform query is contained in query itself. There is no need to manage any other state of application. The resource should be defined separately and not dependent on the state of application (for example its session).
- *Cacheable* – responses should, implicitly or explicitly, define themselves as cacheable, or not, to prevent clients reusing stale or inappropriate data in response to further requests. It could partially or completely eliminate some client–server interactions, further improving scalability and performance.
- *Client*-Server – clients should be at all costs separated from server. Clients should not be concerned by server's storage, which makes application portable, which in case of a mobile application is very important. On the other hand, servers are not concerned by user's interface therefore both components can be developed separately.
- *Layered system* - according to this rule, clients should not be able to tell whether the connection is directly to the server or to an intermediary server.
- *Code on demand* – server should be able to extend or customize the functionality of a client by transferring logic to it that it can execute.

In case of my application, I wanted to fulfill all the rules mentioned above. The queries for data are performed with HTTP.POST (if we want to create a certain resource or change it) and with HTTP.GET (to get the resource to client).

To get data from server, my application is sending a request with parameters, which then is sent to server, where it is delivered and passed to the function, which performs action on data (in this case, pull data from database) returning response. The response is usually in JSON format, if we consider HTTP.GET or returned as String.

After performing a query HTTP response is sent to client and processed exactly on client's side. JSON is parsed using a special class *JSONFunctions* and then mapped into objects, afterwards presented on the screen. To perform such a query my application uses Volley library and standard HTTPRequest class from Android, which will be explained in the next chapter.

After query's success user is informed about it with a notification on the screen. In case of any errors, user is notified as well and all the changes rolled back.

5.9 Custom SQLite Helper

In order to manage properly an internal database and its version on the mobile device it is recommended to implement own class extending SQLiteOpenHelper. The new class has to implement *onCreate(SQLiteDatabase)*, *onUpgrade(SQLiteDatabase, int, int)* and optionally *onOpen(SQLiteDatabase)*. The class takes care of opening the database if it exists, creating it if it does not, and upgrading it if necessary. Transactions are used to make sure the database is always in a sensible state. This class makes it easy to defer opening and upgrading the database until first use, to avoid blocking application startup with long-running database upgrades. The number of version after upgrade is increased in a default way.

What is worth noting is that one SQLiteHelper can be attributed to only one database. It should not be perceived as a helping class for managing one table but the whole database, taking care insertions, updates and deletions of data.

Database is created with SQL statement, executed on database instance. Method *onUpgrade* works in a similar way. To insert values into database I used ContentValues class, used to store a set of values. Values are inserted into set following pattern *values.put(KEY, VALUE).* After that there must be open a writeable instance of database and the values inserted.

I am inserting data into database upon user's login to the application. In order not to drop and create tables every time I am using *insertWithConflict* method, which creates columns and rows only if they do not exist in database, otherwise it updates values in the rows.

Another helpful feature in pulling the data from database is Cursor interface. This interface provides random read-write access to the result set returned by a database query. Cursor also grants many methods facilitating iteration through the records. It should be closed at the end of operations and can be resumed and refreshed to get more accurate data. Combining all this information I created a helper class, accessing the dataset and modifying it only when it is absolutely necessary.

5.10 Views customization

5.10.1 Adapters

In custom Android view creation there are widgets and objects which can be bound to data source. In order to provide them with data Android uses *Adapters*. Adapters are a link between a set of data and AdapterView, which displays the injected data. On the other hand, AdapterView is a ViewGroup which displays child view given to it by an adapter. Adapters are responsible for supplying the data and creating the view representing them.

Android as default is equipped with a set of native adapters, ready to use out of the box. Some of these adapters are: ArrayAdapter, ListAdapter, CursorAdapter, GridView, SpinnerAdapter. If we need to show a simple set of data, they will be probably more than enough.

In my application there was a need to display data in e.g. ListView in a different way, than what Android Framework does a default. Therefore, I was obliged to write my own Adapters, extending basic classes, available out of the box. A noteworthy example can be

EventsAdapter, which is a ListAdapter, showing events details. Adapter class extends default class mentioned above. The main method is an overridden *getView*, which inflates view from a layout resource, passed to it in a constructor. To make code clearer, there is declared an inner static class called EventHolder, which has all the layout objects, needed for mapping, declared as fields, so it is easier to manage it. The example of a custom Adapter used in my project is shown in the code snippet Listing 8.

```
public class MentorAdapter extends ArrayAdapter<MentorContact> {
    private final int layoutResourceId;
    private MentorContact[] data = null;

    public MentorAdapter(Context context, int layoutResourceId, MentorContact[] data) {
        super(context, layoutResourceId, data);
        this.layoutResourceId = layoutResourceId;
        this.data = data;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View row = convertView;
        MentorHolder holder;
        if(row == null ) {
            LayoutInflater inflater = LayoutInflater.from(getContext());
            row = inflater.inflate(layoutResourceId, parent, false);
            holder = new MentorHolder(row);
            row.setTag(holder);
        }
        else  {
            holder = (MentorHolder)row.getTag();
        }
        MentorContact mentor = data[position];
        holder.txtTitle.setText(mentor.name);
        holder.imgIcon.setImageResource(mentor.icon);
        return row;
    }

    static class MentorHolder    {
        @InjectView(R.id.profilePic) ImageView imgIcon;
        @InjectView(R.id.name) TextView txtTitle;

        public MentorHolder(View view) {
            ButterKnife.inject(this, view);
        }
    }
}
```

Listing 8 MentorAdapter.java

After creating adapter, the class is afterwards called in Activity method (in my case in Fragment) and set to an layout object e.g. *ListView* with a following line

listView.setAdapter(adapter);

Adapters are highly adjustable to developer's needs and let them create even very complicated view. At the same time created views are surely more compact, making application store more data in less screen space.

5.10.2 Preferences

Applications often include settings screens that allow user to modify application features or behaviours. If there is a need to provide users with setting screen, there is also a need to use Preference API to build this kind of activity.

Preference API lets us build an interface consistent with other Android applications, even with default Android system settings menu. In this way, we ensure user that this particular screen will be intuitive and easily navigable, so it improves user's experience.

This class is much different from the other activities or fragments, regarding the way of building the view layout. Instead of a simple layout resource, there should be used an XML dependencies file. One Preference object is a building block for a single setting. Each preference appears on the list and provides the appropriate UI for users to modify the setting. Some of the settings can be modified directly from PreferenceList (like Checkbox Preference), some others on the other hand need Preference Dialog [15].

