UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Institute of Computer Science
Information Technology speciality

Martin Vels

Development of a mobile pharmaceutical information environment

Bachelor Thesis (6 EAP)

Supervisor: Ulrich Norbisrath, PhD
Supervisor: Siim Uibokand, MSc (Apteekide Infoteholoogia OÜ)

Author: ..... "....." May 2011
Supervisor: ..... "....." May 2011
Supervisor: ..... "....." May 2011

Allowed to defense
Professor: ..... "....." May 2011

TARTU 2011

Contents

Acknowledgements

First, I would like to thank my supervisors, Siim Uibokand and Ulrich Norbisrath. I would like to thank Siim for giving me the idea for the thesis and helping to test the application. I would like to thank Ulrich for his advice on writing the thesis and his time spent on giving me the feedback for the thesis.

I would also like to thank Aarne Jaansalu for helping me out with the mobile application icon and pharmacy logos shown in the application.

Lastly, I am also very grateful to my family, especially my girlfriend Helina and son Martin for always supporting me.

Introduction

For almost a decade the medicine and pharmacy database/search engine `http://www.raviminfo.ee` has been available on the web. I am the author of that web-site and it has been nice to see that the popularity of the site is constantly growing. More and more people are using it as the Estonia's best source for getting information about drugs and other products sold in pharmacies.

As almost every person is using mobile phone and more and more people are using smart phones, the only logical continuation for developing that site further was to create a native mobile application. As pilot platform we chose iOS - Apple's operating system for all the mobile devices like iPhones, iPod Touches and iPads. The main reason for developing the application for iOS was my interest in smart phone niche and in Apple devices in particular.

Further more, as I was also leading and developing similar project (LasaLara learning environment for the iPhone) for the course Software Project, it seemed logical to go on with the chosen path and create another useful application that every person could use.

There are two main reasons for using the original raviminfo.ee web-site. First one is to get information about pharmacies, like where they are located and what are their opening hours and other contact information. Second one is searching for the medicines from the pharmacies together with the price information as well as all the detailed information about these medicines. Implementing these two functionalities was also the main scope for the mobile application.

However, since mobile devices are more limited, especially in screen sizes compared to regular computers with large screens, the main difference is the data presentation. I had to make compromises for fitting all the needed data into small screen of the mobile phone. In addition, as mobile devices often have GPS built-in this gave an interesting possibility to be able to show users information about the nearest pharmacies, to make the user experience even more comfortable and pleasing.

This thesis is divided into four chapters. The first chapter gives an overview why such a mobile application could be useful and describes some of the similar mobile applications available on iOS. The second chapter contains the requirement analysis for the application. In the third chapter the design and implementation of the application is described. In the fourth chapter a short end-user manual is available to give a brief overview on how to use the application.

Chapter 1

Related work and theoretical background

1.1 The need for accessing information everywhere

With constant evolution of mobile handheld devices like smart-phones, more and more people are using them for tasks that were previously available on personal computers only, like listening music, watching videos, playing games and accessing the Internet. Fast wireless networks are getting very common and more important cheap for everybody. Free WiFi is very widespread in different cities over Estonia, mainly in the centers of the cities but also in a lot of stores, restaurants and office buildings. Different cheap data plans are available from all of the 3 mobile providers (EMT, Elisa, Tele2). It is possible to get an unlimited 3G data plan for ~5€/month from all of these providers. With all this, it means that people are spending more time away from their computers, doing other useful things, but are still very interested in accessing information from wherever they happen to be, using their smart-phones. Using regular web-pages with the small screen of the smart-phone is not comfortable. Therefore it is common that special applications are provided by the web-page owners to give their users a better experience in retrieving the information they offer.

The main motivation for creating a special mobile application for accessing the info available on raviminfo.ee web-site was to offer a better user experience for smart-phone owners. To be able to search for pharmacies as well as medicines available in these pharmacies using special application designed especially for iPhone is more convenient than browsing the regular raviminfo.ee web-page with web-browser of the mobile phone.

1.2 Similar projects

The raviminfo.ee contains the information for over 250 pharmacies all over Estonia and these pharmacies are the ones using the special software (RAX and its successor Noom) to send their stock-level information into raviminfo-server. There is another similar web-site in Estonia called apteegiinfo.ee that contains information for pharmacies that are using different software (Hansasoft). There are about 120 pharmacies with price lists available in apteegiinfo.ee. There is also a mobile version of that web-site available, but unfortunately it is virtually unusable on iPhone because the page is rendered as a very narrow ribbon in the center of the page. To be able to see what is actually

displayed on screen it is needed to manually resize the page using pinching gesture every time the new page is loaded.

Outside Estonia there are several mobile applications available from different companies that are mainly concentrating on showing pharmacy locations on the map as well as displaying the contact information like phone numbers and open hours. However there are not similar databases with pharmacies together with their near real-time price lists available online.

I was searching for mobile applications for iPhone, that would be similar to the one I was planning on develop. It was looking for medical and pharmacy applications. There are dozens of different medical applications available in App Store, Most of the medical applications are quite expensive and containing more of the medical databases than pharmacy contacts and product databases available in these pharmacies. So the following iPhone applications were found from the Apple App Store that were built to help users to locate pharmacies nearby:

- CVS Pharmacy
- PulsePharmacy
- PharmacyIrl
- Walgreens
- Apotheken

I will give a short overview of these applications and will provide some screenshots for every one of these applications. I am also giving my opinion for every screenshot about what seems to be nice and what does not look too good. I am using the knowledge gathered from Apple HIG document to give my opinion about the applications. There are following major principles in HIG that should be followed in creating a great mobile application:

- Aesthetic integrity - it is a measure of how well the appearance of the application integrates its function.
- Consistency - it means that application should use system-provided controls, views and icons correctly and that all the terms and icons are meaning the same things throughout the application.
- Direct Manipulation - it means that user should be able to see their actions have immediate, visible results, that they could rotate the screen to affect the objects on screen and to be able to manipulate the objects on the screen with gestures.
- Feedback - it means that every action that user does should have some kind of feedback, for example an animation, some kind of alert or progress bar.
- Metaphors - it means that objects and actions in an application should be metaphors for objects and actions in the real world. The best known example for such a metaphor is a folder: people can put things in folder in real world as well as in computer world.
- User Control - it means that user should be the one who controls the application not vice versa. If the controls and behavior of the application is familiar to user, then he feels more in control and gets better result with the application.

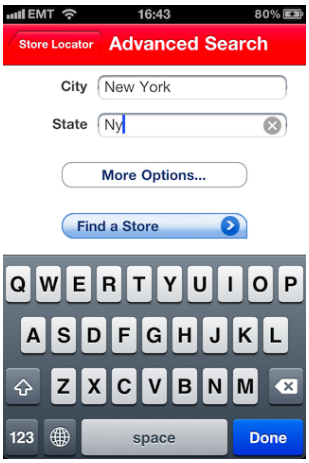

Figure 1.1: Store locator search page contains possibility to search manually by the city and state as well as using the GPS-coordinates from the phone. Interface does not look polished, field and buttons are not with consistent design.

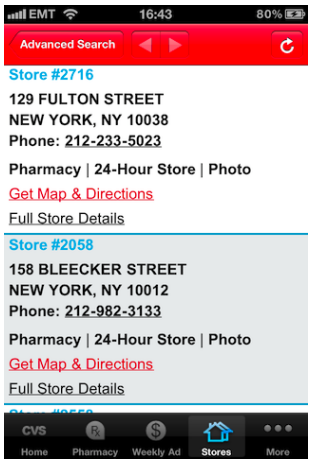

Figure 1.2: Search result page, where pharmacies are shown. Full name, address, phone and links to map and full details are shown. The design of this view looks quite random and hard to grasp.

1.2.1 CVS Pharmacy

This is the official iPhone application from the second largest pharmacy chain in USA (after Walgreens). The application has quite extensive functionality, but the usability is not very user-friendly. The application has the possibility to search pharmacies by address or by GPS-coordinates. Opening hours, contact phone and full address are shown in detailed view of the pharmacy. A drug database is available with detailed information about every drug. There is also a nice feature for registered users to be able to refill the prescription using the application. Usability does not look very good, because standard Cocoa Touch provided elements are not used, and the application is just showing web-pages inside the application, which probably made the application development a lot easier for multiple platforms. However the cost of this shortcut is less user friendly application. More detailed information is located under the screenshots in following figures: 1.1, 1.2, 1.3, 1.4, 1.5, 1.6.

1.2.2 Pulse Pharmacy

Like the CVS Pharmacy, the Pulse pharmacy application is also created by the pharmacy chain to promote their pharmacies. Application contains store locator, online shopping, featured products and loyalty registration parts. Design of the application does not use any of the standard UI elements from Cocoa Touch which makes it harder to learn for first time users. More detailed explanations for every part of the application can be seen next in the table with screenshots. More detailed information is located under the screenshots in following figures: 1.7, 1.8, 1.9, 1.10, 1.11, 1.12, 1.13, 1.14.

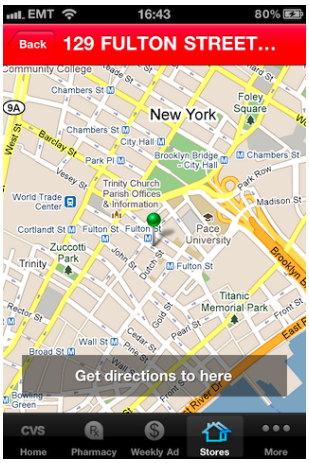

Figure 1.3: Standard Google provided map is used to show the location of the selected store. It is a good idea to have possibility to get directions from current location of the user to the selected pharmacy.

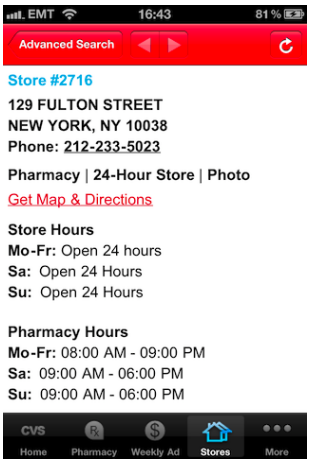

Figure 1.4: Full details of a specific pharmacy are shown with full name, phone number, address and open hours. Also link to map and directions are shown.

