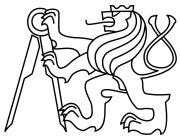
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY



ASSIGNMENT OF BACHELOR'S THESIS

Title:	ElateMe - iOS client II
Student:	Gleb Arkhipov
Supervisor:	Ing. Jiří Chludil
Study Programme:	Informatics
Study Branch:	Software Engineering
Department:	Department of Software Engineering
Validity:	Until the end of summer semester 2017/18

Instructions

The ElateMe system is a web application with mobile clients providing functionality of crowdfunding and wishlist satisfaction of users and their friends (e.g., for Christmas, birthday, etc.). The ElateMe is a team project. The aim is to analyse and implement an iOS client for ElateMe.

1. Analyse/create:
   - entities: user, wish, recommendation, advertisement,
   - domain diagram,
   - platform non-specific class diagrams,
   - payments via the FIO bank.
2. Design:
   - a platform specific model and class diagram,
   - a server part for FIO bank payments.
3. Implement:
   - news feed functionality and its management,
   - wish management including wish product suggestion,
   - crowdfunding via the FIO bank,
   - friendlist management,
   - common use cases (login, registration, logout, setting, notifications, wish sharing),
   - GUI (based on a 3rd party design).
4. Perform usability tests in the usability lab.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague February 14, 2017

Czech Technical University in Prague
Faculty of Information Technology
Department of Software Engineering

Bachelor's thesis

ElateMe - iOS client II

Gleb Arkhipov

Supervisor: Ing. Jiří Chludil

15th May 2017

Acknowledgements

I would like to express my sincere gratitude to Ing. Jiří Chludil and Bc. Michal Maněna for their guidance. I would like to thank Yegor Terokhin, Georgii Solovev, Yevhen Kuzmovych, Maksym Balatsko, and Boris Laskov for making ElateMe possible. I also thank our designer Ing. Jan Hoffman for fresh modern graphics. Finally, I thank every tester, who sacrificed his time for testing ElateMe application.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on 15th May 2017 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Arkhipov, Gleb. *ElateMe - iOS client II*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2017.

Abstrakt

Hlavním cílem této práce je vývoj mobilní aplikace, která obsahuje prvky sociální sítě a hromadného financování. Aplikace poskytne příležitost pro její uživatele, aby si splnily své sny. Jejich přátelé v aplikaci budou moci financovat uživatelova přání, což doufám změní svět k lepšímu. Na tomto týmovém projektu, pracuje pět vývojářů a běží na třech různých platformách: iOS, Android a backend.

Na začátku byla udělaná analýza, jelikož bylo nutné pochopit doménový model a strukturovat existující požadavky. V této fázi byl vytvořen platformě nezávislý model. Pak ten model přešel do platformě závislého, aby vyhovoval omezením a limitacím. Následně provedená implementace byla detailně popsána tak, aby bylo vidět řešení případných problémů. V závěru bude navrženo uživatelské testování, jež bylo provedeno v kontrolovaném prostředí laboratoře uživatelského testování. Na zmíněného testování navazuje shrnutí prezentující nalezené nedostatky.

Klíčová slova ElateMe, iOS, iPhone, hromadné financování, sběr peněz, sociální síť, uživatelské testování

Abstract

The major objective of this study is to develop a mobile application that involves elements of social networking and crowdfunding. It will provide opportunity for its users to make their dreams come true. Their friends inside the application will support users by funding wishes financially making the world a better place. This is a team project consisting of five developers and three different platforms: iOS, Android, and backend.

Firstly, analysis will be done to understand domain objects and structurize the given requirements. In this phase, a platform-independent model will be created. Secondly, the platform-independent model will become a platform-specific model to satisfy platform constraints and limitations. After that, the implementation will be described in a greater detail including solutions to the emerged problems. Then usability testing will be performed. It will be conducted in a controlled environment recording participants from different perspectives. Finally, the aforementioned testing will be analyzed presenting its results.

Keywords ElateMe, iOS, iPhone, crowdfunding, fundraising, social network, usability testing

Contents

List of Figures

Chapter 1

Introduction

Crowdfunding services are slowly making their way into our lives. "*Crowdfunding is a financing method that involves funding a project with relatively modest contributions from a large group of individuals, rather than seeking substantial sums from a small number of investors.*"[3] Just a few years ago no one could have imagined that he would send money to support some product which is not even in production yet. Now it happens every day. People gather money for technical gadgets, films, games, and just for fun.

The first company that is associated with crowdfunding is Kickstarter. It was founded in 2009, and since then it helped to raise almost 3 billion dollars for more than 300 000 projects. Kickstarter is used to fund larger projects, but what if it was possible to invest into something smaller like a gift to a friend. And here ElateMe could come into play. ElateMe is the name for a mobile application that gives its users an opportunity to fulfill their wishes. For the sake of simplicity, the user is referred in a masculine form (he, him) throughout this bachelor's thesis. The user will create his wish describing why his friends should support it. Then his friends will donate him to make this wish real. After a required amount make, the user will receive money to buy what he wants.

Development of ElateMe application has begun during the fourth semester of studying as a team software project. It was unclear at that time whether it will succeed, or even be finished at all. It was more like a proof of concept than anything else. The first working prototype was ready by the end of the fourth semester, but it had crude graphics and limited functionality. In the fifth semester, the decision was taken to discard old prototype and start completely from scratch. Fresh and revamped graphics were created by Ing. Jan Hoffman, who is an actual UI/UX designer in ElateMe team. New domain model was designed to meet new requirements. The new architecture was made to solve some problems of the old one. After that, the development of the current version of ElateMe application started. Creating such service is hardly possible without a proper team. That is why our team consists of

several people: project manager, UI/UX designer, one Android developer, two iOS developers, and two backend developers.

My personal motivation to participate in this team was to try collaborating with different people in a large software project. ElateMe became my first project which had its own scope, deadline, and requirements. I realized that it could provide invaluable experience and increase my programming skills. Besides that, I am inspired to create a service that will help people to become happier.

Chapter 2

Analysis

This chapter focuses on describing main domains that were used in ElateMe application. It also includes description of the platform-independent class diagram in **Figure 2.1**. Next section after PIM is about functional and non-functional requirements of ElateMe application. The last but not the least part of the analysis belongs to organizing payments via FIO bank gateway.

2.1 Domain model

Domain diagram **Figure 2.1** provides visual representation for all model entities used in ElateMe application. It defines several main domains which are essentially data structures that map real world requirements to the model. Domain model was designed and explained in a more detailed way in the bachelor's thesis by Maxim Balatsko [4]. It is important to notice that this model illustrates server entities and relations. Unlike the server model, mobile applications hold their entities with less dependency upon each other. That could be read about in **Section 2.3**.

2.1.1 User

User entity represents application user. It stores essential information about him. The user is responsible for creating new content in ElateMe application. This makes user capable of writing his wishes, receiving surprises, adding comments, or sending donations to other users. The user is in 1:N relation with other users because he has friends who are also users. Due to the fact that Graph API from Facebook does not allow mobile applications to ask for user's friends, only those who have installed ElateMe application could be shown. It is possible to overcome this limitation by creating our own registration, but it will complicate user experience with the application. This functionality may be added in the next release; however, it is still a thing of debate.