Normally, the local preferences are used in SharedPreferences file, which was described in previous chapters. There is still a possibility of binding a certain preference with task, so with those we can change preferences on an external sever as well. From Android 3.0 there is available creation of PreferenceFragments as well as PreferenceActivities.

Every setting in the application is represented by a specific subclass of Preference class. Each subclass includes a set of core properties that allow you to specify things such as a title for the setting and the default value. Each subclass also provides its own specialized properties and user interface. The most common preferences are CheckBoxPreference, ListPreference and EditTextPreference.

In Preference XML file Preferences can be divided into groups, creating preference categories in order to manage preferences more easily. Each preference can be identified by its id, so it is easier to provide them with their backend and also to create a nice looking interface.

In my solution, I store setting of my particular user in SharedPreferences as a part of my user's Session. Because the preferences mainly affect the visibility and data user is sharing with other users (e.g. with GetInTouch functionality every user has an option to contact any other publicly visible user), I am updating the value both in SharedPreferences and on server, sending a HTTP Request with updated data value. Thus, I can ensure that the users will always get up-to-date contact data of other users.

5.10.3 Custom Dialogs

The other highly necessary aspect of views customization is *Dialog.* Dialog is a small window that prompts user to make a decision or enter some information. Dialog does not fully fill the screen it is used for modal events which require user to take action before the application proceeds.

The base class for all the dialogs is the Dialog class, but it is better to use classes with a higher abstraction level as AlertDialog or DatePickerDialog. It is also possible to use DialogFragment to create highly customizable dialog. It provides all the controls to create such a container and manage its appearance. DialogFragment also ensures the lifecycle will

be managed correctly, even in case of unexpected events as pressing *Back* Button by user or rotating the screen.

Dialog consists of three main parts: Title, Content area and Action buttons. Title should be used only when the content area is rather complicated. In other cases the content should be self-explanatory. There should be no more than three buttons in one dialog. The content area can display a message, a list or other custom layout.

Class which provides API for creating a dialog is called AlertDialog.Builder. The Builder is precisely the object to which title, messages and buttons can be added. After initialization, Builder lets us create an instance of dialog.

In my case, I adopted AlertDialog as well as PreferenceDialog. PreferenceDialog has mostly the same functionality as AlertDialog but is used for creating dialogs used in Preference View.

PreferenceDialog should override three methods: *onPrepareDialogBuilder*, where the title is set and the constructor called, *onClick*, which creates a certain behavior for a dialog, depending on the type of button clicked and finally *onBindDialogView***,** binding view of the Dialog with a proper Dialog. Building such dialogs is quite easy to perform and creates a lot of possibilities of customizing dialog templates.

5.11 AsyncTasks

While developing application, I encountered a common problem regarding concurrency. During performing my query to server, I observed that the view for a certain fragment had been updated much quicker that processing of my response. That is because in Android all the components of application – both visible and running in background – are run in the same thread UI Thread. When one of the components is loading, the rest is stuck, waiting for the action to finish. Fortunately, a great solution for this situation – *AsyncTasks* - is embedded into Android platform.

AsyncTask is an abstract class, which moves long running task to a new thread. Additionally, it enables running tasks in a main thread UI Thread, therefore it is possible to control users interface. It is also possible to show the result of an operation on the UI Thread, without having to manipulate threads [15].

AsyncTask consists of four methods:

- *doInBackground* – this method has to contain long running operation. For example, when onClick method is executed on click of button, it calls execute method which accepts parameters and automatically calls doInBackground method with the parameters passed.
- *onPostExecute* - this method is called after doInBackground method completes processing. Result from doInBackground is passed to this method and it is the only one, which can make changes in application view
- *onPreExecute* **-** this method is called before doInBackground method is called
- *onProgressUpdate* - this method is invoked by calling publishProgress anytime from doInBackground call this method

AsyncTask class is defined by its parameters: `AsyncTask<Params, Progress, Result>`. The first is input data type, the second – the type used to represent progress in the action and the last one – type returned by class. AsyncTask is run by calling *execute()* method and in the execute() method can be passed parameters, used in the process of running AsyncTask.

```
public class CreateFacultyLinksTask extends AsyncTask<String, Void, String> {

    private final OnTaskCompleted listener;
    private final Context mContext;
    private ArrayList<String> data;

    public CreateFacultyLinksTask (Context context, OnTaskCompleted listener) {
        this.listener = listener;
        this.mContext = context;
    }

    protected String doInBackground(String... urls) {

        try {
            data = new FacultyNames().getDetailedData(urls[0]);
        }
        catch (Exception e) {
            return "-1";
        }
        return "0";
    }

    protected void onPostExecute(String result) {

        if (result.equals("0"))
            listener.onTaskCompleted(data);
        else
            Toast.makeText(mContext, "Error. Cannot create faculty links!", Toast.LENGTH_LONG).show();
    }
}
```

Listing 9 CreateFacultyLinksTask.java

In order to be able to show the data on the screen after it was pulled and modified by the client side, I used AsyncTasks and, in some cases Volley library, which is an Android library used to perform HTTP requests.

However, I wanted to maintain modular division in my project as well as use AsyncTasks classes to update view of my application, while being located in separate files. To perform this behavior, I decided to use mechanism of callbacks. The main function calls a constructor for Asynctask, inserting an instance of prepared interface as one of parameters. This interface usually has just one method, used for injecting already prepared data to another module.

One of AsyncTasks private fields is exactly an interface. To this interface we assign a new instance, passed in a constructor. In the method *onPostExecute* we call method defined in

the interface, passing our changed object as argument. In the primary function the inserted interface will get data and modify it, having access to the view.

This simple injection of dependencies can help massively in encapsulating code into classes, maintaining the same functionalities. The code example of my solution can be found in the snippet Listing 9.

5.12 JSON to Objects

As it was mentioned in above chapters, the data coming from external database is returned in JSON format. In order to map it on the domain objects, I created a class *JSONFuntions* which helps in achieving this solution.