1.2.3 PharmacyIRL

Nycomed Irish Pharmacy Finder provides users with comprehensive information about heartburn and its treatment. This application also contains a listing of almost 1,600 pharmacies across Ireland, including pharmacy name, address, telephone number and GPS coordinates. The user can search pharmacies by name, around me or by location. Detailed information about the night pharmacies is also available. More detailed information is located under the screenshots in following figures: 1.15, 1.16, 1.17, 1.18, 1.19, 1.20.

1.2.4 Walgreens

Application from the biggest pharmacy chain in USA. Application contains possibility to order refills using the camera of the phone to scan the barcode of the package. Also nearest store locator using GPS positioning is available. A weekly Ad section gives an overview of the in-store savings available. There is also a functionality for creating a shopping list available. More detailed information is located under the screenshots in following figures: 1.21, 1.22, 1.23, 1.24, 1.25, 1.26, 1.27, 1.28, 1.29, 1.30, 1.31, 1.32.

1.2.5 Apotheken

Simple and straight forward application for finding pharmacies in Austria. It shows just pharmacy locations as map and list. Detailed information about the pharmacies is also available. Clean and polished application. More detailed information is located under the screenshots in following figures: 1.33, 1.34, 1.35, 1.36.

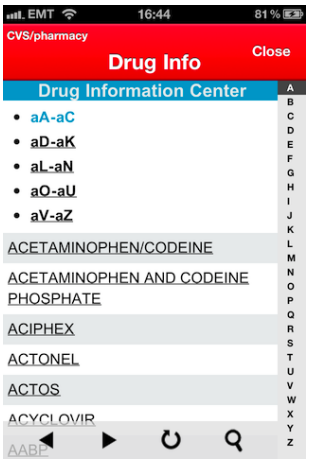

Figure 1.5: Drug information database has extensive list of drugs available. First letter of the active substance can be used to jump to specific names. Design is not using standard Cocoa Touch tables but a html-page instead. This makes the design look a bit unusual for iPhone users. Real-time search-bar is missing, which is a quite widely-used feature to filter out items from the long list. Navigation bar has no regular buttons like "Back" to move back one step in view hierarchy. No tab-bar is shown beneath the table, like it was on the other views of the application.

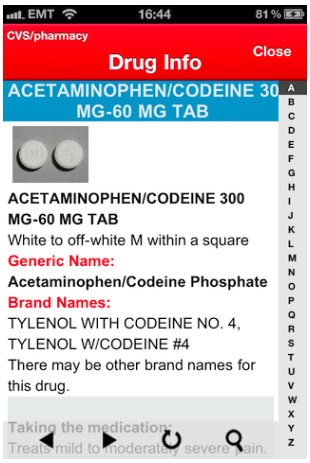

Figure 1.6: Detailed information about a certain drug. It is nice that it has image of the pills, so an user can identify the pills visually as well. Information has subsections for certain parts like generics and brand names. However the problems with the design are similar like were described above, there is no "Back"-button on navigation bar and tab-bar is missing.

1.2.6 Conclusion

To conclude the overview of these applications I will list the things that I liked and did not like in following lists:

1.2.6.1 Good

- Standard UI elements like tab-bars, navigation controllers, buttons, tables and views were used.
- Both the map and list view shown for pharmacies.
- Possibility to filter info from tables with search-box.
- Special icons designed for map annotations and actions inside application.
- Distances from user to pharmacy shown.
- Scan-code reading with the camera of the phone.

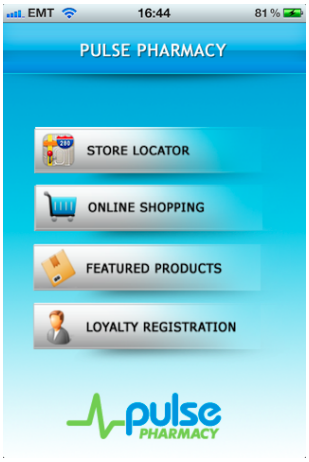

Figure 1.7: Initial screen of the application with all four possible sub-modules displayed: store locator, online shopping, featured products and loyalty registration. Loyalty program is a free program which gives users bonus points for purchases and discount vouchers after certain amount of points are collected.

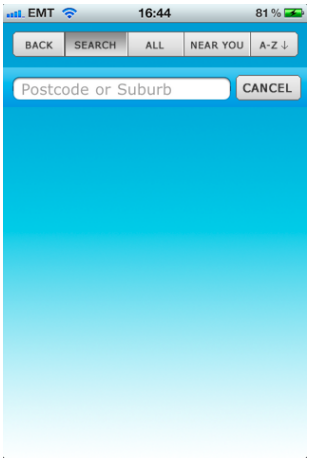

Figure 1.8: Pharmacy location can be searched by postal code or address entered manually. As can be seen, no standard navigation nor UI elements are used.

1.2.6.2 Not so good

- Non-standard UI elements used, users have to think how the interface works.
- Info shown in web-view, meaning that the application is just a shell for a web browser.
- User is redirected to a website which is designed for regular computer.
- Distances shown from the user location to pharmacies are not realistic.

1.3 Roundup

In this chapter I described why there is a need for mobile application that could be used to see pharmacy locations and available item information in these pharmacies. Some of the similar iOS applications were reviewed to get a better understanding on what is already done and what pitfalls to avoid during developments. In the end I compiled a short conclusion of good and bad things found during the analysis of applications.

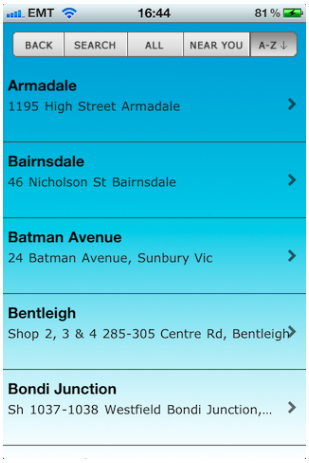

Figure 1.9: All the pharmacies are shown in the list sorted alphabetically. No search box is given to filter out pharmacies from the list. There seems to be a layout issue with too long address which overlaps with the disclosure arrow icon.

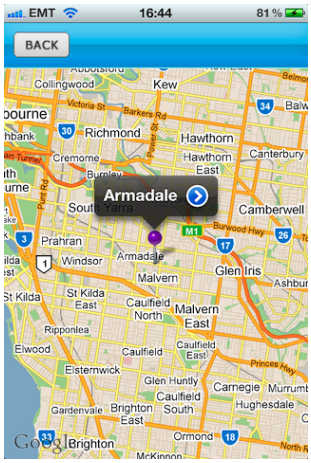

Figure 1.10: Specific pharmacy shown on the Google map. It is standard way to show locations and should be familiar for users. The annotation could have more details written in it, but it is good that disclosure icon is used for expressing the more detailed information available.

Figure 1.11: Online shopping view redirects to company's web-page. This makes the online shopping functionality virtually unusable, because all the problems with small screen and web-page designed for regular PC-screens are present.

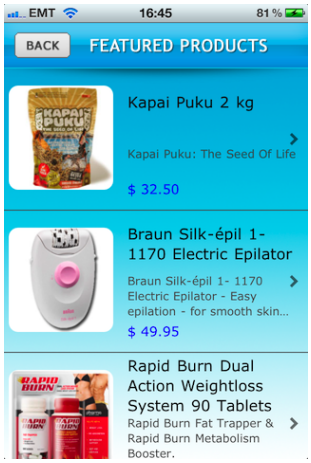

Figure 1.12: Featured products view shows some of the products that are currently promoted by the chain. Again, no possibility to filter these products using a search-bar.

Figure 1.13: Detailed information view for a specific featured product. No possibility to see what are the locations where this item can be purchased nor the specific price for this item in any of the stores. Only short information given about the price may be varying in different stores, which makes the price info not very useful for the users.

Figure 1.14: View where potential loyal customers can register and provide information about them to the store. No description is available inside the application, what this loyalty registration means for the user or what kind of benefits it will give. User has to go to the company web-site to read more info about the program where it is called "Customer Rewards Program" instead.

Figure 1.15: Initial splash-screen of the application, used quite a lot in application during the initial load of the data so user should not watch empty screen. Also gives a great possibility to display company logo to user in more natural way than using banners inside application.

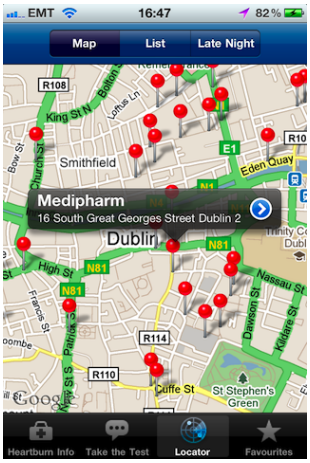

Figure 1.16: Map-view of the pharmacies. Standard Google map is used and standard Cocoa Touch user interface elements like navigation bar above as well as tab-bar below the screen are used. Clean and familiar way to display information to users.

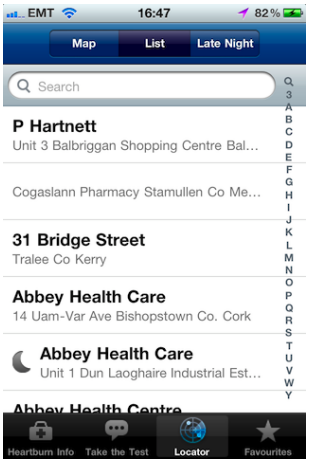

Figure 1.17: List view of the pharmacies. Standard table-view with quick-search and alphabet index are shown. Again, using standard and familiar components for displaying info, is making the application look clean and polished.

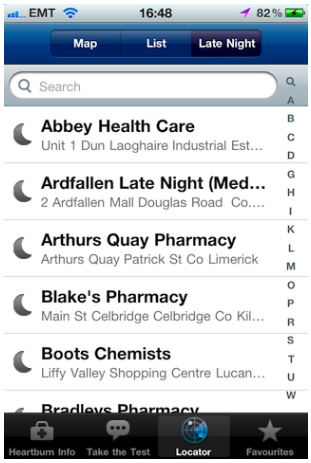

Figure 1.18: Like the list view of the pharmacies, the late night view has the same features, but it makes it more convenient for the user to filter out only the pharmacies that are opened all night. Certainly a great feature to make an application more user friendly.