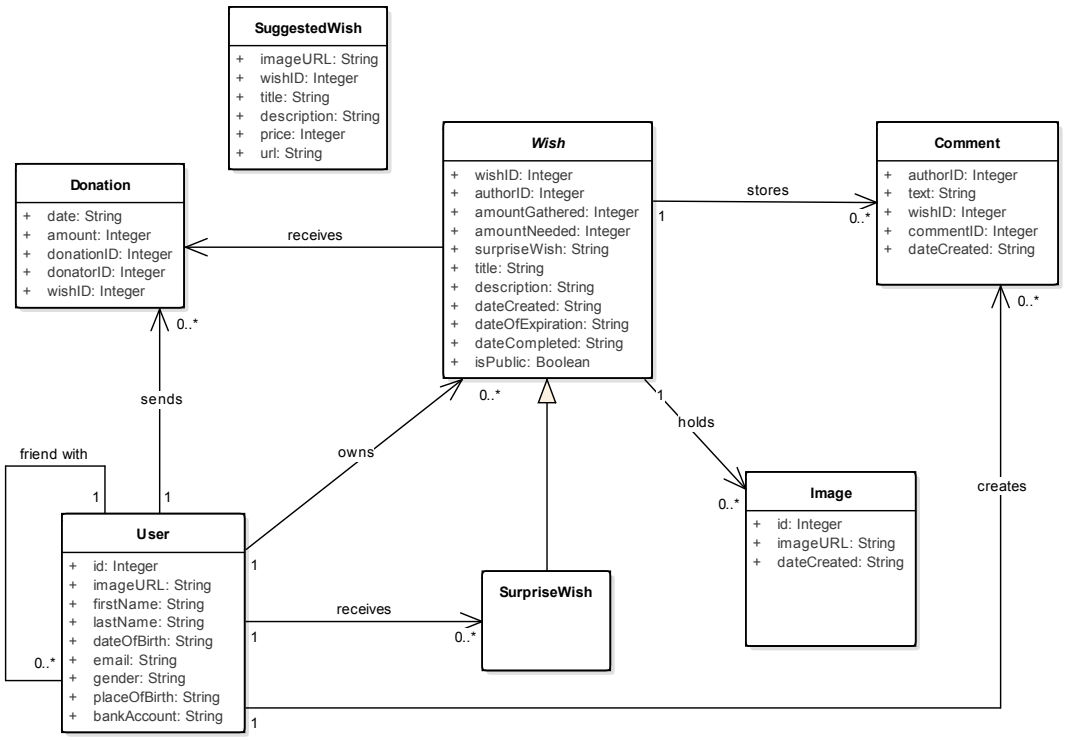


Figure 2.1: Domain model

2.1.2 Wish

Wish entity may be described as the main entity in ElateMe application. It should hold all data required to present user's dream or wish in such a way that other people would like to support his wish. That is why each wish has an owner. Wish could receive one or multiple donations from different users. All users who see a wish have an opportunity to write some comments to it, so it is necessary to store every comment in 1:N relation. Whole wish management will be explained in **Section 4.4**.

2.1.3 Surprise wish

Surprise wish is a special case of the regular wish. Several people plan to make a surprise for someone by buying a gift. Then they create a surprise wish for one of their friends. After that, they chip in to gather the required amount. When the wish is completely funded it will be shown to the recipient. In contrary with the regular wish, which is connected with one user, one more relation exists between the wish and the awardee.

2.1.4 Suggested wish

Suggested wish serves as a helping entity in wish creation. When a user wants to create new wish he may use interactive search that will suggest him several products depending on his search text. The user has an option to choose one of these suggestions which will result in prefilling corresponding fields with data. It makes the process of wish creation much more simple than manual writing.

2.1.5 Comment

Comment is an entity that stores user's comments. Each comment belongs to a certain wish. It is possible to write as many comments as the user wants. Comments should have authors. That is why it is connected to the user entity. All comments are sorted by their creation date which is a property of the comment.

2.1.6 Donation

Donation entity reflects user's contribution to some wish. Every time that user has donated his money the donation entity was created. It serves as a connecting point between the user and the contributed wish. This approach simplifies listing of donations belonging to some user or received by a certain wish. Donation amount plays its role in data mining after ElateMe application will be launched, because it helps to determine user preferences in donating.

2.1.7 Advertisment

This entity was planned to be included in the final release of ElateMe application, but it was postponed until the next one. It had two properties: advertisement id and zone id. First one was used to uniquely identify each advertisement. The second one should have named a zone in the application where an advertisement could be shown. Depending on the advertisement size several types of zones were possible. Some of them could be integrated into user's feed or wish creation. The mobile application would download the list of advertisements and show them at appropriate places.

2.2 Requirements

This section lists all functional and non-functional requirements that were designed before the start of development. Some of these requirements refer to whole ElateMe application, but some refer only to the iOS version.

2.2.1 Functional requirements

Login and registration

F1: Login with Facebook account The user will be able to access ElateMe application using his Facebook account.

F2: Logout from Facebook account The user will be able to log out from ElateMe application.

F3: Store credentials to keep user logged in The user will be automatically logged in after reopening ElateMe application.

Wish management

F4: Wish creation User will be able to create his own wish.

F5: Wish completion (success) The user will be notified that his wish was completely funded.

F6: Wish completion (failure) The user will be notified that his/her wish was unsuccessful, and the desired sum of money was not gathered.

F7: Wish deletion The user will be able to delete the wish that he created.

F8: Wish suggestion ElateMe application will be capable of suggesting wishes based on the user's input. The user could create the same wish that already exists.

F9: Show user's wishes The application will be able to show all wishes belonging to the current user.

F10: Show friend's wishes The application will be able to show all wishes belonging to the user's friend.

F11: Show wish detail The application will be able to show wish details.

Donation management

F12: Send donation User could donate specified sum of money to support his friend's wish.

F13: Show user's donation The application will be able to show the list of all donations made by the current user.

F14: Show wish donations The application will also show the list of donations related to a certain wish.

Friend management

F15: Show user's friends The application will show the list of user's friends from Facebook who have also installed ElateMe application.

F16: Create friend lists It will be possible to create friend lists for privacy purposes.

Comment management

F17: Comment wish User could comment on his friend's wish.

F18: Comment deletion Application will let user to delete his/her comment.

F19: Show comments The application will show the list of comments related to some wish.

Settings

F20: Show user's personal information Application will show user's personal data including name, surname, and photo.

F21: Add bank account number The user will be able to add his bank account number to simplify the donation process.

2.2.2 Non-functional requirements

N1: Native iOS application Application will be written using native iOS SDK.

N2: iOS version 9.0 support Application will support all iOS versions starting from 9.0.

N3: UI in accordance with Apple guidelines [5] The application will tend to follow latest trends and best practices.

N4: Additional language support Application architecture will support multiple languages. It will be relatively simple to add an additional language.

N5: Scalable application architecture Application architecture will be able to scale and integrate future modules into the current solution.

N6: User-friendliness The application will be user-friendly allowing its users to accomplish their wishes without complex instructions.

N7: Error handling The application will support handling non-standard situations like disconnected internet or lack of memory.

2.3 Platform-independent model

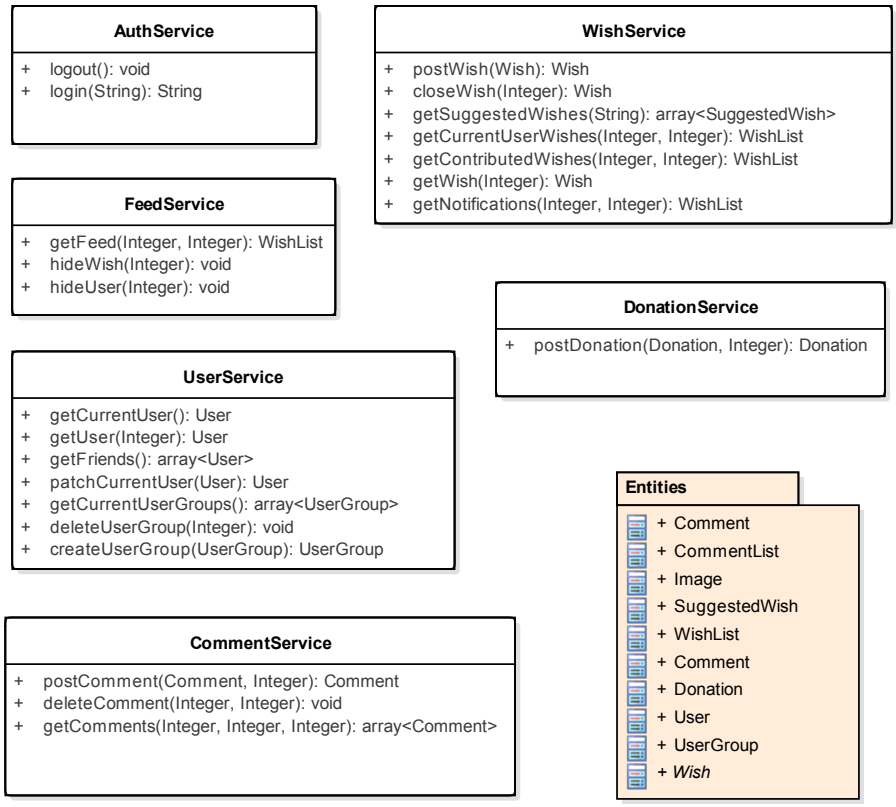


Figure 2.2: Platform-independent model

The main purpose of the current section is to tell about the architecture of the application. Platform-independent model visualizes software system in such a way that no constraints regarding specific implementation are taken into consideration. This type of model is used in a model-driven architecture together with the platform-specific model. Such an approach was structurized and documented by an Object Management Group [6].

The PIM diagram in **Figure 2.2** shows several services that are used to connect UI and the API throughout the application. Network layer represents a substantial part of ElateMe application and will be described in a greater detail than UI. Each network service has its own responsibility. The initial intention of dividing logic into different services was to make classes smaller and to provide an interface only for one type of entities. This decision greatly simplified network communication because usually one singleton class handles all interactions between the mobile application and the API, and it becomes complicated to make changes and extend functionality. Besides that, ElateMe

is a long-term project, which makes time investments into well-thought-out architecture negligible in comparison with the time required to change the heavily dependent code in the future. There are detailed descriptions of each service down below.