In the class there are a few methods basing on *com.json.JSONObject* library - a standard Android JSON addition. The methods take a JSONObject as a parameter and return an object of a chosen type. JSONObject itself is an unordered collection of name/value pairs. Its external form is a string wrapped in curly braces with colons between the names and values, and commas between the values and names. The internal form is an object having get method for accessing the values by name, and put methods for adding or replacing values by name.

The values can be any of these types: Boolean, JSONArray, JSONObject, Number, String, or the JSONObject. A JSONObject constructor can be used to convert an external form of JSON text into an internal form.

When accessing and creating JSONObject there is a possibility of throwing an Exception by the object, therefore it requires cautiously written code, which can serve the Exceptions.

If there is an opportunity that certain field in JSONObject is optional, we can check it by using method *has*, which operates in a similar way to the method *has* used in case or Lists in Java. Thus, we can ensure we will be accessing a certain field of JSONObject only if the field exists.

5.13 Server side

The server side is written in PHP language, usually used exactly for this purpose. The main directory contains php files, which are used as an endpoint for querying from mobile application to server. The structure of project is presented below:

- main
  - include
    - DB_data
    - JSON_data
    - partners_pics
    - user_profile_pics

/include directory contains four folders, each having a separate role in the process. DB_data stores classes' declarations of object, used to access and modify database, returning a response. */partner_pictures* contain pictures of all the partners, featured in ESN Offers Activity. Finally, the last directory contains user's pictures uploaded on server.

Files from the main directory just simply access HTTP methods in search of parameters and then, passing parameter further to a certain initialized object and return response with PHP echo function.

Files in DB_data are just classes, containing construct and destruct method, run before accessing the database. They fetch DB results in associated arrays or JSON objects, encoding data with *json_encode.* This provides a secure way of accessing data with no possibility of leaks.

Additional and separate class is *DB_Connect* class, enabling user to establish connection with open database. After all data is transferred, the connection is closed. In the include/DB_data directory there is also situated a *config.php* file, storing database connection data. In order to be able to connect to database, each file has to include config.php with required clause.

6. Testing of the application

After creating an entire application, it is important to ascertain that the application is running correctly and in accordance with our initial plans and projects. In this chapter I will present all the ways of testing my application, briefly describing all of them.

6.1 JUnit testing

In Android it possible to use the regular JUnit framework, designed specifically for programming tests. It is a unit testing solution created for Java programming language. JUnit is used mainly for creating applications with test-driven development, in need for repeatable tests.

Generally JUnit tests are contained in a separate directory, located in the */app* module of the project. Thus, the code of the application and test code are separated and the project can be managed more easily. Additionally, in the test directory special classes have to be created in order to test all the features of the application properly. They can be divided accordingly to the classes they are testing, which is desirable [15].

JUnit's main features are Assertions, which are methods able to compare two values and basing on the result of the assessment they mark the test as passed or failed. There are several types of assertions e.g. *assertEquals, assertFalse, assertNotNull, assertSame, assertNotSame,* which let developers establish precisely, which line of code is causing the problem in the application.

In my project I utilize JUnit testing mainly for testing object classes, used for storing the data in my application. I also used this method to test my specific class, used for parsing data in JSON format and mapping it to Java objects. In this way I was able to test these functionalities thoroughly and using well known techniques.

6.2 Robotium

So far I have described frameworks used mainly for unit testing of the application. However, wanting to be able to test GUI as well, I found a perfect solution for my problem, which was Robotium framework.

Robotium is Android automatic test framework. It is characterized by a full support for native and hybrid applications. It lets developer write powerful test and black box UI test for Android. It requires minimal knowledge of the application. It handles multiple activities automatically, requiring minimal time to write test cases. Robotium tests inherit from class ActivityInstrumentationTestCase2, allowing developer to create test cases across Android activities.

The main and most important object in Robotium is Solo object, initialized in the setup method in the test class. Solo object represent an activity being tested and grants access to the activities objects and views elements. During creation of the tests, it is possible to map newly created View objects to the ones already existing in the activity, setting their value for testing.

There are several methods, created by Robotium, including asserting current activity and getting its fragment manager, mocking the clicking of the buttons of selecting a position from the list. It is also possible to mock usage of keyboard buttons, several view adapters including checkbox and spinners and finally asserting the value of onscreen prompts as well as the newly invoked activities [25].

Robotium is also equipped with a plugin written specifically for Android Studio, enabling black-box testing. It is possible to record the moves required to be done by the test cases and then the moves are mimicked on the mobile devices or emulator. If the result of the test is positive, the test is passed, otherwise the test is failed [24].

Using Robotium, I was able to test the functionalities in my application and more importantly the flow and changing of the activities and fragments in the fragment manager. It helped me with verifying the outcome of user's actions by testing the integration of several modules, featured in my application.

6.3 Real devices and emulators

During the development of my application I tested my application on the real devices. I performed it on two smartphones: LG F70 and Samsung GT-N7000 Note. The first one is operating on Android 4.4.2 while the second on 4.1.2. The main difference between these two models is the diagonal of the screen, which helped me in checking the outcome of my project on a slightly wider and narrower screen.

Being able to use only these two devices, I decided to use Android emulators, described briefly in the previous chapters. For testing my application I chose the virtual devices with the newest API possible. I used Google Nexus with API 22 (the newest at the time of writing this paper) and to my mind my application met requirements established by me.

7. ESN PW application outline

After installing the application, the user is presented with a logging screen. It consists of two empty fields to insert user's login and password. After filling them, user clicks the *Login* button as show on the Figure 35 and is forwarded to the main screen of the application, which I presented on Figure 34. Home screen enables user to browse the official social media profiles of the Warsaw University of Warsaw as well as provides menu bar with option to choose from. The menu bar is located in the upper part of the screen, containing buttons for opening the main menu, the settings screen or logging user out of the application.

Upon clicking on the menu list button, located on the left side of the menu as shown on Figure 34, the user is presented with a sliding, scrollable main menu occupying the left part of the screen, as on Figure 33. The main menu consists of a header with user's profile picture and menu positions, divided into categories.