Figure 1.19: Detailed information about a specific pharmacy. Using buttons for phone number and locations on the map makes it very easy to understand for users. Clear and explicit way to show that the pharmacy is opened all night is used.

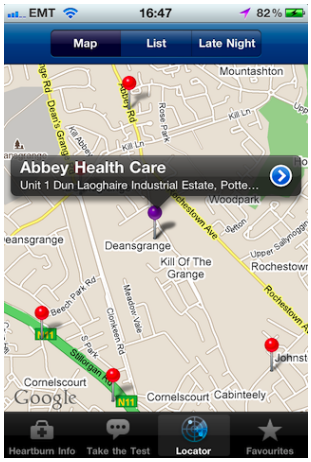

Figure 1.20: When user taps on the purple "Show on Map" button in the pharmacy detail-view, the map-view will be opened with the pharmacy location shown with purple pin on it.

Figure 1.21: Opening/navigation screen of the application. Nice icons are used. The overall look is clean and polished. Only problem seems to be the non-scalability of this screen. For example if there is a need to add one or several new sub-views, then all the icons have to be redrawn to make them fit nicely. also it is hard to add just one new item here, as the current 3 x 3 icon view will not be available any more.

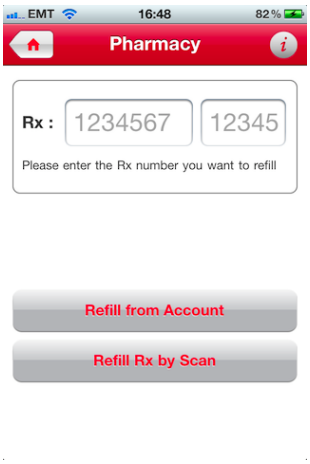

Figure 1.22: A little confusing view, because under the pharmacy user probably expected to find a list of pharmacies or a map, but instead prescription number field is shown for refilling. Also it is confusing because there was "Refill by Scan" icon on the opening screen, now the same functionality is also available in Pharmacy view as a button.

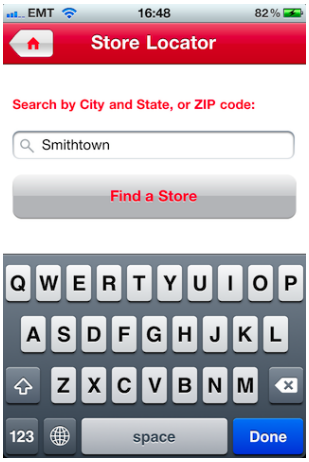

Figure 1.23: In Store Locator, it is possible to search for stores by city and state or by zip code. No surprises here, view is clean and understandable, maybe it would have been nice to see some way to locate a store using GPS-coordinates.

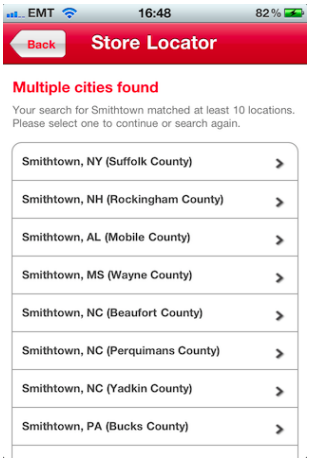

Figure 1.24: Search results from Store Locator. Clean and polished look, nice to see all the available locations with the same name together so the correct one can be located.

Figure 1.25: All Walgreens stores are shown on the standard Google map. It is a nice touch that special icons are used instead of standard pins to show the positions of the stores.

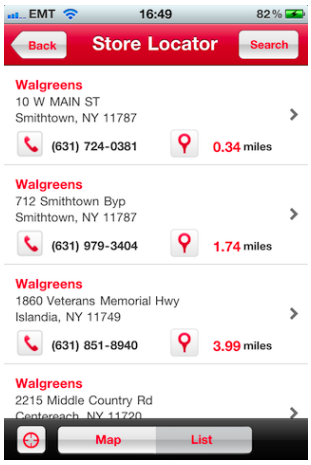

Figure 1.26: Pharmacies shown as a list. Standard Apple-provided table is used, making the view look clean and polished. It is a nice touch that special icons are designed for phone and distance. The distance shown is a bit confusing. If GPS-coordinates are used, then it does not make sense that the user in Estonia is only 0.34 miles away from the store in New York.

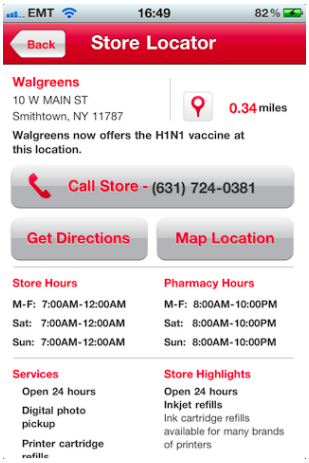

Figure 1.27: Detailed information about a specific pharmacy. Very well designed view. All the needed info is fit on one small screen of an iPhone. Address together with explicit phone number and call button are shown first. Open hours and additional services are shown below.

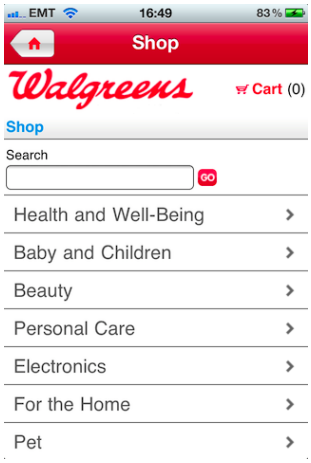

Figure 1.28: Categories of the items available in the stores. Categories are not alphabetically sorted, making it a little confusing to search for a specific category. Non-standard search-box is used.

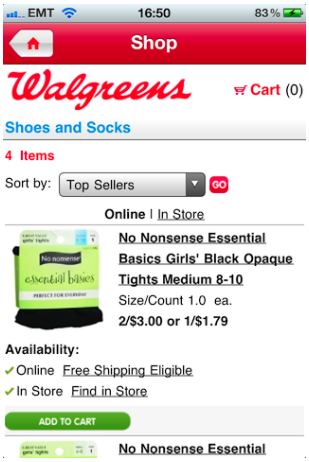

Figure 1.29: Result of the specific category. It can be seen that no standard Cocoa Touch components are used any more, but special web-page result are displayed. Screen looks quite random and detecting important information is not very easy.

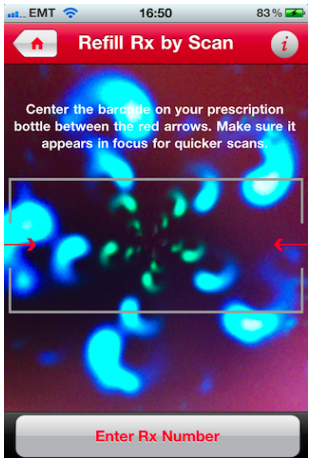

Figure 1.30: Interesting feature allowing the user to scan barcode from the package. Definitely a nice-to-have feature that makes it easy for users to get quick information about some specific drug using the camera of their phone instead of entering the name of the drug manually with a not-so-comfortable virtual keyboard.

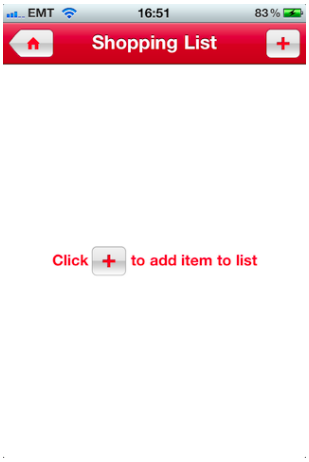

Figure 1.31: Again nice feature that allows user to create the shopping list before heading to pharmacy.

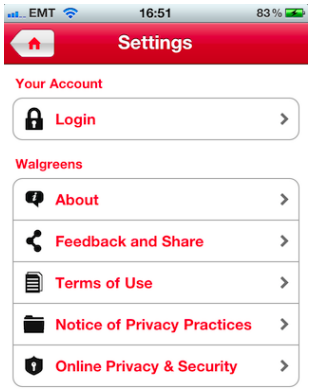

Figure 1.32: Settings view of the application. It is a bit unusual to have the settings directly inside the application but not in the Settings application like it usually the case with iPhone applications. However, the execution of the view is nice and clean.

Figure 1.33: Map view of the pharmacies. Special icons are used instead of standard pins. Night-pharmacies are using an additional moon-icon which make them nicely distinct from the rest.

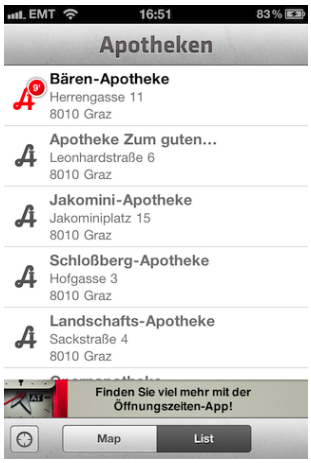

Figure 1.34: The list-view of the pharmacies. It would be nice to have a search-bar functionality available here to quickly filter out pharmacies by name.

Figure 1.35: Detailed view of a specific pharmacy. Information is presented in a clear and explicit way. Maybe the phone number should have been in the first part of the scree instead of the second half which can only be accessed after scrolling down.

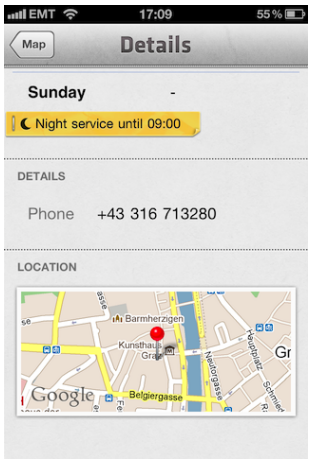

Figure 1.36: Second half of the detailed view of the specific pharmacy. Nice touch is the small image from the map so that the location of the pharmacy can quickly be seen from the detailed view.

Chapter 2

Requirement analysis

As there already was a web-based version for raviminfo.ee, it was only natural that the requirements for the mobile version were similar. However some of the differences were also present and the following chapter will give an overview of the requirements of the application that will be created using the FURPS model.