2.3.1 Authentication service

This service manages user authentication to access ElateMe API. ElateMe uses OAuth 2.0 protocol [7] for authorization. The main goal of the service is to provide facebook token to ElateMe server in order to get application token which will grant access to other requests. Nevertheless, authentication service serves only as an interface to the lower network layer and does not store any tokens. It is also used to log out the user from ElateMe application and send the corresponding request to ElateMe API.

2.3.2 Wish service

Wish service is responsible for wish management. It includes wish creation, deletion, and getting more detailed information about the particular wish. Wish detail consists of additional information like a description. In addition to this functionality, wish service acts as an interactor to get different lists of wishes. Those lists are wishes of the current user, contributed wishes, and suggested wishes. Moreover, the service is capable of getting notifications, which keep track of user's activity.

2.3.3 Feed service

To realize feed management functionality this service was created. It enables the mobile application to request a smart list of wishes that is provided by ElateMe API. Additionally, it is possible to hide some user or some particular wish, so they will never be shown again in the feed. Most of the display logic lays on ElateMe API because the mobile application will show wishes in the exact order as they were downloaded.

2.3.4 User service

User service works with users and user groups. It helps to download data for the current logged in user or any other available user. It could also get user's friends and update the current user with some changed properties. Further functionality includes user group management. Each user may add his friends into different groups. He could create new groups or remove old ones. In the future it will be possible to modify group data in the same way as the user data modification works.

2.3.5 Comment service

This service allows getting comments for some specific wish. Other functionality includes comment creation and comment deletion. Comment editing is planned for the next development phase. As an extra feature, it might be considered beneficial to enable commenting to other comments, not to the wish itself.

2.3.6 Donation service

The last service maintains payments and donations. For sure, it will be improved in the future because it plays the core role in ElateMe application. Now it only sends the donation amount, so that ElateMe API could register a donation. To work as intended complete transaction scheme should be known. All normal payments should be realized via "FIO banka", but there are several difficulties, which are described in the next section.

2.4 Payments via the FIO bank

In order to conduct direct or credit card payments, it is necessary to use a third-party system that is certified for conducting transactions or to certify your own. The latter variant is overcomplicated, so the first option was chosen. There are several businesses that provide payment services. Most of them are so-called payment gateways, and they are managed by either a bank or a facility that cooperates with some bank. A lot of these services gather substantial fees for every transaction made. Our intention was to find the best possible service for our needs. After tedious search, the solution was found in the form of "FIO banka" [8]. It is a bank with headquarters in the Czech Republic, and it provides required services for a reasonable sum of money. It uses a payment gateway [9] for its online transactions. To implement payments on the client side it is mandatory to study the documentation, which is not freely available in its full form. It is compulsory to sign a contract with "FIO banka" to obtain this documentation. The whole process of signing the contract took from two to three month. Such a delay made it impossible to integrate credit card payments into the mobile application. Based on some research the concept of interaction between the mobile client and the payment gateway was developed to save future time. It could be found in the next chapter.

Chapter 3

Design

This chapter contains the concept of interaction with the payment gateway and in-depth look into the network layer architecture. To make platform-independent model suitable for the real mobile system it should be translated into the platform-specific model. All the needed changes will be described in **Section 3.2**. It would be misleading not to mention that ElateMe iOS application is written in Swift [10], which is a relatively new language compared to Objective-C, but still very powerful. Therefore architecture design was heavily inspired by core Swift features.

3.1 Payment gateway interaction

Figure 3.1 represents the concept, which was developed in cooperation with Yevhen Kuzmovych [1]. It illustrates the communication between the mobile client, ElateMe API, and the payment gateway of "FIO banka". This flow starts after a user presses "Donate" button in the mobile application. He then chooses a suitable amount of money he would like to donate. Different type of currencies may be added to this stage later. For now, all transactions are in Czech crowns. All data is sent to ElateMe API, where the server creates an unconfirmed donation to track its status. The server responds with donation information, which is used by the mobile client to initialize transaction with the payment gateway. The web view is opened in the next step with previously received data. Then the user will follow instructions in the web view and confirm his payment. After that, the payment gateway will conduct this transaction in two possible scenarios. The user will be redirected to the success URL of ElateMe API, where the API will confirm successful donation, or to the failure URL, where the donation will be removed. In the last step, the user will be notified whether his donation succeeded or not. This concept will be implemented as soon as full "FIO banka" documentation becomes available.

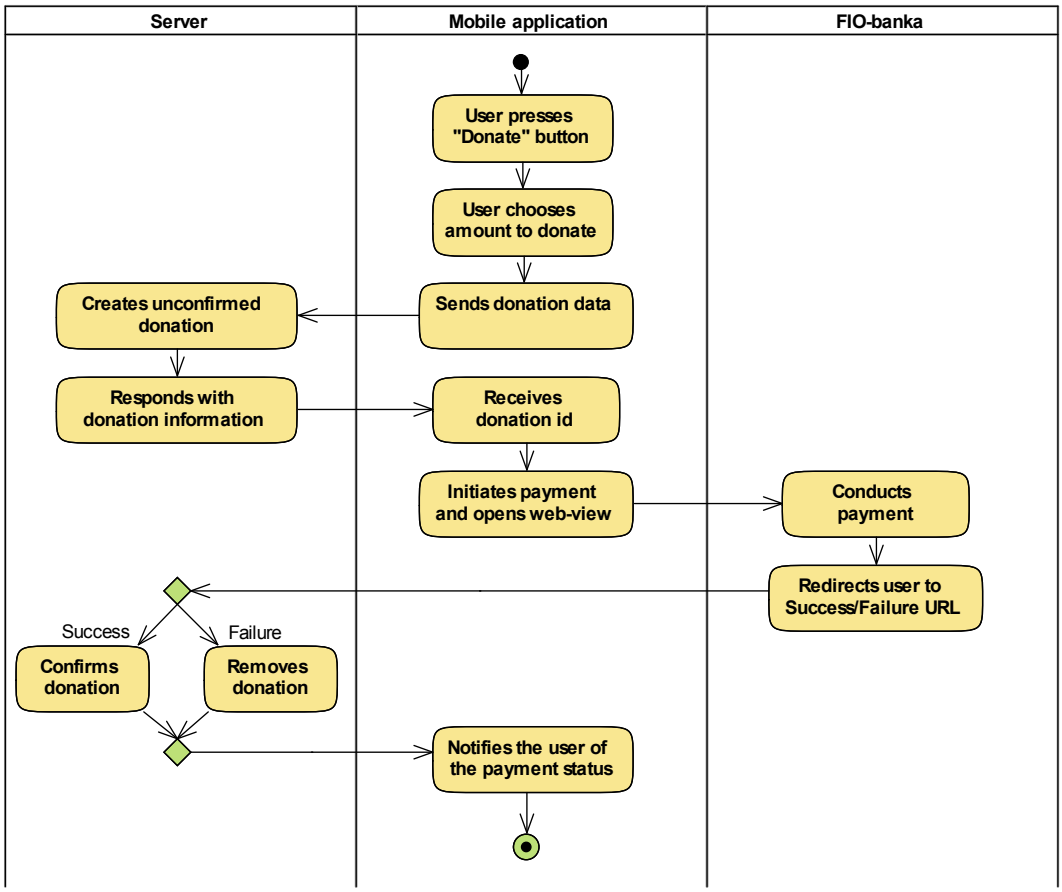


Figure 3.1: Payment gateway interaction. Source: [1]

3.2 Platform-specific model

Figure 3.2 and **Figure 3.3** show entities belonging to the PSM. This model usually is changed slightly from PIM to suit programming language and platform requirements. Not all changes are illustrated in the diagrams because it would make them unreadable. From the title of this bachelor's thesis, it is evident that the mobile client uses iOS from Apple, and ElateMe is written in Swift. Curious reader may learn more about the application development on Apple developer portal [11].

Network services should be changed in a special way to work as intended. To observe these changes it is better to look at an example. Donation service is a class that has a method with the following signature:

```
func postDonation(amount: Int, wishID: Int) -> Donation
```

This method will call underlying network layers, which are responsible for conducting URL request. Because it takes time to complete the request, this

method will block the UI, and a user will experience "freezed" screen. To avoid such performance issue, it is necessary to make this method non-blocking. The signature will be changed to

```
func postDonation(amount: Int, wishID: Int,
     success: @escaping (Donation) -> (),
    error: @escaping (Error) -> ())
```

Additional two parameters are added, and their type is a *closure*. The closure is a part of Swift features, but the reader may be more familiar with a term *callback*. Basically, they are a delayed code that will be called when the request is completed. The request may either succeed and call `success` closure or fail and call `error` closure. Such an approach does not block UI, but it should be used with caution due to the concept of capturing. More information about capturing and `@escaping` could be read in the official Apple guide [12]. All other services also receive additional closures to their methods.