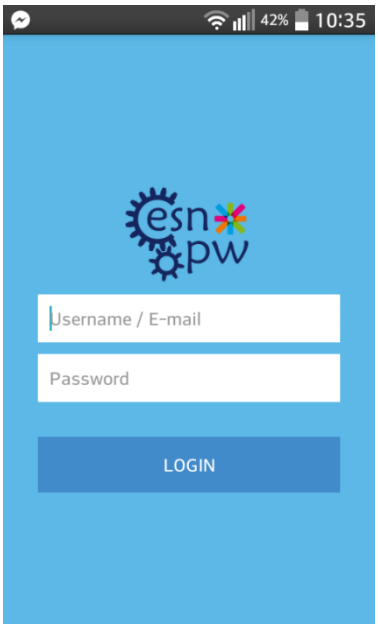

Figure 35 Login Activity

Figure 34 Home screen

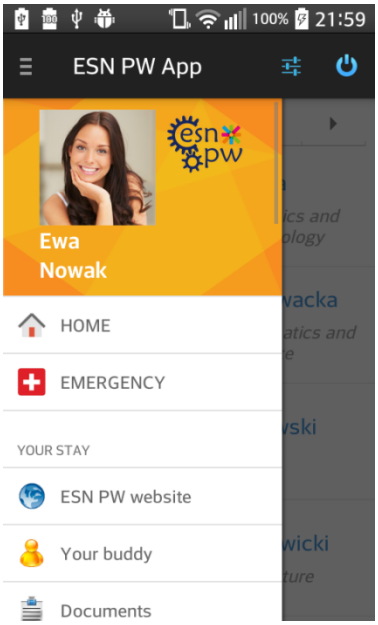

Figure 33 Main Menu

After clicking on the profile picture, a new screen to manage the picture is shown as on Figure 54. There are three buttons available to click. To take a new picture user has to click the "Take picture" button. It enables built-in camera and after user has accepted the taken picture it shows it on the screen. On the other hand, to upload the picture from the mobile device's memory user has to click "Choose picture" button. It opens a picture gallery and after choosing preferred picture, it is shown on the screen. To upload it on the server, user is required to click "Upload" button.

The next position of the menu, apart from *Home* option, enabling user to go back to the very first screen of the application, is *Emergency* (*see Figure 36)* showing the list of emergency contacts. It is possible to select one row by clicking on it, which immediately calls the selected emergency service.

Category “Your stay” features several options. *ESN PW website* position opens embedded Internet browser, showing the user the official website of the ESN section of the Warsaw University of Technology (*see Fig. 37*). It possible to browse it as any other website and the mobile device’s “back” button is used as the back button for the browser.

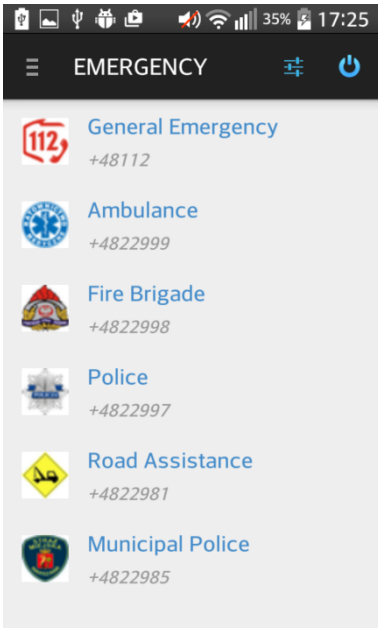

Figure 38 Emergency contacts

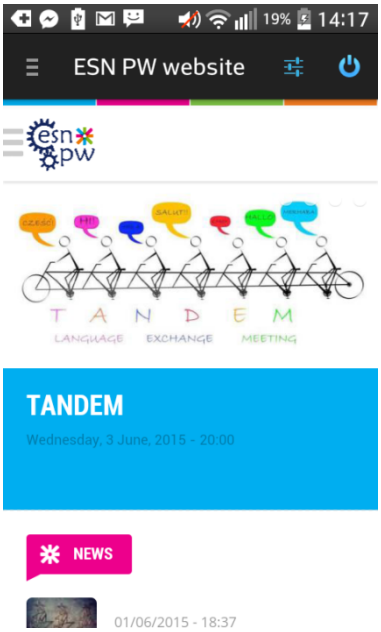

Figure 37 ESN PW website

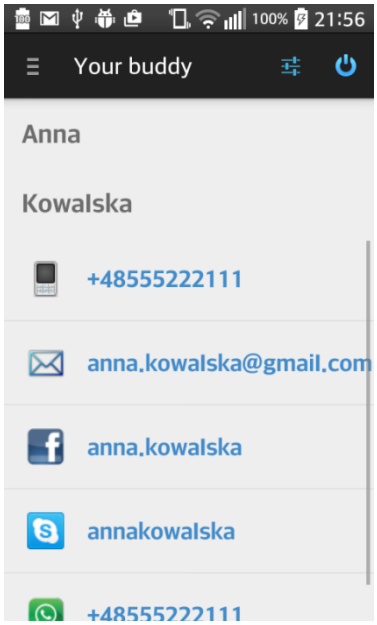

Figure 36 Your buddy screen

Your buddy option (*figure)* shows the screen with information about an assigned mentor (*see Fig. 38*). It is possible to choose from the available ways of contact, including e-mail and telephone along with popular applications such as Skype, Facebook and Whatsapp. After clicking on chosen contact, the appropriate application is opened and the mentor can be contacted.

Documents option shows the user all the information about the important documents that have to be taken for the international exchange. There are also listed some handy documents, possible to obtain in Warsaw or at the university (*see Fig. 40*).

Transportation option thoroughly describes the public transport in Warsaw. It is possible to read the written information (*see Fig. 39*) and, by swapping the screen, to see the list of important transportation website links (*see Fig. 41*). Upon clicking on one of them the user is redirected to the Internet browser, which shows a chosen page. *Rent a house* option is very similar to *Transport* option. It works in the same way, but it presents user information regarding the way of finding a cheap accommodation in Warsaw.

ToDos option shows a swipeable screen with tasks which ought to be done by the user after arrival to the exchange country and before leaving. Both lists can be changed easily by swiping the screen to the left or right. After clicking a position on the list it is marked as done or undone, depending of its initial state. If a task becomes done, it is marked by a tick in the checkbox next to its description and the task name becomes crossed out (*see Fig 43*).