2.1 Functional requirements

Functional requirements are written down as use cases. For every use case special template will be used which consists of following points:

- Use case number
- Priority
- Use case name
- Description of a situation
- Actor
- Pre-conditions
- Trigger
- Result expected
- Main scenario
- Alternative scenario(s)

2.1.1 Summary of the use cases

1. Browsing the pharmacies using the map
2. Reloading the pharmacy data from the server
3. Positioning the map view to user's physical location
4. Browsing the list of pharmacies

5. Viewing detailed information of the pharmacy
6. Searching for the medicine by name
7. Viewing the list of pharmacies selling the chosen medicine
8. Viewing the detailed information for the chosen medicine
9. Viewing the package information sheet for the chosen medicine
10. Viewing the item image
11. Viewing the diagnosis-related information for the chosen medicine
12. Browsing the ATC-list (Anatomical Therapeutic Chemical)

2.1.2 Detailed use cases

2.1.2.1 Browsing the pharmacies using the map

Use case number	1
Priority	1
Use case name	Browsing the pharmacies using the map
Description	User sees the map with nearby pharmacies on it.
Actor	User
Pre-conditions	Application is running.
Trigger	User taps on the “Pharmacies” tab on the tab bar.
Result expected	Map is shown with pharmacies marked as pins on it.
Main scenario	1. User taps on “Pharmacies”-tab on tab-bar 2. Map-view is activated 3. Map is displayed with pharmacies show on it as pin-icons
Alternative scenario	3.1. If there is no Internet connection, no map is shown and user is notified about the missing Internet connection 3.2. User will be on the Map-view without Map shown on it

2.1.2.2 Reloading the pharmacy data from the server

Use case number	2
Priority	3
Use case name	Reloading the pharmacy data from the server
Description	User can reload pharmacy data from the server
Actor	User
Pre-conditions	Application is running and user is in pharmacy map of list view
Trigger	User taps on the "Reload" button which looks like a round arrow.
Result expected	Data is reloaded from the server and fresh info shown on the map or list.
Main scenario	1. User taps on "Reload"-button 2. Activity indicator is shown while data is received from the server 3. Fresh info is shown on the map or list view.
Alternative scenario	3.1. If there is no Internet connection, no map is shown and user is notified about the missing Internet connection 3.2. User will stay on the map or list view of the pharmacies

2.1.2.3 Positioning the map view to user's physical location

Use case number	3
Priority	2
Use case name	Positioning the map view to user's physical location
Description	User can make the map view to show his position on using the coordinates of the mobile device
Actor	User
Pre-conditions	Application is running and user is in pharmacy map view
Trigger	User taps on the "Position" button which looks like a circle and cross on it.
Result expected	Map view is animated to user's location and blue pin is dropped on the map indicating the the location of the user
Main scenario	1. User taps on "Position"-button 2. Map view animates to user's location 3. Blue pin is shown on the map indicating the position of the user.
Alternative scenario	2.1. If coordinates can not be determined, user will get notification about the situation 2.2. User will stay on the map view with the same position as it was before action.

2.1.2.4 Browsing the list of pharmacies

Use case number	4
Priority	1
Use case name	Browsing the list of pharmacies
Description	User can browse the list of pharmacies in table view
Actor	User
Pre-conditions	Application is running and user is in “Pharmacies” tab.
Trigger	User taps on the “List”-button on the “Pharmacies” tab.
Result expected	List of pharmacies is shown grouped by regions.
Main scenario	1. User taps on “List”-button 2. List view is activated 3. List of pharmacies is shown to user as a table 4. User can scroll up and down in the table
Alternative scenario	3.1. If there is no Internet connection, no pharmacies are shown in the table and user is notified about the missing Internet connection 3.2. User will be on the empty Pharmacies list-view

2.1.2.5 Viewing detailed information of the pharmacy

Use case number	5
Priority	1
Use case name	Viewing detailed information of the pharmacy
Description	User can see detailed information about the chosen pharmacy, fields like open hours, address, phone and amount of medicines in the price list together with last refresh-date of the database are shown
Actor	User
Pre-conditions	User is in pharmacy list view or map view
Trigger	User taps on any of the pharmacy names in list view or disclosure icon on annotation.
Result expected	User will be redirected to detail information view of the chosen pharmacy with following fields displayed: pharmacy chain logo, pharmacy name, open hours, address, phone, last refresh date of the price list and small map with pharmacy location shown as a pin.
Main scenario	1. User taps on the pharmacy name in pharmacy list view or disclosure button on map annotation 2. User is redirected to new view with pharmacy detail info 3. Following fields are shown to user: chain logo, pharmacy name, address, open hours, phone, last refresh date of the price list, small map image with pharmacy location shown as a pin.
Alternative scenario	1.1. If there is no Internet connection, user will be notified about the situation and no detail-view of pharmacy is shown 1.2. User will stay in pharmacy list view

2.1.2.6 Searching for the medicine by name

Use case number	6
Priority	1
Use case name	Searching for the medicine by name
Description	User can search for items in pharmacies price lists by the name
Actor	User
Pre-conditions	User is in “Items” view
Trigger	User taps on the “Items” tab on tab bar and the items search view is activated with search box
Result expected	User can enter letters from the virtual keyboard and item names matching the search criteria are shown as a list in table to user.
Main scenario	1. User taps on the “Items” tab on tab-bar 2. Item search view is activated with search-field shown 3. User can activate the search box by tapping on it, virtual keyboard slides out 4. User enters letters using virtual keyboard to search item he is interested in 5. List of item names matching the search criteria are retrieved from the server 6. List of items retrieved from the server are shown to user as table
Alternative scenario	4.1. If there is no Internet connection, user will be notified about the situation 4.2. No items are retrieved from server and table of items will be empty
Alternative scenario	5.1. No items are matching the criteria entered by user thus the answer from the server is empty list 5.2. No items will be shown to user in table

2.1.2.7 Viewing the list of pharmacies selling the chosen medicine

Use case number	7
Priority	1
Use case name	Viewing the list of pharmacies selling the chosen medicine
Description	User can see and browse the list of pharmacies that are currently selling the item that was chosen by user from the item list view
Actor	User
Pre-conditions	User has searched the items by name from the item search view
Trigger	User taps on any of the names in items list table that were retrieved based on the search criteria entered by the user
Result expected	List of pharmacies which are selling currently the chosen item are shown in table
Main scenario	1. User taps on the item name in item search result table 2. New view is shown, with pharmacy list that are currently selling the chosen item 3. User can scroll up and down on the list
Alternative scenario	1.1. If there is no Internet connection, user will get notification about the situation 1.2. User will stay on the item search result list view

2.1.2.8 Viewing the detailed information for the chosen medicine

Use case number	8
Priority	1
Use case name	Viewing the detailed information for the chosen medicine
Description	User can see detailed information view of the chosen item
Actor	User
Pre-conditions	User is in the list view of pharmacies selling the chosen medicine
Trigger	User taps on the pharmacy name
Result expected	User will be directed to item detail view where following fields are shown: item name, pharmacy name, price, limit price, discount price (50%), discount price (75%), discount price (90%), discount price (100%) and three buttons: “Info sheet”, “Image” and “Diagnose”
Main scenario	1. User taps on the pharmacy name in pharmacy item list view 2. User will be redirected to item detail view 3. Item name, pharmacy name and price fields of item are shown to user together with three buttons.
Alternative scenario	2.1. If there is no Internet connection, user will be notified about the situation 2.2. User will stay on the pharmacy list view

2.1.2.9 Viewing the package information sheet for the chosen medicine

Use case number	9
Priority	2
Use case name	Viewing the package information sheet for the chosen medicine
Description	User can see the detailed information about the chosen item as PDF-file. This information is taken from the government database and is the same as the information pamphlet that comes with medicine package.
Actor	User
Pre-conditions	User is in item detail view
Trigger	User taps on the “Info sheet”-button in item detail-view
Result expected	User will be directed into separate view where the downloaded information sheet is shown in the form of PDF-file
Main scenario	1. User taps on the Info-button in item detail-view 2. Info sheet is downloaded from the server 3. Info-pamphlet is shown to user as PDF-file in separate view
Alternative scenario	2.1. If there is no Internet connection, user will get notified about the situation 2.2. No data is downloaded form the server 2.3. User will stay in item detail-view

2.1.2.10 Viewing the item image

Use case number	10
Priority	2
Use case name	Viewing the item image
Description	User can see the item image.
Actor	User
Pre-conditions	User is in item detail view
Trigger	User taps on the “Image”-button in item detail-view
Result expected	User will be directed into separate view where the image is shown.
Main scenario	1. User taps on the Image-button in item detail-view 2. Image is downloaded from the server 3. Image is shown to user in separate view
Alternative scenario	2.1. If there is no Internet connection, user will get notified about the situation 2.2. No data is downloaded form the server 2.3. User will stay in item detail-view

2.1.2.11 Viewing the diagnosis-related information for the chosen medicine

Use case number	11
Priority	2
Use case name	Viewing the diagnosis-related information for the chosen medicine
Description	User can see detailed information about the prescription-medicine
Actor	User
Pre-conditions	User is in item detail view
Trigger	User taps on the Diagnoses-button in item detail-view
Result expected	User will be redirected into special view, where detailed information about the diagnoses are shown for the chosen item
Main scenario	1. User taps on the Diagnoses-button in item detail-view 2. User will be directed into item diagnoses-view 3. Following fields are shown to user: item name, producer name, ATC, Active substance, separate fields for Diagnoses (50%, 75%, 90%, 100%), information fields for diagnoses (75%, 90%, 100%), legend-field with explanations about the abbreviations used in diagnoses info-fields
Alternative scenario	2.1. If Internet connection is missing, user will get notification about the situation 2.2. User will stay in item detail view

2.1.2.12 Browsing the ATC-list (Anatomical Therapeutic Chemical)

Use case number	12
Priority	1
Use case name	Browsing the ATC-list (Anatomical Therapeutic Chemical)
Description	User can browse the hierarchical ATC-list
Actor	User
Pre-conditions	Application is running
Trigger	User taps on the “ATC”-tab on tab-bar
Result expected	ATC-view is activated and user will see a table filled with ATC-tree items
Main scenario	1. User taps on the “ATC”-tab on tab-bar 2. ATC-view is activated and data downloaded from server 3. Table view is shown to user with ATC-codes and group names shown in it
Alternative scenario	2.1. If Internet connection is missing, user will get notification about the situation 2.2. User will be in empty ATC-view

2.2 Non-functional requirements

Non-functional requirements are divided into following sub-categories: usability, reliability, performance and supportability. More detailed overview about every category is given below.