Domain model entities should be transformed to Swift `struct` type. Possible transformation of the `Donation` entity may look like this.

```
struct Donation {
    let amount: Int
    let dateString: String
    let donationID: Int
    let donatorID: Int
    let wishID: Int
}
```

All domain model entities are converted in the same way. There are also several new entities in **Figure 3.3**. `CommentList` and `WishList` are used to simplify pagination. These entities serve as wrappers for an array of comments or wishes. Another interesting entity is `UserGroup` that represents several friends united as a group with some certain color. Each user could create such groups with different friends and make his wishes visible only for certain groups.

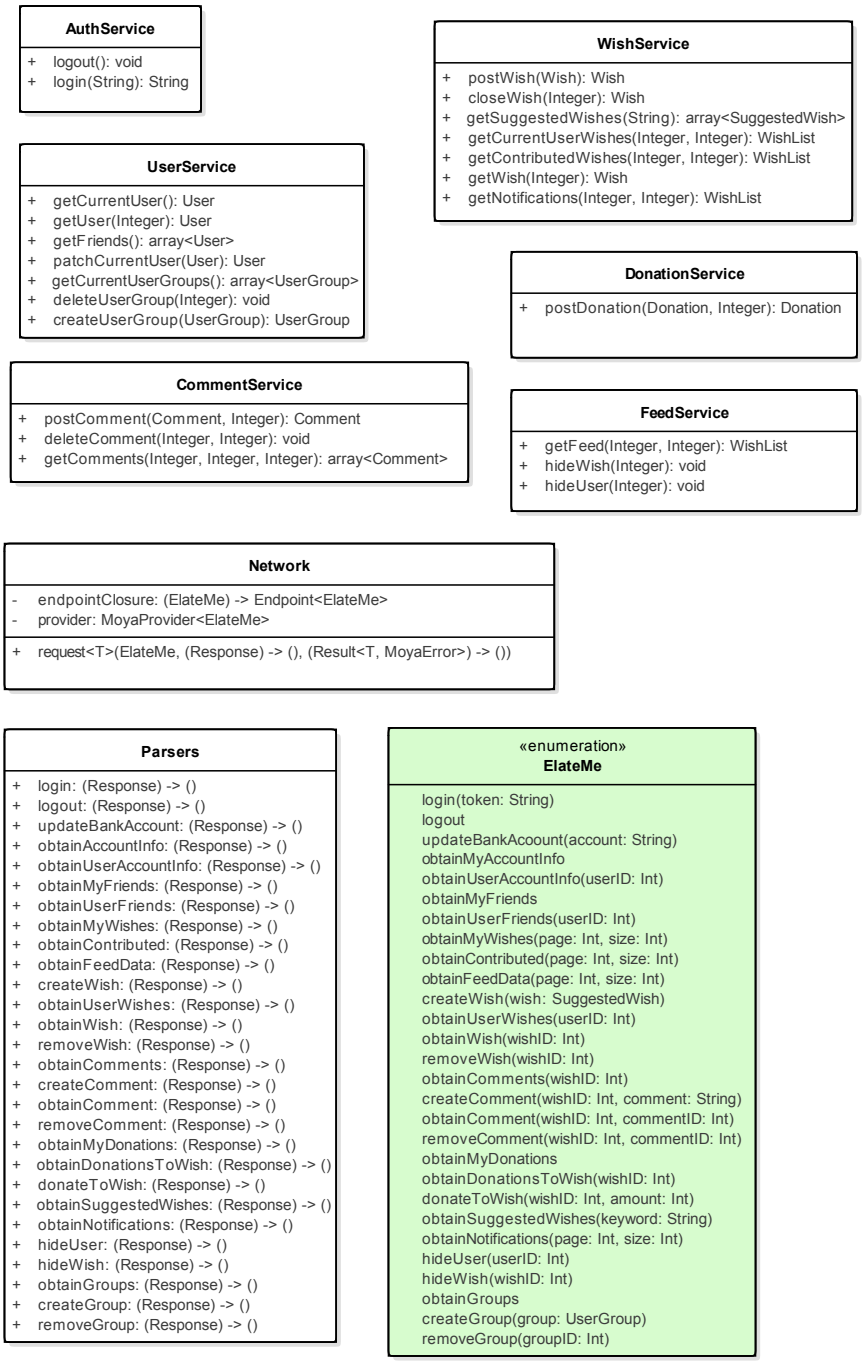


Figure 3.2: Platform-specific model

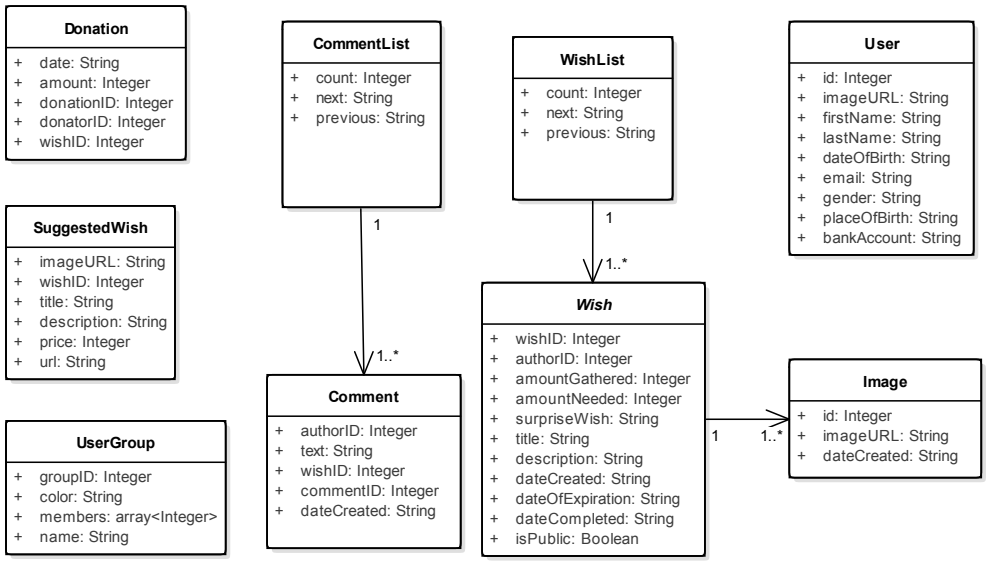


Figure 3.3: Entities

3.2.1 Moya network layer

To make architecture cleaner Moya framework [2] was used in designing the network layer. It will be explained later in this section. On the low level of networking, the only way to communicate with ElateMe API is an HTTP request. It is an application protocol that specifies the standard of data communication on the world wide web. ElateMe API uses five types of requests:

- **GET**
  The mobile client uses this request to download data and show them on the screen.
- **POST**
  This type is used to upload new data to the server, for instance during wish creation.
- **PUT**
  It is used to overwrite existing resource on the server.
- **DELETE**
  This type deletes some resource from the server.
- **PATCH**
  This request type is utilized to send partial data that will update existing resource.

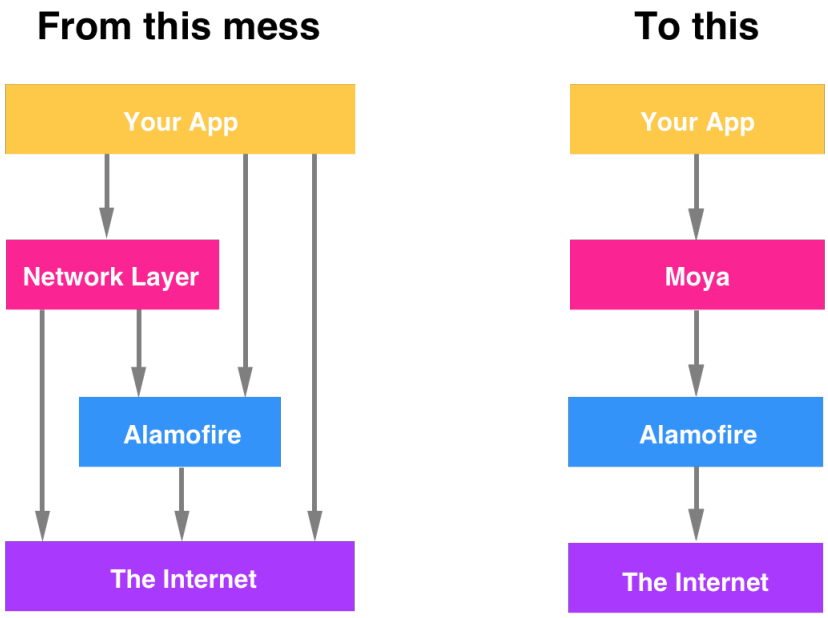


Figure 3.4: Moya framework scheme. Source: [2]

Each HTTP request with a body has its body formatted to JSON in ElateMe. JSON [13] is a convenient data format for sending and receiving data. To send data requests `URLSession` and several other classes exist in Foundation native framework, but working with it requires a lot of boilerplate code. The solution was found in a group of frameworks, which are illustrated in **Figure 3.4**. One of them is Alamofire [14] that makes a wrapper for `URLSession`. The second one is Moya that manages request parameters, URLs, and encoding in a neat way. There are three more significant classes in **Figure 3.2** involved into network communication.