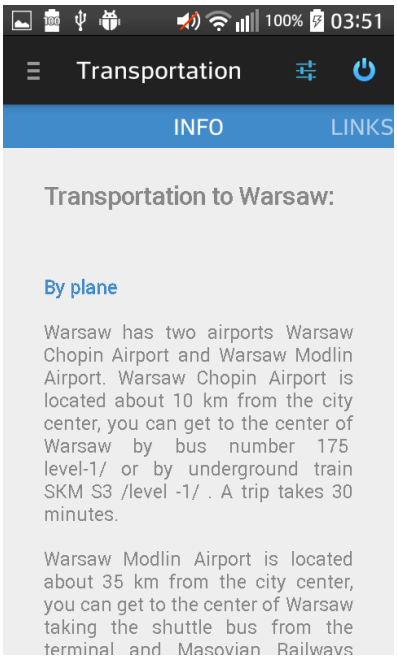

Figure 39 Transportation information

Figure 41 Transportation links

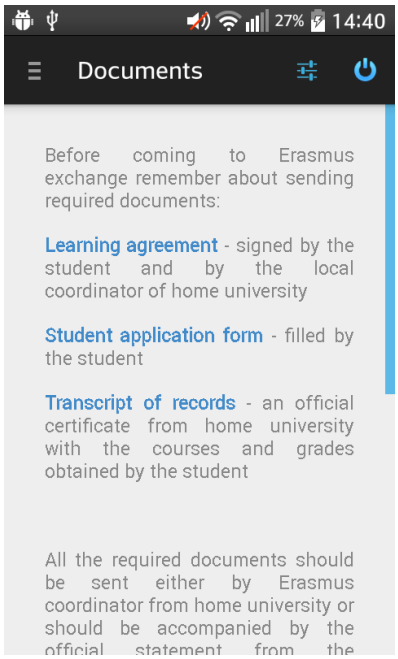

Figure 40 Documents information

FAQ option opens a screen with a list of questions. Upon clicking on one of them, , on the sliding part of the screen underneath it is shown the response to it. If the user clicks already an open question, the response part closes automatically (*see Fig 42*).

Useful apps option shows a gallery of applications handy for the user during the stay in Warsaw. After clicking on one of them, the user is transferred to the chosen application's Google Store Page, where it is possible to download the application (*see Fig. 44*).

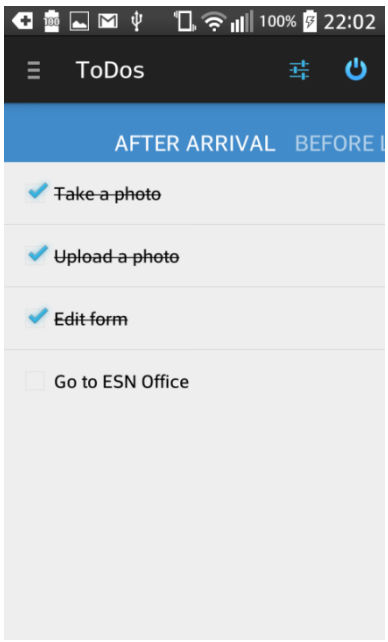

Figure 43 ToDos tasks

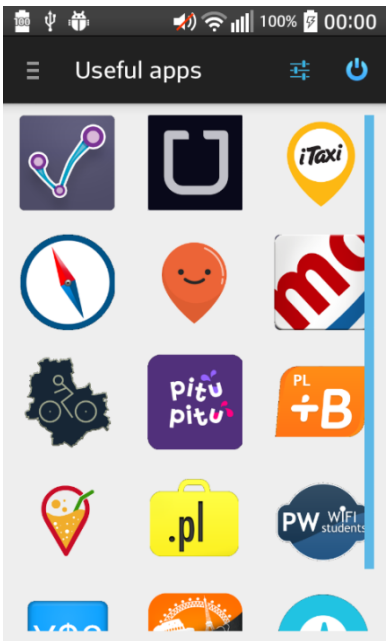

Figure 44 Useful applications

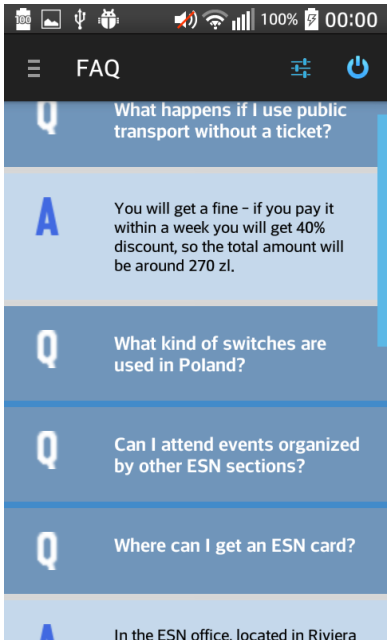

Figure 42 Frequently Asked Questions

The next category to choose from is “University”. *Your faculty* option shows the user’s faculty name as well as the links to faculty website, student’s grade and faculty’s e-mail. Clicking on the appropriate link opens Internet browser with a chosen page.

ESN Office option opens a screen with official ESN Office timetable. It is possible to check opening hours and the location of this office.

Campus map option shows the university campus map (*see Fid. 46*), with all the faculties and offices marked. The main faculty of a logged student in marked in yellow, the offices – in red and other faculties – in blue. User can click each marked place on the map, opening a window with more information regarding selected place (*see fig. 47*) – its picture, specific name and the location of student offices in the building.

PW website option (*see Fig. 45*) is similar to *ESN PW website* option, described earlier in the paper. The only difference is that is shows the official website of the Warsaw University of Technology in English.

PW online option shows a screen with images, suggesting social media on which it is possible to find the Warsaw University of Technology. After clicking on each image, it changes its appearance from grey to colorful and opens an official profile of the Warsaw University of Technology in the chosen application. If this application is not installed on user’s device, the profile is opened in the Internet browser.