2.2.1 Usability

- Application works on an iPhone, which is a smart phone with relatively small screen compared to computer monitors. Screen resolution is 480x320 or 960x640 pixels depending on the version of the iPhone. The larger resolution (960x640) will in our case only mean better dpi (dots per inch), the textual information fitted on the screen is still the same as on the smaller (480x320) resolution screen. Both displays are only 3.5 inch (89 millimeters) in diagonal. Considering the size of the screen, only minimal amount of user interface elements need to be shown on the screen to allow user to accomplish needed functionality.

- User interface follows the Human Interface Guidelines[1] provided by Apple. This will help user to be able to start using the application as quickly as possible, avoiding the long learning process that would be the case if completely new user interface would be used.

- All the views of the application are designed following the previously mentioned HIG, only using standard UI components provided by the Apple development tools.

- Installation of the application is done using the App Store of Apple, that can be accessed using the iOS device (e.g. iPhone) or iTunes software on a computer.

- Internet connection is needed for the application to work. All the information that is shown to user will be downloaded from the central database using HTTP protocol.

2.2.2 Reliability

- The application should handle all the errors, without quitting unexpectedly, that could arise while interacting with the central database over the Internet or displaying the information.

- The application may not contain any known critical functional errors on committing to Apples App Store. All known critical functional errors must be fixed before the commit. This does not apply to non-functional nor unknown functional errors.

2.2.3 Performance

Application is depending on the Internet to be able to work. This means that the application will not have its own database of medicines saved on the device. This also means that the speed of the application is dependent on the speed of the Internet connection user has. The data queried from the server is plain text. It should take no more than 5 seconds for any of the views to load its content from the central server. Only a portion of the data will be returned from the server based on the query sent

[1]

– http://developer.apple.com/library/ios/#documentation/UserExperience /Conceptual/MobileHIG/Introduction/Introduction.html

by the application. The next portion will be downloaded if needed (user has reached the end of the visible table). This way, the user interface will be very responsive and there is no need to download large amounts of data as a whole, but it can be divided into smaller parts and downloaded only if there is a need for the additional data.

2.2.4 Supportability

There will be no personal assistance available for the application users. There will be short manual available for the application that will describe the main functionality briefly. In case of problems there will be contact information shown on the web-page of raviminfo.ee.

2.3 Roundup

In this chapter I reviewed the functional and non-functional requirements for the mobile application. A detailed list of functional requirements in the form of use cases was also given.

Chapter 3

Mobile Application: Raviminfo

In this chapter I will give an overview of implementation of the mobile application. This chapter will be divided into three major sections: design, implementation and future work. In the end there will be a short roundup. Diagrams and code samples will be given to get better understanding of the implementation details.

3.1 Design

First I will give an overview of how the information flows starting from the pharmacy and ending with the mobile application. First the pharmacy information system will send the information about the items into data collection server. After the data is processed, it will be sent to raviminfo.ee server where it is saved into MySQL database. This server is responsible for serving the www.raviminfo.ee web-site and now also the API that is used by the mobile application. The API is written in PHP and uses simple GET-queries sent by the mobile application. After the query is processed and according data retrieved from the MySQL database, the result will be assembled in JSON-format and sent back to mobile application over HTTP.

As the mobile application is written in Objective-C and uses Cocoa Touch framework, it is obvious that the object oriented paradigm is used as well as MVC-design pattern. The MVC-pattern consists of three parts: model, view and controller. Models are responsible for keeping the application data. Views display the data to application user and get responses from the user. Controllers are mediating the logic between the models and views. This pattern is widely used and helps to reuse the existing components. By using the Cocoa Touch framework, it is natural to use MVC-pattern through out the whole application development. Every view that is created will contain both, a controller and a view and it is up to developer how the data is organized in models. In iOS development, there are several ways to store data, I am using dictionaries and arrays for this, as the application doesn't need to save data persistently and all the needed data is retrieved from the server in real-time.

The application is divided into three logical parts: pharmacies, items and ATC (Anatomical Therapeutic Chemicals). All these parts are controlled by the tab-bar controller. Inside the tab-bar controller, navigation controllers are used. Navigation controllers are useful for showing hierarchical data because the idea behind navigation controllers are simple stacks. Top level or root controllers can push new view controllers into the stack which in turn can push new view controllers into the navigation stack. If the view controller has done it's job, it will be popped out of the navigation stack.

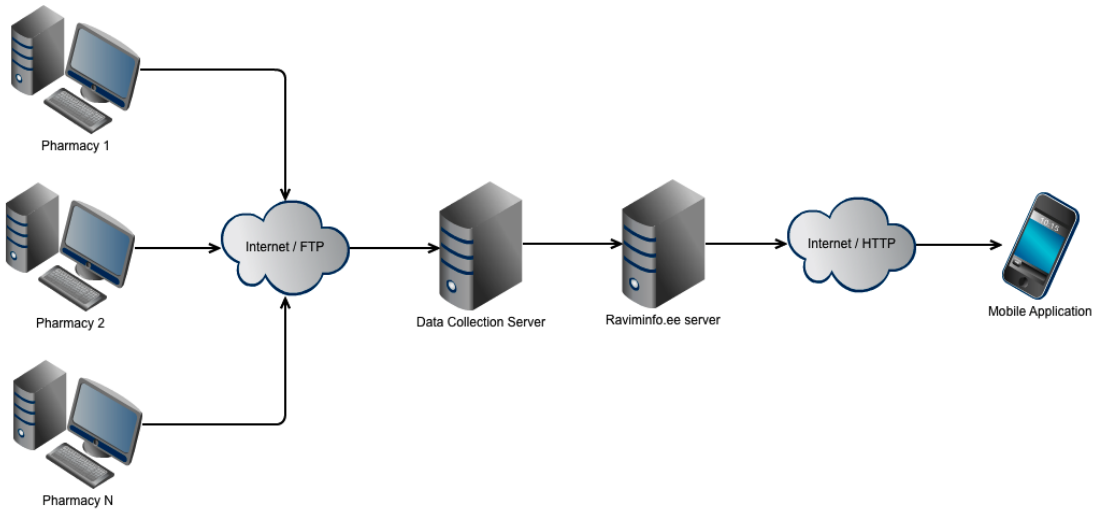

Figure 3.1: An overall diagram of how data flows from pharmacy to mobile application.

3.2 Implementation

3.2.1 Tools needed for development

For implementation I used the XCode IDE, which is freely available from the Apple website. The development tools are free, but if you are interested in developing using your iOS device, like iPhone, you have to buy a developer license from Apple. This license costs $99USD and is valid for one year. It is possible to develop the application using the iPhone simulator, but there are limitations present. For example you can not use GPS-coordinates with simulator, which was important in my case. Also, the simulator runs on a fast computer, which means that you are unable to see how your application really performs on a mobile device, so it is important to test your application on a real device. Without the developer license it is also not possible to send your application into Apple App Store, which is the only legal way to distribute your application to other iOS devices.

3.2.2 Knowledge needed for development

To be able to develop for iOS device with XCode, the developer needs to know the Objectice-C programming language. Objective-C is an object-oriented programming language that is built on top of the C-language. You can get a short overview of the language from this Wikipedia article. [1] I was studying this language from these two books [1] and [2]. I really liked the first one, as it was very well written and explained all the needed principles. The second book was also quite good, but was not as well written in my mind as the [1]. However each one of these books would be a nice start to get familiar with Objective-C language.

I would also recommend [8] as it has a great collection of quick tips and tricks for getting your answers for simple questions like how to convert strings into numbers with Objective-C. This is very useful if you just need some refresher on any of these issues.

When starting actually developing for the iOS, you need to get to know the Cocoa

[1] http://en.wikipedia.org/wiki/Objective-C

Touch and the principles and ideas behind the iOS application development. I started with the books [3] and [4] which is basically the same book as [3], with updates regarding to iOS 4. These books were also very well written and gave an overview of all the major ideas of the iOS development systematically. All the code examples in these books are also available online, but it is recommended to type the code in manually to learn more efficiently. I went through almost all of the code examples and can say that this was very useful. Using the XCode code-sense feature which really helps you to write code very quickly, as it provides you with the available method, variable and class names as you type, makes the coding a very pleasurable activity.

I was also reading a lot of other books about the iOS development, but not from cover-to-cover but just using some of the chapters that were currently needed to get a better understanding about some of the problems that occurred. All the books that were used are listed in bibliography part of this document. I will mention some of the books below next to some of the implementation parts, for example about how to create application icons, how to distribute your application to beta testers and how to distribute your application into App Store.

3.2.3 Interacting with the server

As was mentioned several times before, the application does not store any of the data locally. Application is retrieving everything that will be shown to the user in real time from the server. Data that is returned from the server is in JSON-format. I am using a third-party library for parsing the JSON-messages, because Apple does not provide a class for this. There are XML-parsers available from Apple, but I did not want to use XML, as it is more inconvenient to use the classes provided by Cocoa Touch. The XML-messages are also larger in size, which means that downloading the data from server will take longer and this makes the application slower for the end user.

To send a query to server, following url is used: http://api.raviminfo.ee/ There are currently five different queries that can be sent to server.

3.2.3.1 Pharmacy list

This query is used for retrieving the list of pharmacies from the server with all the according attributes. The main query string sent to server is "?q=p", however there can be three more parameters sent. These parameters are latitude, longitude and onlyNearest. Latitude and longitude are real numbers indicating the location coordinates of the mobile device which is sending the query and "onlyNear" parameter can be "0" or "1", indication wether user wanted to see all of the pharmacies or only the ones which are near the location where the user currently is. If these parameters are used, the query string will look like this: "?q=p&&latitude=59.433818&longitude=24.770077&onlyNearest=1" Fragment of the result message in JSON will look like this:

```
{"regions":[
    {
        "n":"TALLINN",
        "p":[
            {
                "i":"5060",
                "n":"AASA APTEEK",
                "d":"20.05.2011",
```

```
                "p":"+372-608-6565",
                "a":"Mustakivi tee 3A, 13912 Tallinn",
                "c":"20",
                "o":"E-R 10-21; L,P 12-20",
                "l":"59.4380086,24.8713330",
                "di":"163 km"
            },
            ....
        ]
    },
    ....
]}
```

The root node is "regions", which holds all the possible regions as an array. For every region there are two attributes, "n" for name of the region and "p" for the array of pharmacies in that region. In the pharmacy array there are the objects of the pharmacies with following attributes: "i" - id of the pharmacy, "n" - name of the pharmacy, "d" - last data refresh date, "p" - phone number, "a" - address, "c" - pharmacy chain id, "o" - open hours and days, "l" - location coordinates, "di" - distance from the mobile device which sent the query.