- `Network`
  This class interacts with Moya framework to register network request.
- `Parsers`
  This class is used to parse data from JSON format to the convenient Swift structs.
- `ElateMe`
  It represents a Swift enumeration that is processed by Moya to determine URL, headers, request type, and encoding.

The designed network flow starts with UI action or retrieving data for UI. Then the corresponding service is called, which calls `Network` using appropriate

parser from `Parsers` and enumeration case from `ElateMe`. After that, Moya builds URL request and passes it to Alamofire. When the request is finished, `success` closure or `error` closure is called updating UI.

Chapter 4

Implementation

The chapter shows few insights and describes common use cases. Mobile client on iOS is also being developed by Yegor Terokhin [15], who covers a lot of development aspects in his bachelor's thesis. His thesis is written in Czech language. The work of creating ElateMe client was divided into two parts, so each one of us developed his own part, and then the changes were merged together. ElateMe application was developed in accordance with the design that was given to us in the form of screen mockups.

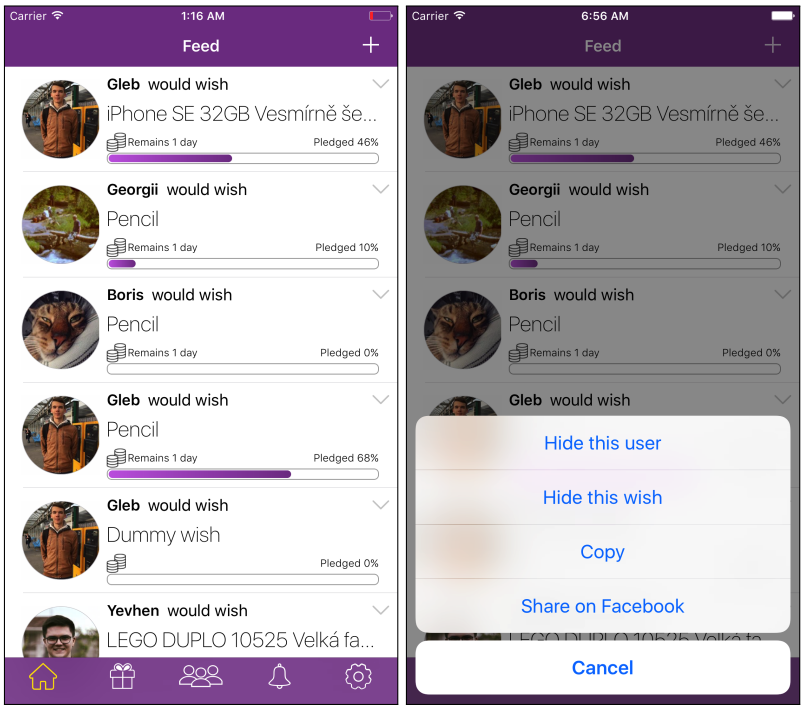
4.1 Development setup

It is crucial to use right IDE for developing the mobile application. In case of iOS, it is better to use official tools for the development. Apple has developed its own IDE called *Xcode* [16]. There are a few issues with it, but overall performance and functionality make it unique, so *Xcode* was our choice. To manage dependencies with various libraries and frameworks, dependency manager *CocoaPods* [17] was used. In order to organize collaborative work with Yegor Terokhin, the *Git* [18] system was installed. This setup enabled us to develop ElateMe application independently from each other, so we could add new features gradually and combine them if needed. To build ElateMe application, the instructions are provided in **Appendix D**.

4.2 Feed management

Feed screen shows different wishes of user's friends that received a donation recently. Feed appearance is visible in **Figure 4.1a**. Each user has several actions he could execute from Feed. One of them is wish creation that is described in **Section 4.4**. If the user clicks on the small arrow at the top right corner of the wish, he will be presented with an alert view. This alert view enables him to hide some specific wish from Feed or to hide some user. It

also allows to create new wish based on another one, and the user could share information about any wish on Facebook, what is described in **Section 4.6**.



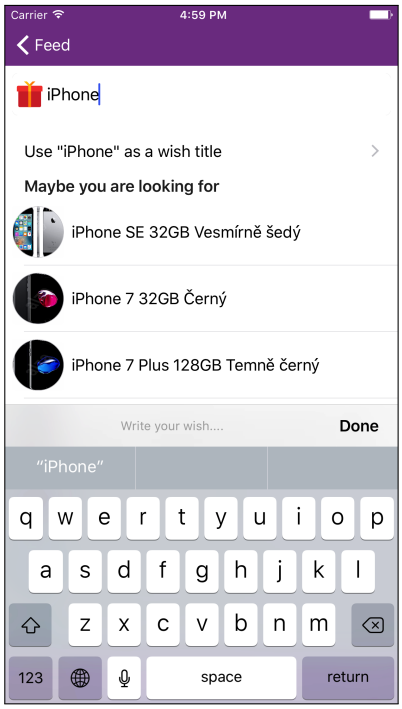
(a) Feed (b) Feed actions

4.3 Donation via the FIO bank

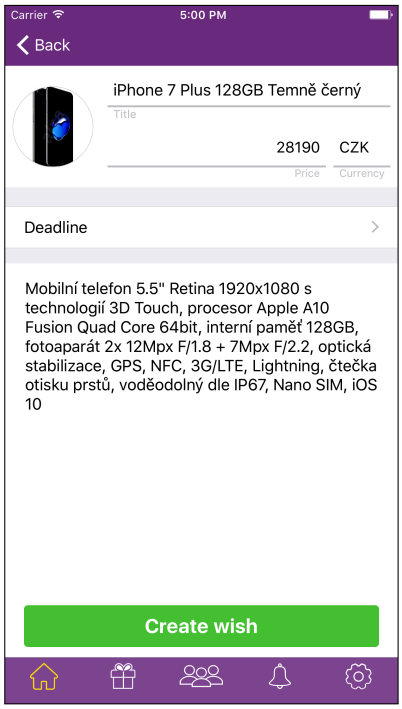
Donations using debit or credit cards were only created as a concept because of unforeseen circumstances. More about that is written in **Section 2.4** and in **Section 3.1**.

4.4 Wish management and suggestions

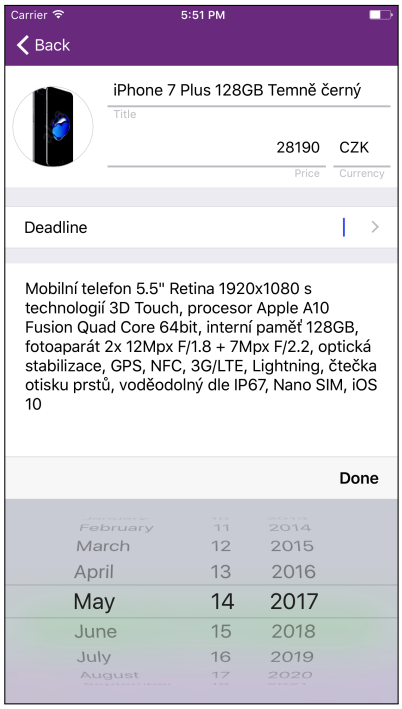
Wish management consists of wish creation, hiding, and sharing. Wish could be created in three ways: by copying someone's wish, by creating it manually, or by using a suggestion. The first option is available from Feed as seen in **Figure 4.1b**. To create a wish manually the user should click plus icon in Feed. He will be presented with a search field to write a keyword. If he chooses to skip this step, he will fill the fields of the wish by himself. Another option is to start writing to the search field to get a list of suggestions. **Figure 4.2a** illustrates this process. If the user chooses some suggestion, all fields will be prefilled as in **Figure 4.2b**. Suggestions are produced by ElateMe API based on the received keyword.