Figure 45 Warsaw University of Technology online

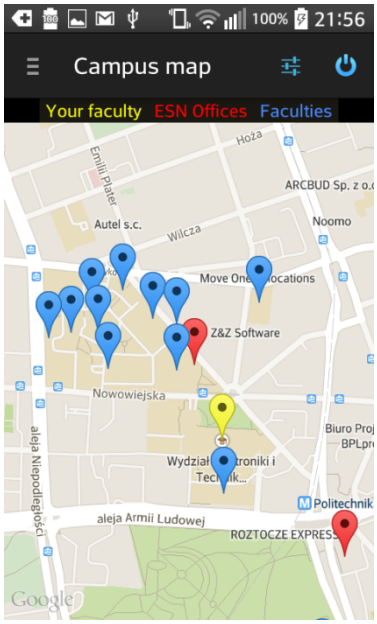

Figure 47 Campus map

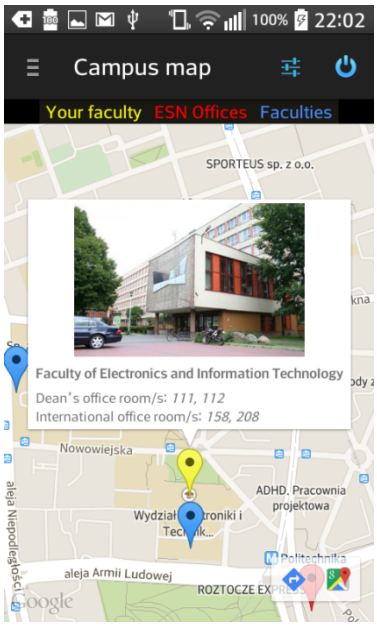

Figure 46 Campus map detail

The next category is “Entertainment”. The first option to choose from in this category is *ESN offers*. Upon selecting it, a new screen is shown with a list of ESN partners, who are offering discounts for students (*see Fig. 48*). After clicking on the interesting offer, it unfolds a hidden part of the screen, containing details about the chosen offer. Clicking an open offer results in the closure of its details.

ESN events option shows a list of events (*see Fig. 50*), gathered from various Facebook pages, belonging to ESN sections in Warsaw. After loading it on the screen, if user clicks one of the upcoming events the application redirects the user to the appropriate event’s page via Facebook app or website browser.

ESNFunMap option opens a map with marked locations. In the upper part of the screen there is a scrollbar from which the user can choose category for filtering the points (see Fig. 49). After clicking on one point its details are shown. In the upper right part of the screen there is also a button entitles “Your places”. After clicking it, the user is presented with list of the points added by him to the map. On clicking a particular point a new window appears (see Fig. 53), showing editable details of the point. It is possible to change its name, description and category. The user can save the changes by clicking “Save” button or completely delete a point by using “Delete”. If user wants to add a new point, he has to long click the screen and the new point will be created and marked in red. Unless user clicks it and edits its details, the point will not be saved in the database. User can also use a built-in Google feature to automatically discover his current location.

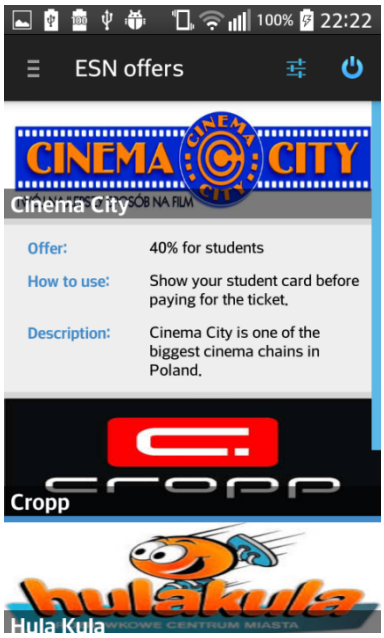

Figure 48 Offers

Figure 50 ESN events

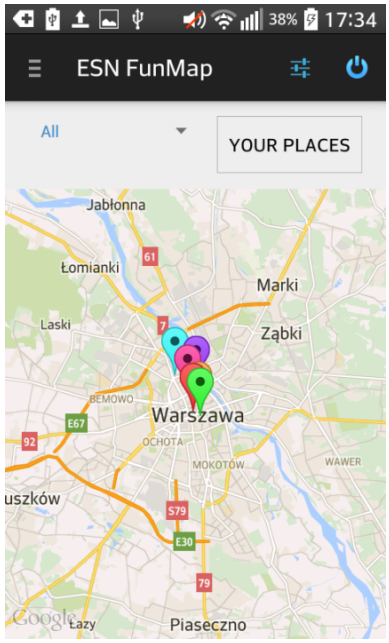

Figure 49 FunMap main view

Get in touch option shows a screen with the list of students (*see Fig. 52*), currently available in the system. In the upper part of the screen there is a search bar, enabling user to search by students’ names and faculties. After clicking on one person in the list, the new screen with selected person’s details is opened (*see Fig. 51*). It is possible to contact the person via the most popular application, which are opened automatically on this person’s profile.