Distance is calculated on server side using the great circle distance algorithm. Listing of the php-code that calculates the distances for pharmacies is following:

```
class Location {
    const EARTH_RADIUS = 6371;

   /*
     * http://en.wikipedia.org/wiki/Great-circle_distance
     * */
    public static function getDistance($latitude1, $longitude1,
                $latitude2, $longitude2) {
        $dist = acos(
              cos(deg2rad($latitude1))
            * cos(deg2rad($latitude2))
            * cos(deg2rad($longitude1) - deg2rad($longitude2))
            + sin(deg2rad($latitude1))
            * sin(deg2rad($latitude2))
          );
          return round($dist * self::EARTH_RADIUS, 3);
    }
}
```

3.2.3.2 Item search

This query is used for retrieving the list of items matching the search string entered by the user. The main query will look like this: "?q=i&search=<search string>". The "search" parameter contains the actual string that user wanted to look up in database. Result of the query "?q=i&search=ibu" will be following JSON-message:

```
{"items":[
    {
        "a":"MM0034978",
```

```
        "n":"IBUMETIN TBL 400MG N10",
        "p":"NYCOMED"
    },
    {
        "a":"MM0034371",
        "n":"IBUMETIN TBL 400MG N30",
        "p":"NYCOMED"
    },
    ....
]}
```

The root node is “items”, which holds the possible items as an array. For every item there are following attributes available: “a” - aptcode, the unique ID for the item, “n” - item name, “p” - producer name.

3.2.3.3 Item in pharmacies

This query is used for retrieving the list of pharmacies that the selected item is available in. The main query will look like this: “?q=ip&aptcode=<item unique id>”. There can be additional parameters added like in pharmacy list query. This means that “latitude”, “longitude” and “onlyNearest” parameters can be used. Fragment of the result in JSON format for the query “?q=ip&aptcode=MM0007145 &latitude=59.433818 &longitude=24.770077 &onlyNearest=1” is following:

```
{"item_pharmacies":[
    {
        "n":"Nearest",
        "p":[
            {
                "i":"MM0007145",
                "pi":"31937",
                "n":"SELMA APTEEK",
                "p":"2.37€",
                "di":"368 m"
            },
            {
                "i":"MM0007145",
                "pi":"31461",
                "n":"ANNA APTEEK",
                "p":"2.37€",
                "di":"423 m"
            },
            ...
        ]
    }
]}
```

The root node is “item_pharmacies”, which holds the array of regions or just one array item called “Nearest”. Objects in the array have two attributes, “n” for name and “p” for pharmacies. For every pharmacy object these attributes are available: “i” - item unique ID, “pi” - pharmacy unique ID, “n” - pharmacy name, “p” - price with currency symbol, “di” - distance of the pharmacy from the location of the mobile device.

3.2.3.4 Item details

This query is used for retrieving the detailed information about a certain item. The query will look like this: "?q=id&aptcode=<item id>&client_id=<client id>". Returned message from the server in JSON-format for the query "?q=id &aptcode=MM0007145 &client_id=31937" is following:


```
{"item_details":[
    {
        "itemId":"MM0007145",
        "itemName":"DOXY-M-RATIOPHARM TBL 100MG N10",
        "producerName":"RATIOPHARM GmbH",
        "pharmacyName":"SELMA APTEEK",
        "price":"2.37€",
        "priceLimit":"1.89€",
        "priceDisc50":"2.37€",
        "priceDisc75":"",
        "priceDisc90":"",
        "priceDisc100":"1.75€",
        "diagnose50":"",
        "diagnose75":"",
        "diagnose100":"(A50-A53)*",
        "info50":"",
        "info75":"",
        "info100":"*rv dermatoveneroloogil",
        "atc":"J01AA02",
        "actSubst":"Doxycyclinum",
        "diagExpl":[
            {
                "name":"rv",
                "text":"ravimi väljakirjutamise õigus"
            }
        ],
        "infoUrl":"http:\/\/koodikeskus.ravimiamet.ee\/Data\/PIL
           \/PIL_1144976.pdf",
        "imageUrl":"http:\/\/www.magnum.ee\/pictfiles\/7145.jpg"
    }
]}
```


The root node is "item_details", which contains an array with just one element in it. That element contains following attributes: "itemId" - unique id of the item, "itemName" - item name, "producerNam" - item producer name, "pharmacyName" - name of the pharmacy where this item is sold, "price" - regular price of the item, "priceLimit" - limit price of the item, "priceDisc50" - price of the item with 50% discount, "priceDisc75" - price of the item with 75% discount, "priceDisc90" - price of the item with 90% discount, "priceDisc100" - price of the item with 100% discount, "diagnose50" - diagnoses for 50% discount, "diagnose75" - diagnoses for 75% and 90% discount, "diagnose100" - diagnoses for 100% discount, "info50" - information for 50% diagnoses, "info75" - information for 75% and 90% diagnoses, "info100" - information for 100% diagnoses, "atc" - ATC of the item, "actSubst" - active substance of the item, "diagExpl" - array of diagnose explanations, "infoUrl" - url where package info sheet can be downloaded, "imageUrl" - url where item image can be downloaded.

3.2.3.5 Atc list

This query is used for retrieving the list of ATC-codes and names by the ATC-code beginning. The query will look like this: "?q=a". Additional parameter can be used to specify which sub-tree of the ATC is wanted: "?q=a&atc=<ATC code>". Fragment of the returned message from the server in JSON-format for the query "?q=a&atc=A01" is following:

```
{"atc":[
    {
        "a":"A02A",
        "n":"ANTATSIIDID",
        "c":"31"
    },
    {
        "a":"A02B",
        "n":"HAAVANDTÕVE RAVIMD",
        "c":"26"
    },
    ...
]}
```

The root node is "atc", which contains an array of subitems of the given ATC-code. Following attributes are available for every array element: "a" - ATC-code, "n" - name of the ATC group, "c" - number of subitems for this given ATC-code.

3.2.4 Mobile application

In this subsection I will give an overview how the mobile application is organized into modules and what is

The main.m file of the application is very simple:

```
#import <UIKit/UIKit.h>
int main(int argc, char *argv[]) {
    NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
    int retVal = UIApplicationMain(argc, argv, nil, nil);
    [pool release];
    return retVal;
}
```

The most important part here is the UIApplicationMain function which creates the application object and application delegate and sets up the event cycle.

The application delegate itself will declare the UITabBarController instance variable which will be used as an IBOutlet in MainWindow.xib. Xib-files are XML-files that contain the interface objects and their relationships with the Objective-C code. Xib-files are created using special tool called "Interface Builder", which is integrated into XCode IDE since the version 4.0. In MainWindow.xib I created the tab-bar controller which is connected to the UITabBarController instance in application delegate object.

For every tab-bar item (pharmacies, items, ATC) there is a navigation controller associated with it. Navigation controllers are useful for creating view hierarchies. The idea behind navigation controller is a simple stack. New view controllers can be pushed into stack and popped out if these view controllers are no longer needed.

Next I will give a more detailed overview about these three different parts of the application.

3.2.4.1 Pharmacies

For pharmacies I am using two different ways to display information: map view and list view. For map view I created a PharmacyMapViewController which subclasses UIViewController and uses MapKit and CoreLocation API-s. The MapKit API is showing a Google Map based map, which can hold annotation pins on given coordinates. CoreLocation API is used for retrieving the GPS-coordinates of the mobile device.

For the list view of pharmacies, I created a PharmacyTableViewController which subclasses the UITableViewController. UITableViewController is one of the most used ways to show information to user. It has several built-in methods that developer can override to be able to specify how many sections your table has, how many rows are in every section, how the information in the cells are formatted and how the tap on the cell is handled.

Both the map view and table view of the pharmacies are using the PharmacyDetailViewController, which is a subclass of UIViewController and is a simple class for showing detailed information for the given pharmacy.

3.2.4.2 Items

For item search there is a class ItemSearchTableViewController, which is a UITableViewController subclass. ItemSearchTableViewController contains a UISearchBar instance variable which is the IBOutlet for according UI element. If user enters letters into search box, this triggers a searchBar: textDidChange method call and new query string is assembled and passed to an instance of UrlDataDownloader class. All the interaction with the server is done asynchronously in separate thread, which means that the UI will stay responsive while the data is retrieved from the server. After the response is retrieved from the server, the contents of the table is refreshed and user will see a new information.

If user taps on any of the items in table, this will fire a tableView: didSelectRowAtIndexPath method which initiates a ItemPharmacyTableViewController instance and pushes it into navigation controller stack. ItemPharmacyTableViewController is a subclass of UITableViewController and it will retrieve the information about the pharmacies that the selected item is available in from the server. This class is again using UrlDataDownloader to retrieve the data from server. Table is filled with information asynchronously after the data has been downloaded.

If user taps on any of the pharmacy names, it will result in call of tableView: didSelectRowAtIndexPath method in ItemPharmacyTableViewController instance. The ItemPharmacyDetailViewController instance is created and pushed into navigation controller stack. ItemPharmacyDetailViewController is a subclass of UIViewController and is responsible for showing the price information of the selected item in selected pharmacy. If there are diagnoses, image or package info sheet information available in retrieved JSON-message, then according buttons are enabled for the user.

If user taps on the package info sheet button, it will fire the showInfoSheet method in detail view controller and a new WebViewController instance is created and shown as a modal view. WebViewController uses WebKit API which is basically the Safari browser and loads the given url into it.

If user taps on the image button, it will fire the showImage method which creates a WebViewController instance and displayed as a modal view just like it was done with package info sheet.

If user taps on the diagnoses button, the showDiagnoses method is used to create a ItemDiagnoseViewController instance and created instance is pushed into navigation controller stack.

ItemDiagnoseViewController is a subclass of UITableViewController and it uses grouped table view to display the information retrieved from the server using the instance of the UrlDataDownloader class.