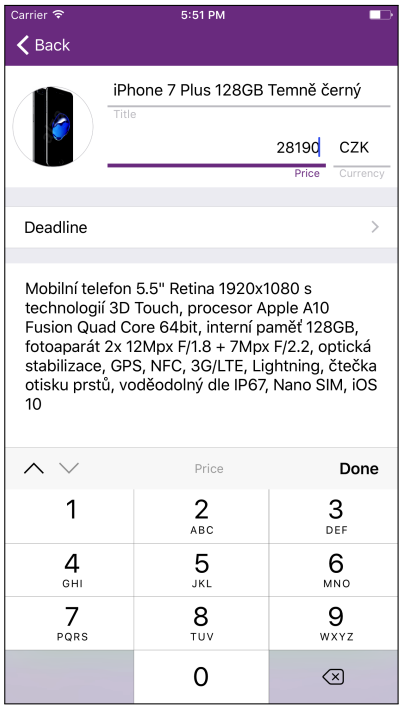
(a) Wish suggestions



(b) Wish creation



(c) Wish deadline



(d) Wish editing

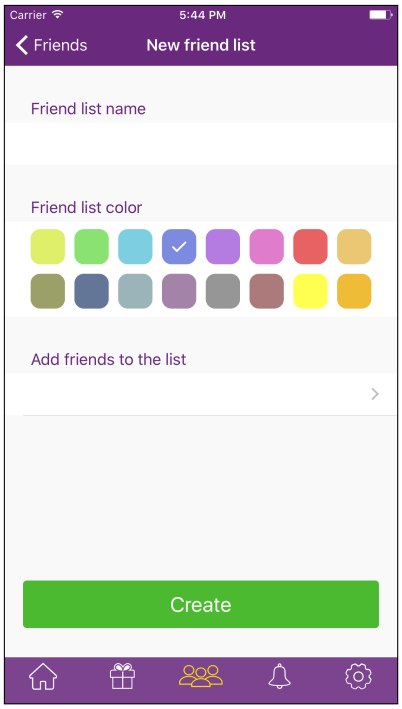


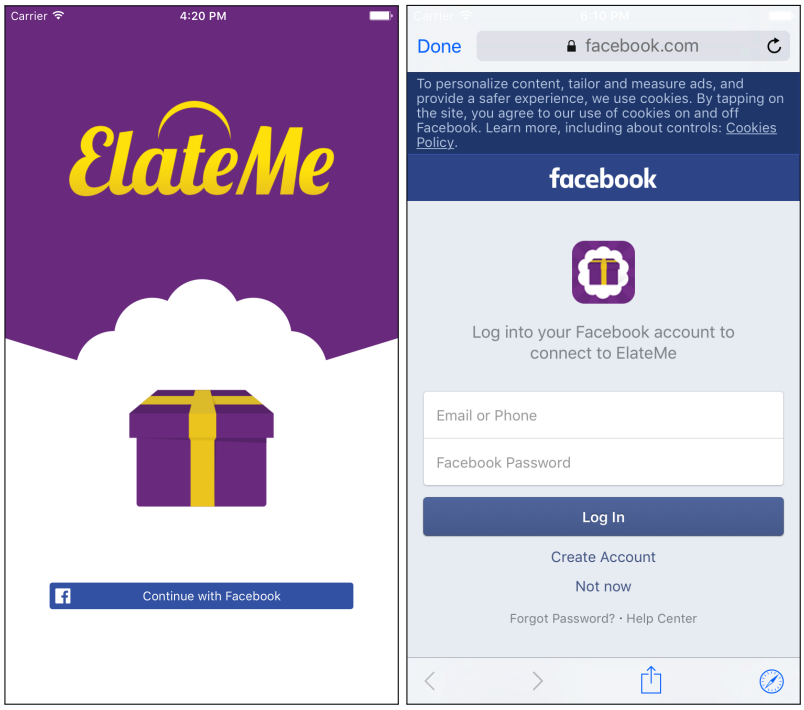
Figure 4.3: Friendlist creation

4.5 Friendlist management

Friendlist management includes creation, editing, and removal of friend lists. Friend lists are essentially groups of users, which are created to simplify wish sharing between people. This feature was not completely implemented due to the lack of time. Using the screen showed in **Figure 4.3** the user could create a friend list. He would choose a name for his friend list and select everyone he wants to add. Then he chooses a color for this list and confirms creation. All the required infrastructure for friendlist creation is already done, but several UI elements are missing. Although one friend could be located in multiple lists, it is not clear how to show such users. An interesting challenge was to create color selection cell because no similar default element exists. There are two solutions that come to mind: using `UICollectionView` with custom cells and using `UIStackView` to embed views. The latter variant was chosen. `UITapGestureRecognizer` was added to each colored view to show checkmark on tap.

4.6 Common use cases

To show more examples of common use cases, several of them are described in this section. Many of the next use cases involve Facebook iOS SDK [19].



(a) Login (b) Login via Facebook

It provides a lot of useful methods to communicate with Facebook API and adds few UI elements. The only downside is that it is impossible to control some default behaviour, but Facebook SDK supports classes to implement custom independent solutions.

- **Login**
  The only option available for now is login via Facebook. The user needs only to press Facebook button in **Figure 4.4a** and follow instructions.

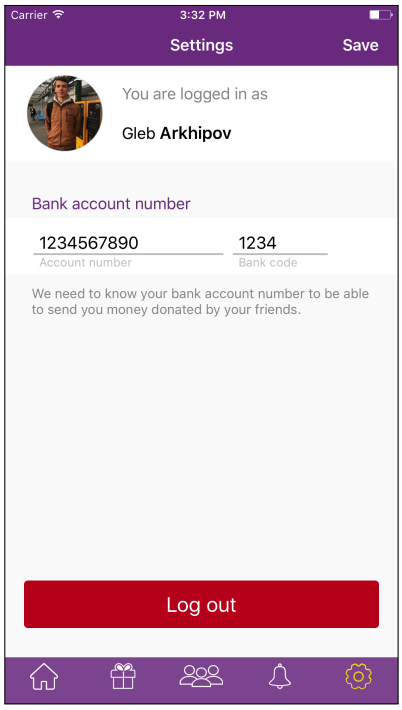
- **Registration**
  There is no such term *Registration* in ElateMe. After first successful login via Facebook, the user will be automatically registered.

- **Logout**
  It is possible to return back to login screen if user clicks logout button in **Figure 4.5a**.

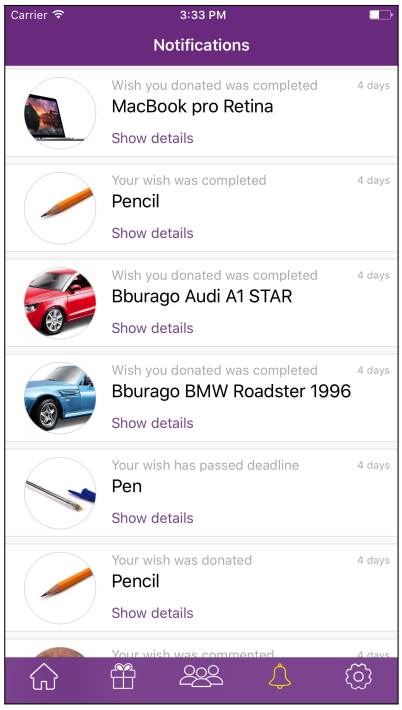
- **Settings**
  The user may change his bank account in **Figure 4.5a** by editing and saving it.

- **Notifications**
  Notifications show a list of recent activity. They are displayed in **Figure 4.5b**.

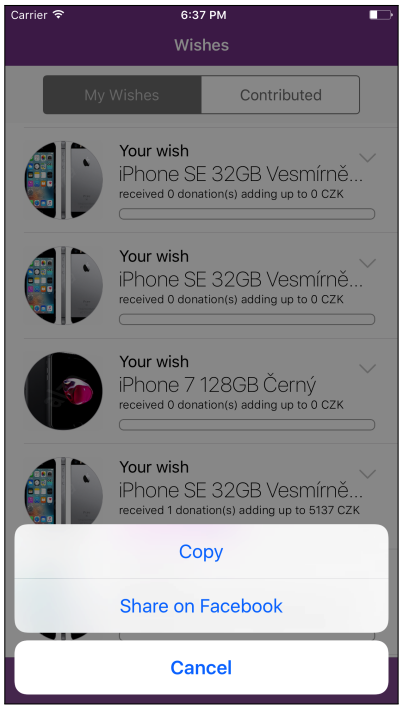
- **Wish sharing**
  To share a wish on Facebook the user taps on a small arrow in the right corner of the wish and chooses appropriate option showed in **Figure 4.5c**. He is presented by a web view as in **Figure 4.5d**.