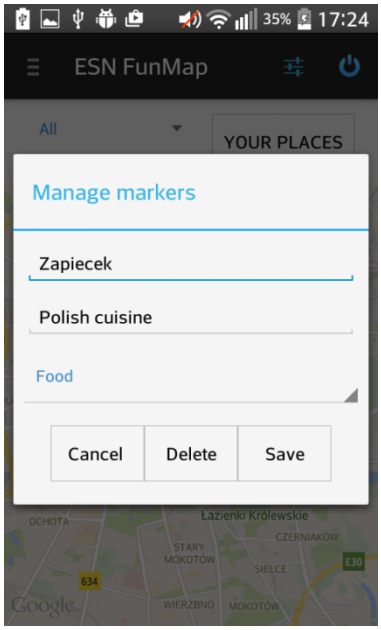

Figure 53 FunMap edit / add point

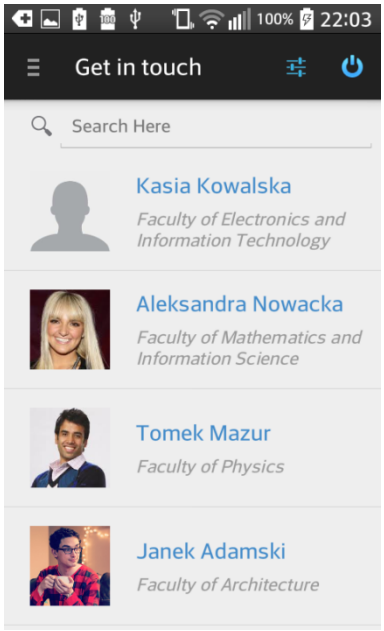

Figure 52 Get in touch list

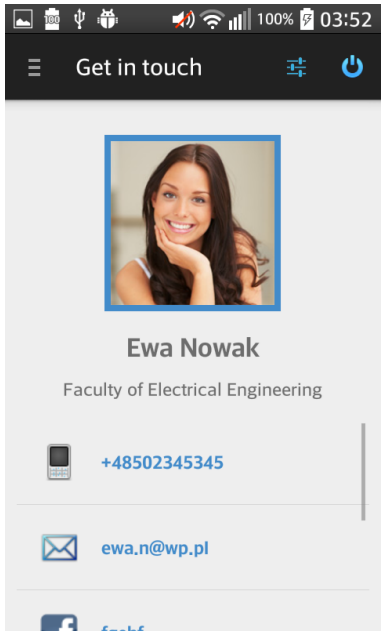

Figure 51 Get in touch a single person view

The last category is “Options”. There are two option attributed to this category. The first one is *Settings* and it opens a new screen with settings menu (*see Fig. 56*), well known from other mobile applications or even Android system settings. It is possible to change some user data, clicking on the position on the setting list. After selecting option *Change password* a new window appears (*see Fig 55*). To successfully change the password user has to insert the old password correctly and insert a new password twice. To accept the change user is required to click “OK” button. User is informed about the outcome of this action by a notification on the screen.

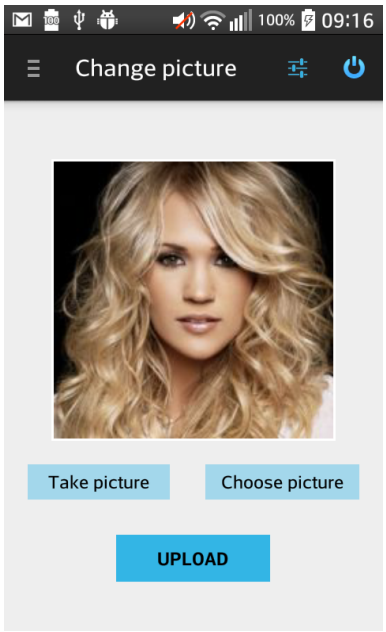

Figure 54 Change picture screen

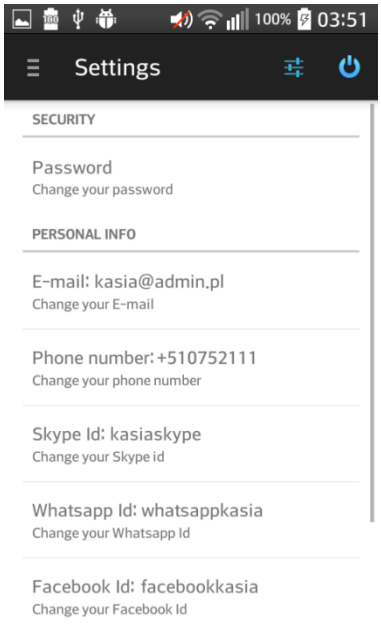

Figure 56 Settings screen

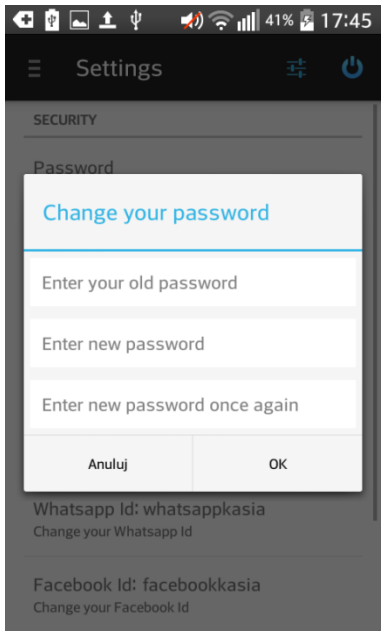

Figure 55 Change password screen

To change any other profile data user has to select an appropriate option from the list. A new window appears and user can either cancel the change by clicking “Cancel” button or accept it choosing “OK” button. The data is updated on the server. To change the visibility of user’s profile user has to click the visibility option and it changes its state from visible to invisible, depending on its initial state.

The last option in the menu is *Help*. It contains an envelope button, after clicking which the email application is opened, ready to send a message to the official ESN email address

8. Summary

The created ESN PW application is a result of a thoroughly conducted research, based on the author's experience and observations. It bridges the gap in the mobile application market, offering a dedicated solution which could serve foreign students, who decided to take part in the international exchange program at the Warsaw University of Technology.

I followed the application creation process, starting with collecting the ideas and requests coming from the student themselves. Then, I analyzed all the data and decided to introduce some of the proposed ideas to the possible extent. I maintained the connection with clients' side in order to satisfy their needs.

In my application I used the most modern and common technologies on the market, making the application as widely available and easy to use as much as possible. I used the newest solutions and programming patterns regarding the GUI presentation. In the project I implemented some of the most widely used libraries in order to improve the application performance and increase its simplicity.

8.1 Conclusions

Whilst the creation of my project I have learnt about the development process of a mobile application, trying to properly fulfill all the steps of the development I was introduced to during my studies. It was my first time designing application in these technologies, thus I can say that I certainly gained skills regarding Android mobile application development. I learnt about the basic programming patterns for this platform, basing on solutions provided by Android SDK library. On the other hand, using graphical and testing frameworks I learnt about simplifying the code by using the newest solutions available on the market

The main problems I encountered during the development process were mainly regarding programming of the application itself. As I mentioned earlier, I was working with Android platform for the first time and initially even very simple tasks took me a lot of time to implement. As I progressed, I learnt some of the basics and started working at a regular pace.

The design of the GUI was probably one of the most interesting aspects of the development, probably because I am very enthusiastic about computer graphics and art itself. However, I have to admit that one of the biggest issues was probably the fact that so many devices operate on Android system and it is extremely hard to design an application, which would look presentable on the majority of them.

As a whole, I can say that I really enjoyed working on this platform and I would really gladly study the development of the applications for other popular mobile platforms such as iOS or Windows Phone.

8.2 Further development

The system being a dedicated solution for the Warsaw University of Technology and its foreign students could be and should be developed in order to be introduced to the users. There are plans of presenting the system to Erasmus Student Network, which could result in the acceptance and improvement of the system.