3.2.4.3 ATC

For ATC-list there is a class AtcTableViewController which is a subclass of UITableViewController. It uses the instance of the UrlDataDownloader class download the JSON-messages from the server. If the result from the server shows that there are subitems available for the selected ATC-code, then in AtcTableViewController method tableView: didSelectRowAtIndexPath an instance of the AtcTableViewController is created and pushed into navigation controller stack. If there are no more sub-levels of ATC left, then an instance of ItemSearchTableViewController is created instead and pushed into navigation controller stack resulting in item search by the ATC-code.

3.2.5 Artwork restrictions

Because of different screen resolutions available on iOS devices, there are some important notes about the images used in iOS applications. There are currently three different screen resolutions available:

- 320x480 px, used in iPhones and iPod Touches prior to version 4
- 640x960 px, used in iPhones and iPod touches from version 4
- 768x1024 px, used in iPad and iPad2

As the physical screen size of the iPhones and iPod Touches are the same, but resolutions can be 320x480 or 640x960, it means that there is a need for images for both resolutions. If using the same images that were designed for lower resolution iOS devices on high resolution iOS device, these images will look pixelated and ugly. Apple makes it quite easy for developers to be able to overcome this problem. There is the naming convention available, where you can name you original image as “image.png” and high resolution version as “image@2x.png”. Now in your application you are just going to use the original image name and iOS itself chooses the correct image for you on the high resolution devices.

For the application icon it is important to create the 512x512 px version first, as it will be the size that is needed for App Store. From this large version you can create all the smaller versions for your application:

- 57x57 px - regular iPhone icon (named as Icon.png)
- 114x114 px - icon for the high-resolution iPhone (named as Icon@2x.png)
- 72x72 px - icon used on iPad (named as Icon-72.png)
- 29x29 px - small icon used on search screen of the iPhone (named as Icon-Small.png)

- 58x58 px - small icon used on search screen of the iPhone with high-resolution display (named as Icon-Small@2x.png)
- 50x50 px - small icon used on search screen of the iPad (named as Icon-Small-50.png)

3.2.6 Submitting to the App Store

As the App Store review process is quite strict, it means that all of the applications are tested by the Apple employees. If any of the issues are discovered, the application will be rejected and you will get just a very short note about what caused the rejection. However, that only means that you get noted about first problem that the review team discovered and if you have more issues, you will only get info about them after you have fixed the first problem and resubmitted your application. To be able to submit you application into App Store, there are several important things that needs to be done first. Most important part of course is to make sure the application does not crash. Other issues that can result in rejection of the application [11]:

- Incorrect version number
- Incorrect icon image
- Using Apple artwork in your application
- Using copyrighted material in your application
- Violating the HIG guidelines
- You application is copying the functionality of some other existing application
- Not giving correct notifications in case of missing network connection
- Using a large amount of network bandwidth
- Using incorrect keyboard type
- Incorrect minimal OS version is specified in compiler settings
- Description of the application is missing
- Functionality of the application is too minimal
- Application does not work as advertised

I completed the application and submitted it into App Store. The process was not trivial. First you have to connect Itunes Connect website at https://itunesconnect.apple.com/. There you can manage your applications. After you add new app, you need to fill out a form with all the important details about you application like its name, description, logo image, keywords and screenshots of your application. Next you can upload your application binary with a special application called Application Loader. After uploading your application binary into iTunes Connect server, it will go into the review queue and will be reviewed by Apple employees. In about a week you will get an e-mail which says if your application was approved or rejected. If your app got rejected, you need to fix the issues pointed out by the review team and resubmit for the review again. If your application got accepted, then your application will be available in the App Store for download.

3.3 Future work

The mobile application was developed with minimal needed set of functionality. During the research and development some interesting ideas came up that could be potential candidates for future developments.

There is a sister site of raviminfo.ee in Latvia called zales24.lv, it should be quite interesting to include the data from that site into the mobile application as well.

Right now the application was created in only one language, it is important to add more translations, so English, Estonian, Finnish, Swedish, Latvian and Russian would also be available.

It would be great to add support for mobile-ID so that users could check out the digital prescriptions written out for them by their doctors.

As the mobile phone has a camera, it would be interesting to add bar code scanning functionality to make finding items by scan code very simple and convenient.

There could be possibility for users to set up reminder events like time when medicine needs to be taken using the application.

There is a database for medicine interactions available that could be used by the application so users could find out if two medicines can be used together or not.

I am also considering other mobile platforms, like Android, Windows Phone OS 7 and Symbian. It would be interesting to develop similar application for other mobile platforms in the future.

3.4 Roundup

In this chapter I described the design and implementation of the mobile application. The application is available for everybody in Apple App Store for free. A list of ideas for future development was also given.

Chapter 4

User manual

In this chapter I will give an overview on how end users can use the mobile application.

4.1 Installing the application

Installing the application on iPhone will be very simple, user needs to launch the App Store application and search for "Raviminfo". After the application is found, the user just needs to tap on the Install-button and enter his iTunes password. After the installation process is finished, user will see new application called "Raviminfo" on his iPhone screen (figure 4.1).

4.2 Launching the application

Launching the application is also very simple, user has to find the previously installed application on his iPhone desktop and tap on the icon of the application. Application will start in pharmacy map view mode, where user can see the pharmacy icons near his location (figure 4.2).

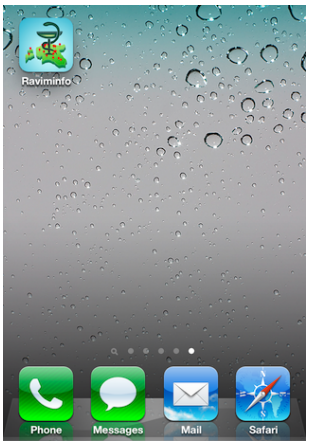

Figure 4.1: Launching the application.

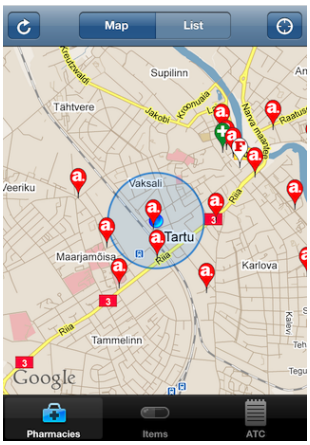

Figure 4.2: Pharmacies on the map.

Figure 4.3: Showing the annotation of a pharmacy.

Figure 4.4: Listview with the nearest pharmacies.

Figure 4.5: Pharmacy detail view.

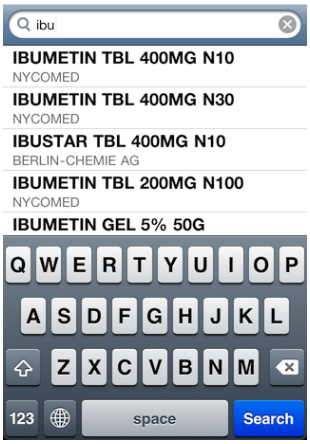

Figure 4.6: Item search view.

Figure 4.7: Nearest pharmacies where given item is currently available.

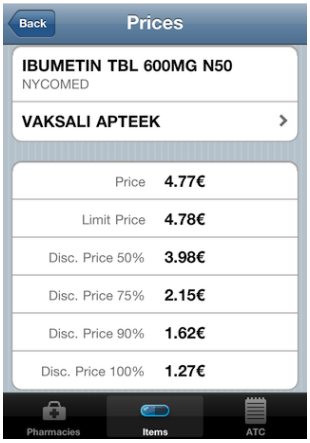

Figure 4.8: Item price view.

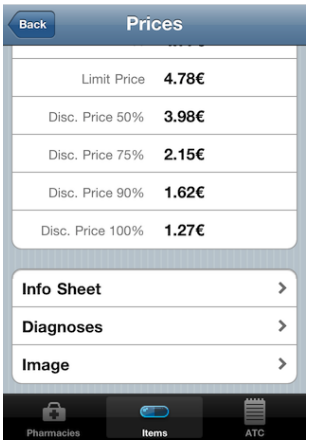

Figure 4.9: Buttons to info sheet, diagnoses and image views.

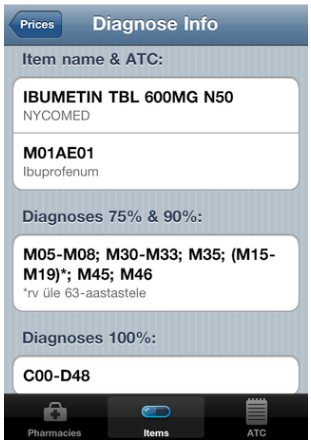

Figure 4.10: Diagnose info view.

Figure 4.11: ATC list view.

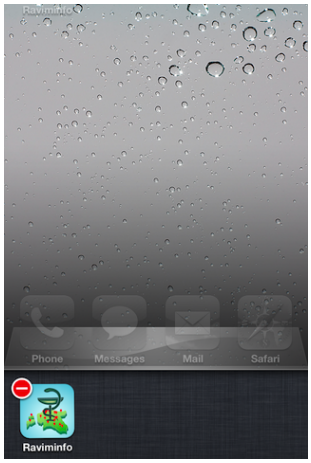

Figure 4.12: Closing the application.

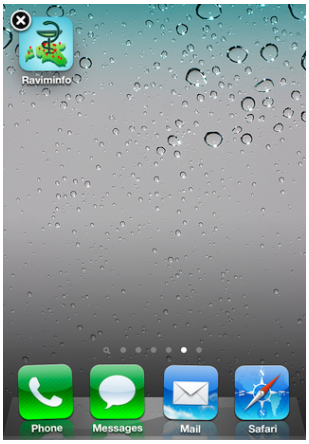

Figure 4.13: Uninstalling the application.

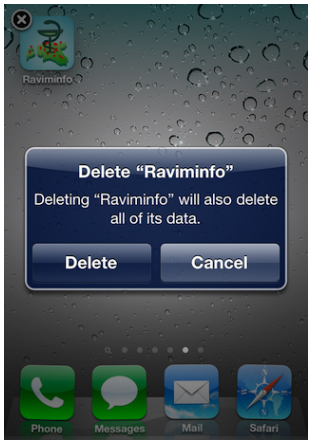

Figure 4.14: Uninstall confirmation alert view.