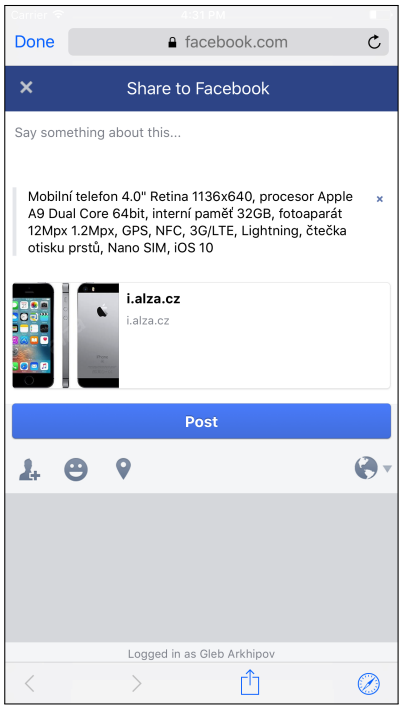
(a) Settings



(b) Notifications



(c) Wish sharing. Step 1



(d) Wish sharing. Step 2

Chapter 5

Testing

This chapter describes usability testing. Usability testing is a research that is conducted to find possible flaws in UI and to decide whether an application is suitable for the target audience, or not. Testing chapter includes testing goals, the creation of pre-test and post-test survey, test setup in the controlled environment, and the whole process of testing. Besides that, all results will be evaluated in the final section.

5.1 Testing goals

There are several main goals that usability testing helps to achieve. One of them is to observe participant's reactions to ElateMe application and UI elements in particular. By measuring participant's time to complete different tasks it is possible to make adjustments to UI/UX making it more user-friendly. Participants could also provide valuable feedback regarding different features they would like to see. Usually, their commentary about existing interface is also crucial, since they may use this application in the future.

5.2 Pre-test and post-test surveys

Pre-test survey is used to gather additional information about each participant. It also helps to explain participant's actions during testing. Each of the participants will answer pre-test questions and proceed to the testing itself. The post-test survey takes place right after testing, and it mainly serves as a feedback. The most important question in the post-test survey is about missing functionality in ElateMe application. It helps to realize what features should be implemented before the official release. All questions could be found in **Appendix B** and all answers are illustrated as charts in **Appendix C**.

5.3 Test setup

Usability testing took place in the SAGElab [20]. It is a collaboration project between CESNET, the FIT, and the FEE at the Czech Technical University. Its main aim is to support research in branches of visualization and network technologies. One of the SAGElab subprojects is a usability lab that could record audio/video data and transmit it to the large-scale visualization wall.

Testing setup of the usability lab, which was used in our testing, consists of two rooms. One room contains three cameras, one microphone, and a testing device (mobile phone). The first camera records what happens on the device. The second camera records participant's face, and the third camera records general view. This is the room where participant sits and performs given tasks which are read aloud by a moderator. The main job of the moderator besides reading is to assist participant if something goes wrong. Moderator's role is also important from the psychological point of view because the participant has another person to speak with. Participant is allowed to speak freely and express his thoughts about this or that task.

Another room holds visualization wall that is used to observe how participants are doing predefined tasks. Live stream of audio and video from the first room is transmitted to this room. While the test is in progress, developers have time to make notes. All data is thoroughly recorded and is available for further analysis after the testing ends.

5.4 Task list

In order to accomplish testing goals, task list was designed to cover basic use cases in ElateMe application. Task list essentially consists of two parts: wish creation and donation. In the first part, each participant tries to create his own wish based on some interactive suggestions. After that, he makes a donation to a certain friend. Down below is the task list itself.

1. Login via facebook if it is required.
2. Check notifications.
3. Realize that your wish "Pencil" was completed and "Pen" was not.
4. Return to Feed and click wish creation button.
5. Use the search field to find suggested wishes (for example: "iPhone").
6. Select one of the items offered.
7. If some field does not suit you, change it.
8. Set expiration date to 15.05.2017.

9. Confirm wish creation.
10. Go to Wishes screen.
11. Find newly created wish and share it on facebook (use small arrow).
12. Go to Settings and change your bank account number to 1234567890/1234.
13. Find a wish titled "Macbook pro retina" made by your friend "Jan Novak" (Go to Feed).
14. Open wish detail.
15. Donate any amount to this wish.
16. Leave a comment below.

5.5 Testing process

There were two groups of five participants in each one: Android and iOS. Android group is analyzed in the bachelor's thesis of Georgii Solovev [21]. Testing process began with filling out pre-test survey. After it was done each participant and a moderator proceeded to testing ElateMe application on a mobile device. During the test, the moderator had read one task letting participant complete it. The participant was expressing his ideas while doing this task. At the same time, all developers were making notes to record possible problems and suggestions. When the test was finished, each participant needed to fill out a post-test survey. That concluded usability testing.

5.6 Evaluation

Some of the prepared tasks in our scenario took longer time than expected, so it will be better to provide alternatives or rethink the existing way of things. Each found problem was assigned with one of the three priorities:

- High — problem needs to be solved before the release
- Medium — problem better be solved before the release
- Low — problem could be solved before or after the release

To summarize all problems happened during the process of testing a list of problems with corresponding priorities was created.

- **Notifications are not intuitive**
  Priority — High

  Description
  It took a lot of time for some participants to find notifications. Maybe not all were familiar with this term or notification icon was resembling something else, for example alarm.

  Possible solution
  There are several ways to solve this problem. The simplest one would be to change the icon or to add titles below the icon. More complicated solution involves making a graphical tutorial to acquaint the user with the application.

- **Save button was not pressed**
  Priority — High

  Description
  Several people have not pressed save button while changing their bank account number.

  Possible solution
  It is necessary to add dialog that user leaves screen with unsaved data.

- **Blank images in Notifications**
  Priority — Medium

  Description
  Some of the images were not loaded during usability testing. Usually asynchronous loading works as intended.

  Possible solution
  It could have happened because asynchronous loading uses a third-party framework. To solve this problem loading should be written without frameworks or at least some default image placeholder should be provided.

- **Wish detail does not refresh**
  Priority — Medium

  Description
  When user donates to some wish, he is returned back to wish detail, but wish detail does not update itself automatically. Changes are only visible after next opening of wish detail.

Possible solution
Add asynchronous update after donation.

- **Small arrow to share wishes**
  Priority — Low

  Description
  When user wants to share some wish, he needs to tap a small arrow. This tap may result in opening of wish detail instead of sharing dialog.

  Possible solution
  Make arrow icon larger.

- **Accidental press of logout button in Settings**
  Priority — Low

  Description
  If user presses logout button accidentally, he will be redirected to Login immediately.

  Possible solution
  It is required to add confirmation dialog to the logout button. This will prevent users from accidental logout.

All of the problems described above could be solved in a reasonable amount of time. Participants also suggested some new functionality that may be added in the future. A lot of people would like to see search and filtering in Feed. Some suggested adding other translations for the application, but it is already planned. One more suggestion was to add payment confirmation to the donation dialog. All these suggestions are taken into consideration and will be implemented afterwards. To sum up, usability testing provided crucial feedback from participants or maybe even future users that is why it was important to invest time and conduct one.

Conclusion

In order to evaluate ElateMe project, it is crucial to recap key points. Overall satisfaction with this project is high. A lot of planned work was done in time. This project was complex from the architectural point of view, which enhanced my skills in designing and implementing the architecture. On the one hand, it was a challenge for me to collaborate with my team because it was my first team software project, but on the other hand, it gave me a priceless experience of handling different situations and managing schedules. Besides that, a few custom UI elements were implemented based only on their look. This process of creation vastly improved my programming skills.

If similar project started today, I would do certain things differently. For example, more time should be dedicated to a project of such scale. Moreover, time management and task division could also be improved. One more thing that comes to mind is continuous integration. It includes a testing of regular builds and a code review. It helps to find bugs and errors in the early stages, but continuous integration only makes sense for two or more developers on one platform.

Finally, for the future development, it is necessary to fix some errors found during the usability testing. Code refactoring would also be beneficial. As an additional testing, it is possible to integrate UI tests in the mobile application. The new website is already being developed by another team, and it will see light in the nearest future. ElateMe project will continue until it will be released into production.

Bibliography

[1] Kuzmovych, Yevhen. *ElateMe - Backend*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2017.

[2] *Network abstraction layer written in Swift. [online]*. [viewed 20 April 2017]. Available from: `https://github.com/Moya/Moya`

[3] Rouse, Margaret. *Crowdfunding. Techtarget [online]*. [cited 17 April 2017]. Available from: `http://whatis.techtarget.com/definition/crowdfunding`

[4] Balatsko, Maxim. *ElateMe - Project management and Advert server*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2017.