However, as it was noticed on one of the seminars, the system is so flexible that it could also help Polish students, coming to study at the Warsaw University of Technology. A big part of the functionalities is based on the information about the university or the city of Warsaw

therefore without doubt it could be convenient for those, coming to Warsaw from other cities. Moreover, it could also be possible to create a plain, non-dedicated version of the system and share it with other universities, creating an original solution for foreign students in Poland. This version however would require creation of WWW client for system administrator to manage some of the information available in the system. This client could be developed using recently web development techniques as Spring, Java and AngularJS for client side.

There are still some features which could be developed or improved, including:

- WWW client for system administrator
- Using NoSQL database for storing the data regarding recommended places
- Introducing preference and voting system for the recommended places
- Improving the effectivity of the server side of the application
- Creating a common, non-dedicated version which could be changed and personalized for other universities or Polish students
- Introducing functionality, which could help foreign students in organizing trips, parties and gathering, informing the users about meeting via e.g. push notifications
- Providing better synchronization of the online and offline data using Services or other Android based solutions

9. References

1. *Apppszoom.com*, "SPQE Erasmus Roma", 2015 [online] Available at: http://www.appszoom.com/android_applications/communication/spqe-erasmus-roma_cwulq.html [Accessed June 2015]
2. Burnette Ed, *Hello Android: programowanie na platformę Google dla urządzeń mobilnych.,* Trans. Krzysztof Sawka. Gliwice: HELION, 2011. Print.
3. Darwin Ian D., *Android. Receptury,.* Trans. Tomasz Walczak. Gliwice: HELINON, 2013. Print.
4. *Erasmus Student Network*, "ESN Annual Report", 2015 [online] Available at: http://esn.org/annual-report [Accessed June 2015]
5. *Erasmus Plus*, "Przegląd statystyk", 2015 [online] Available at: http://www.erasmus.org.pl/sites/erasmus.org.pl/files/Erasmus_statystyki_07-2014_1.pdf [Accessed June 2015]
6. *Facebook Inc.*, "Graph API Overview", 2015 [online] Available at: https://developers.facebook.com/docs/graph-api/overview/ [Accessed June 2015]
7. *Freeapp.com*, "Erasmus Phone", 2015 [online] Available at: http://freapp.com/apps/android/com.dani1913.erasmusphone/ [Accessed June 2015]
8. *Genymotion,* "Genymotion", 2015 [online] Available at: https://www.genymotion.com/#!/developers/user-guide#overview [Accessed June 2015]
9. *Google Inc.,* "Android Studio Overview", 2015 [online] Available at: https://developer.android.com/tools/studio/index.html , [Accessed June 2015]
10. *Google Inc*., "Transmitting Network Data Using Volley", 2015 [online] Available at: https://developer.android.com/training/volley/index.html [Accessed June 2015]
11. *IDC.com*, "Smartphone OS Market Share, Q1 2015", 2015 [online] Available at: http://www.idc.com/prodserv/smartphone-os-market-share.jsp [Accessed June 2015]
12. *International Exchange Erasmus Student Network*, "List of ESN sections", 2015 [online] Available at: https://esn.org/sections [Accessed June 2015]
13. *Iwco.com*, "Online vs. Offline Data: Which is More Effective?", 2013 [online] Available at: http://www.iwco.com/blog/2013/12/13/online-vs-offline-data/ [Accessed June 2015]
14. Lehtimatki Juhani. *Podręcznik dla projektantów Android UI.,* Trans. Mikołaj Szczepaniak. Gliwice: HELION, 2014. Print.
15. Komatinen Satya, et al. *Android 3: tworzenie aplikacji,.* Trans. Krzysztof Sawka. Gliwice: HELION, 2012. Print.
16. Neil Theresa, *Mobile Design Pattern Gallery.* Sebastopol: O'Reilly, 2014. Print.
17. *Picasso*, "A powerful image downloading and caching library for Android", 2015 [online] Available at: http://square.github.io/picasso/ [Accessed June 2015]
18. *Play.google.com*, "Erasmus ESN Events", 2015 [online] Available at: https://play.google.com/store/apps/details?id=ch.epfl.xchange [Accessed June 2015]
19. *Play.google.com*, "ESN Aveiro", 2013 [online] Available at: https://play.google.com/store/apps/details?id=com.lourencomarinho.esnaveiro [Accessed June 2015]
20. *Play.google.com*, "ESN Lund", 2014 [online] Available at: https://play.google.com/store/apps/details?id=com.ima.fantastic.esnlund9 [Accessed June 2015]

21. *Play.google.com*, "ESN Messina", 2013 [online] Available at: https://play.google.com/store/apps/details?id=com.n3tsoft.esnmessina [Accessed June 2015]
22. *Play.google.com*, "ESN Roma ASE", 2014 [online] Available at: https://play.google.com/store/apps/details?id=it.primitera.esnromaase [Accessed June 2015]
23. *Play.google.com*, "ESN Sea Battle", 2015 [online] Available at: https://play.google.com/store/apps/details?id=com.esn.seabattle [Accessed June 2015]
24. *Robotium*, "Test Android Calculator APK- Black Box Testing", 2015 [online] Available at: http://robotium.googlecode.com/files/TestAndroidCalculatorAPK-BlackBoxTesting-V2_0.pdf [Accessed June 2015]
25. *Robotium*, "Test Android Calculator - White Box Testing", 2015 [online] Available at: http://robotium.googlecode.com/files/TestAndroidCalculator-WhiteBoxTesting-V2_0.pdf [Accessed June 2015]
26. *Sensortower.com,* "Erasmus ESN Cordoba", 2014 [online] Available at: https://sensortower.com/android/us/signlab/app/erasmus-esn-cordoba/com.signlab.erasmusevents [Accessed June 2015]
27. *TextJustify*, "TextJustify Wiki page", 2015 [online] Available at: https://github.com/bluejamesbond/TextJustify-Android/wiki/1-%C2%B7-Home [Accessed June 2015]
28. *TikTalk.com*, "Better Buttered: A look at Butter Knife, a View "injection" library for Android", 2013 [online] Available at: http://www.tikalk.com/better-buttered-look-butter-knife-view-injection-library-android/ [Accessed June 2015]
29. *Vandelaydesign.com*, "12 'Must-Know' Advantages of PHP", 2012 [online] Available at: http://www.vandelaydesign.com/advantages-of-php/ [Accessed June 2015]
30. Wantoch-Rekowski Roman. *Android w praktyce. Projektowanie aplikacji.* Warszawa: PWN, 2014. Print.