4.3 Browsing the pharmacies

There are two ways to get information about pharmacies, first there is a map view of the pharmacies and then there is list view of the pharmacies. User can open pharmacy section by tapping on the "Pharmacies" tab on tab-bar. In pharmacies view there are two buttons in the upper area of the screen: "Map" and "List". User can see pharmacies on the map if tapping on the "Map" button and see pharmacies as a list by tapping the "List" button. In the map-view, user can see pharmacies on the map of Estonia and tap on the pharmacy icons. After tapping on the icon, small annotation will be displayed with possibility to see detailed information about the pharmacy (figure 4.3). In the list view, user can see pharmacies as a grouped list (figure 4.4), with sections named as regions and pharmacies sorted alphabetically in these sections. User can tap on any of the pharmacy cell and will be redirected to detailed view of the pharmacy.

In the detail view of the pharmacy (figure 4.5), user can see pharmacy name, address, phone number, last data refresh date and small map with pharmacy location shown as a pin. To return from the detail view, user hast tap on the "Back" button in navigation bar.

4.4 Searching for items

User has to tap on the "Items" tab in tab-bar to open the item search view. In search view, there is a search box where user can tap on. After tapping on the search box, a virtual keyboard will slide open and user can start entering item name. After every letter-entry, real-time search is performed and result retrieved from server. Results of the search are automatically shown in table below the search bar (figure 4.6). After user taps on any of the search results he will be redirected to a view where all the pharmacies are shown where this item is currently available (figure 4.7).

4.5 Browsing the pharmacies where selected item is available

Default view of the item pharmacies is maximum 20 nearest pharmacies. User can also tap on the navigation bar button called "All" to see all the pharmacies the selected item is available at the moment. In the list view user can see the pharmacy name, the price of the item in that pharmacy and distance between user location and the pharmacy location. User can return to search view by tapping on the "Back" button on navigation bar. If user wants to see more detailed information about the item in any of the pharmacy listed, he can tap on the table row.

4.6 Detailed item info in selected pharmacy

In detailed information view of the item, user can see the full name of the item and pharmacy. All the possible prices are shown below the item and pharmacy names. There can be six different prices available: regular price, limit price and prices for 50%, 75%, 90% and 100% discounts (figure 4.8). In the bottom of the screen there are three buttons: "Info sheet", "Image" and "Diagnose" (figure 4.9). "Info sheet" button

will open a separate view, where the package info sheet pdf is shown. “Image” button will open a separate view where item image is shown. “Diagnose” button will open a separate view where all the diagnose codes are shown for the discount prices (figure 4.10).

4.7 Browsing the ATC table

User has to tap on “ATC” tab on tab-bar to enter the ATC-view. In ATC-view user can see a table with ATC codes and according ATC group names. User can tap on any of the blue disclosure icons to see sub-level of any given ATC code (figure 4.11). If user is not on the top level of the ATC hierarchy tapping the cell will redirect to item search view with given ATC code entered in search box.

4.8 Closing the application

To close the application, user has to press the “Home”-button of his iPhone. This will put the application in the background. If user wants to completely close the application he has to enter the task-view of the iPhone by double-tapping on the “Home”-button. In the task-view, user has to touch any of the icons listed for couple of seconds until the red icons appear on the upper left corner of the icons (figure 4.12). After entering the application closing mode, user can tap on the red icon to close any of the applications currently running. To close “Raviminfo” application, user has to find the “Raviminfo” icon from the list and tap on the red icon.

4.9 Uninstalling the application

User can uninstall the “Raviminfo” application by locating the application icon on his iPhone desktop and tap and hold on that icon for couple of seconds until the icon starts to shake and black icon appears on the upper left corner of that icon (figure 4.13). After tapping on that black icon with cross on it, user will get a confirmation dialog that asks if user is sure about uninstalling the application (figure 4.14). After agreeing on removing the application from the device, the application will get uninstalled.

Conclusion

As the main outcome of my Bachelor thesis, a mobile application called Raviminfo was developed. It consists of three main functionalities: possibility to get information about pharmacies in Estonia, the possibility to search for items available in these pharmacies and the possibility to browse the ATC table and search items by the ATC-code. It is possible to see locations of the pharmacies on the geographical map and it is possible to find the distances of the pharmacies from the application user's physical location. It is possible to see detailed information about the items sold in pharmacies, for example different prices, package info sheet, image and diagnoses are available.

The application was developed using the Objective-C language and the Cocoa Touch framework provided by Apple. Interaction with the server is done through a special API. The API is using the HTTP-protocol for data transfer, GET-requests are used for sending queries to server and data is returned from the server in the JSON-format.

The application is complete, working as expected and submitted into Apple App Store.

There are a lot of new ideas about what new functionalities could be implemented in this application. One of the most interesting and important ideas is to add information about the Latvian pharmacies that are currently available in raviminfo.ee sister site called zales24.lv. Also it would be interesting to add possibility for users to find out about the interactions between different medicines. This can help reducing the threat that two or more medicines are consumed together, that could result in fatalities.

There are also other mobile platforms available, that are interesting and there are ideas to develop similar application for the Android and Windows Phone platforms.

As the raviminfo.ee is already a very well established web-site in its niche, it will be interesting to see how popular the mobile application will be among its users.

Mobiilse farmatseutilise infokeskkonna arendus

Martin Vels

Bakalaureusetöö (6 EAP)

Sisukokkuvõte

Juba ligikaudu kümme aastat on aadressil www.raviminfo.ee olnud kõigile kättesaadav apteekide ja ravimite andmebaas ning otsingumootor. Mainitud veebilehe populaarsus on aastatega pidevalt kasvanud ning üha enam inimesi kasutavad seda veebilehte kui ainsat võimalust, et saada infot apteekides müüdavate ravimite ja teiste apteegikaupade kohta.

Kuna peaaegu iga inimene kasutab tänapäeval mobiiltelefoni ja üha suurem hulk inimesi kasutab nutitelefoni, siis oli raviminfo.ee arengu järgmine loogiline etapp luua spetsiaalne tarkvararakendus nutitelefonile. Pilootprojektina valisin arendusplatvormiks Apple'i poolt loodud operatsioonisüsteem iOS, mis on mõeldud erinevatele mobiilsetel seadmetele nagu iPhone, iPad ja iPod Touch. Bakalaureusetöö tulemusena valmiski mobiilirakendus, mille abil on võimalik saada infot Eesti apteekide asukoha, lahtioleku aegade ja kontaktandmete kohta.

Mobiilirakenduse arendusel kasutati Apple'i arenduskeskkonda XCode, programmeerimiskeeleks Objective-C ning raamistikuks Cocoa Touch. Kogu arendusprotsess on dokumenteeritud, kasutades levinud FURPS-mudelit. Kõik kasutuslood on samuti detailselt esitatud.

Mobiilirakendus koosneb kolmest loogilisest osast, apteekide vaatest, nimetuste otsinguvaatest ning ATC e. hierarhilise toimeainete andmebaasi vaatest. Rakendus võimaldab kasutajal näha apteekide asukohta geograafilisel maakaardil, samuti on võimalik apteekide nimekirja vaadata grupeerituna regioonideks. Täiendav võimalus on näha kasutajale parajasti lähimate apteekide nimekirja. Nimetust otsing võimaldab leida kasutajat huvitava nimetuse hetkel apteekides müügiolevate nimetuste hulgast. Nimetuste kohta on võimalik näha detailset infot nagu erinevad hinnad, tootja ja toimeained, pakendi infoleht, diagnoosid, millega vastavat ravimit toihib välja kirjutada ja pakendi pilt. ATC-tabel võimaldab otsida hierarhilisest toimeainete andmebaasist ning leida vastava ATC-koodi alusel parajasti apteekides müüdavaid nimetusi.

Tulevikus on kavas rakendust edasi arendada, lisada raviminfo.ee Läti sõsarlehe zales24.lv andmed ning lisada võimalus ravimite koostoimete leidmiseks spetsiaalsest andmebaasist. Samuti on idee laiendada platvormide hulka, millel loodud mobiilirakendus töötaks, plaanis on Andoroidi ja Symbiani või Windows Phone 7 platvormile rakenduse loomine.

Kokkuvõttes saab öelda, et töö oli edukas, sest valmis Apple'i rakendustepoest kõigile vabalt kättesaadav ning kasulik mobiilirakendus, mis võimaldab leida kiiresti ja mugavalt infot lähedalasuvate apteekide kohta ning otsida ravimite kohta käivat infot.

Bibliography

[1] Mark Dalrymple, Scott Knaster. Learn Objective-C on the Mac. Apress, 2009.

[2] Stephen G. Kochan. Programming in Objective-C 2.0. Addison-Wesley, 2009.

[3] Dave Mark, Jeff LaMarche. Beginning iPhone 3 Development. Exploring the iPhone SDK. Apress, 2009.

[4] Dave Mark, Jack Nutting, Jeff LaMarche. Beginning iPhone 4 Development. Exploring the iOS SDK. Apress, 2011.

[5] Joachim Bondo, Dylan Bruzenak, Steve Finkelstein, Owen Goss, Tom Harrington, Peter Honeder, Ray Kiddy, Noel Llopis, Joe Pezzillo, Florian Pflug, Jonathan Saggau, Ben Britten Smith. iPhone Advanced Projects. Apress, 2009.

[6] David Barnard, Joachim Bondo, Dan Burcaw, David Kaneda Craig Kemper, Tim Novikoff, Chris Parrish and Brad Ellis Keith Peters, Jürgen Siebert, Eddie Wilson. iPhone User Interface Design Projects. Apress, 2009.

[7] Dave Mark, Jeff LaMarche. More iPhone 3 Development: Tackling iPhone SDK 3. Apress, 2009.

[8] David Chisnall. Objective-C Phrasebook. Essential Code and Commands. Addison Wesley, 2011.

[9] James A. Brannan, Blake Ward. iOS SDK Programming: A Beginner's Guide. McGraw Hill, 2011.

[10] Vandad Nahavandipoor. iOS 4 Programming Cookbook. O'Reilly, 2011.

[11] Alasdair Allan. Learning iPhone Programming. O'Reilly, 2010.

[12] Maher Ali. Advanced iOS 4 Programming: Developing Mobile Applications for Apple iPhone, iPad and iPod touch. Wiley, 2010.

Appendix A

Resources

The source code of the mobile application is available on the CD attached to this thesis.