[5] *Apple UI guidelines. [online]*. [viewed 16 April 2017]. Available from: `https://developer.apple.com/ios/human-interface-guidelines/overview/design-principles/`

[6] *OMG. Object management group [online]*. [viewed 17 April 2017]. Available from: `http://www.omg.org`

[7] *OAuth 2.0. Authentication standard [online]*. [viewed 16 April 2017]. Available from: `https://oauth.net/2/`

[8] *FIO bank. [online]*. [viewed 17 April 2017]. Available from: `https://www.fio.cz/`

[9] *FIO bank payment gate. [online]*. [viewed 17 April 2017]. Available from: `https://www.fio.cz/bankovni-sluzby/platebni-karty/platebni-terminaly-brana`

[10] *Swift. Programming language from Apple. [online]*. [viewed 16 April 2017]. Available from: `https://developer.apple.com/swift/`

[11] *Apple developer web. [online].* [viewed 16 April 2017]. Available from: https://developer.apple.com/develop/

[12] *Swift closure. [online].* [viewed 15 April 2017]. Available from: https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/Closures.html

[13] *JSON. JavaScript Object Notation. [online].* [viewed 15 April 2017]. Available from: http://www.json.org

[14] *Alamofire. Networking framework for iOS [online].* [viewed 16 April 2017]. Available from: https://github.com/Alamofire/Alamofire

[15] Terokhin, Yegor. *ElateMe - iOS client I.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2017.

[16] *Xcode. Apple IDE for programming. [online].* [viewed 16 April 2017]. Available from: https://developer.apple.com/support/xcode/

[17] *CocoaPods. Dependency manager for Swift and Objective-C [online].* [viewed 17 April 2017]. Available from: https://cocoapods.org

[18] *Git. Version control system. [online].* [viewed 15 April 2017]. Available from: https://git-scm.com

[19] *Facebook iOS SDK [online].* [viewed 16 April 2017]. Available from: https://developers.facebook.com/docs/facebook-login/ios

[20] *SAGElab. Visualization labaratory [online].* [viewed 17 April 2017]. Available from: https://sagelab.cesnet.cz/en/

[21] Solovev, Georgii. *ElateMe - Android client.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2017.

[22] *Google Forms. Survey creation tool. [online].* [viewed 15 April 2017]. Available from: https://www.google.com/forms/about/

Appendix A

Acronyms

JSON JavaScript object notation

iOS Mobile operating system

GUI Graphical user interface

UI User interface

UX User experience

FIT Faculty of Information Technologies

FEE Faculty of Electrical Engineering

PIM Platform-independent model

PSM Platform-specific model

URL Uniform Resource Locator

HTTP Hypertext Transfer Protocol

API Application programming interface

IDE Integrated development environment

Appendix B

Pre-test and post-test surveys

B.1 Pre-test questions

1. Your age
   - 12-16
   - 17-18
   - 19-25
   - 26-34
   - 35-50
   - 50+
2. Your current income
   - up to 10 000 CZK monthly
   - up to 25 000 CZK monthly
   - up to 40 000 CZK monthly
   - greater than 40 000 CZK monthly
3. How do you make gifts to friends?
   - Personally
   - Gather money as a group and buying a gift
   - Prefer money as a gift
   - Other

4. Your experience with mobile phones
    - Experienced user (Any application is simple to use)
    - Medium user (Mobile applications are normally easy to handle)
    - Non-experienced user (You would prefer to call instead of writing SMS)
5. How often do you use your phone?
    - Once a week
    - Once a day
    - 3+ times a day
    - 20+ times a day
    - It never leaves my hand
6. Your experience with mobile applications. (Check all that apply)
    - Facebook
    - Uber
    - Airbnb
    - Booking
    - Instagram
    - Mobile Banking
7. Have you ever bought something with a mobile application?
    - Yes
    - No
8. Have you used bitcoin?
    - Yes
    - No
    - What is bitcoin?
9. Do you have debit/credit card?
    - Yes
    - No

10. How often do you use it?
    - Never
    - Annually
    - Quarterly
    - Monthly
    - Weekly
    - On a daily basis
11. Do you use Android or iOS?
    - iOS
    - Android
    - Other

B.2 Post-test questions

1. Did icons/titles make sense regarding which type of content was in the tab?
    - Yes
    - Other (explain)
2. Would you use ElateMe in the future?
    - Yes
    - No
    - Maybe
3. What features would you like to see in our application?

Appendix C

Test infographics

These infographics present results of pre-test and post-test surveys. They were made with the help of Google Forms [22] which is a web application by Google that helps to make online surveys. Answers in the infographics include participants from both iOS and Android groups.

C.1 Pre-test answers

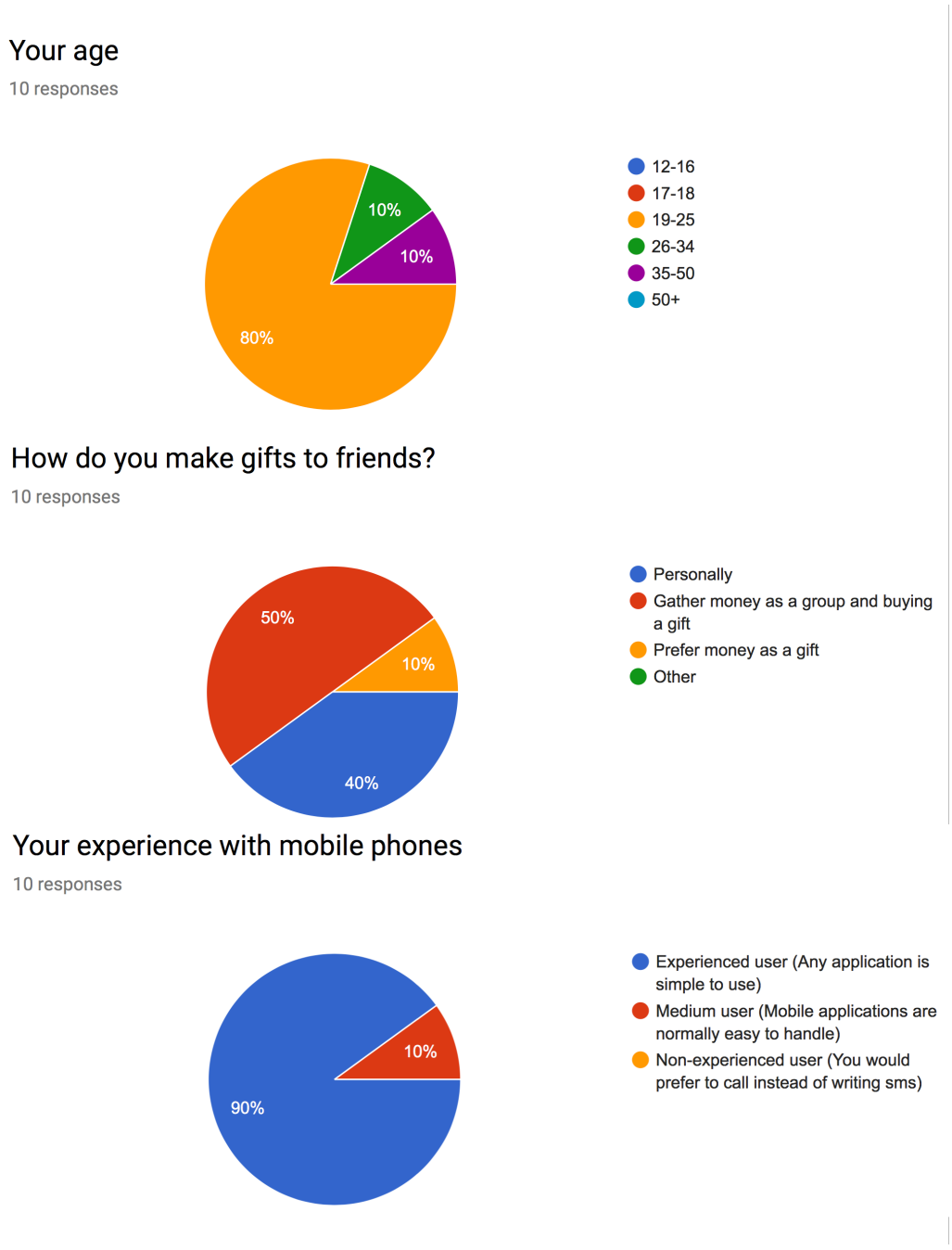


Figure C.1: Pre-test questions 1,3,4

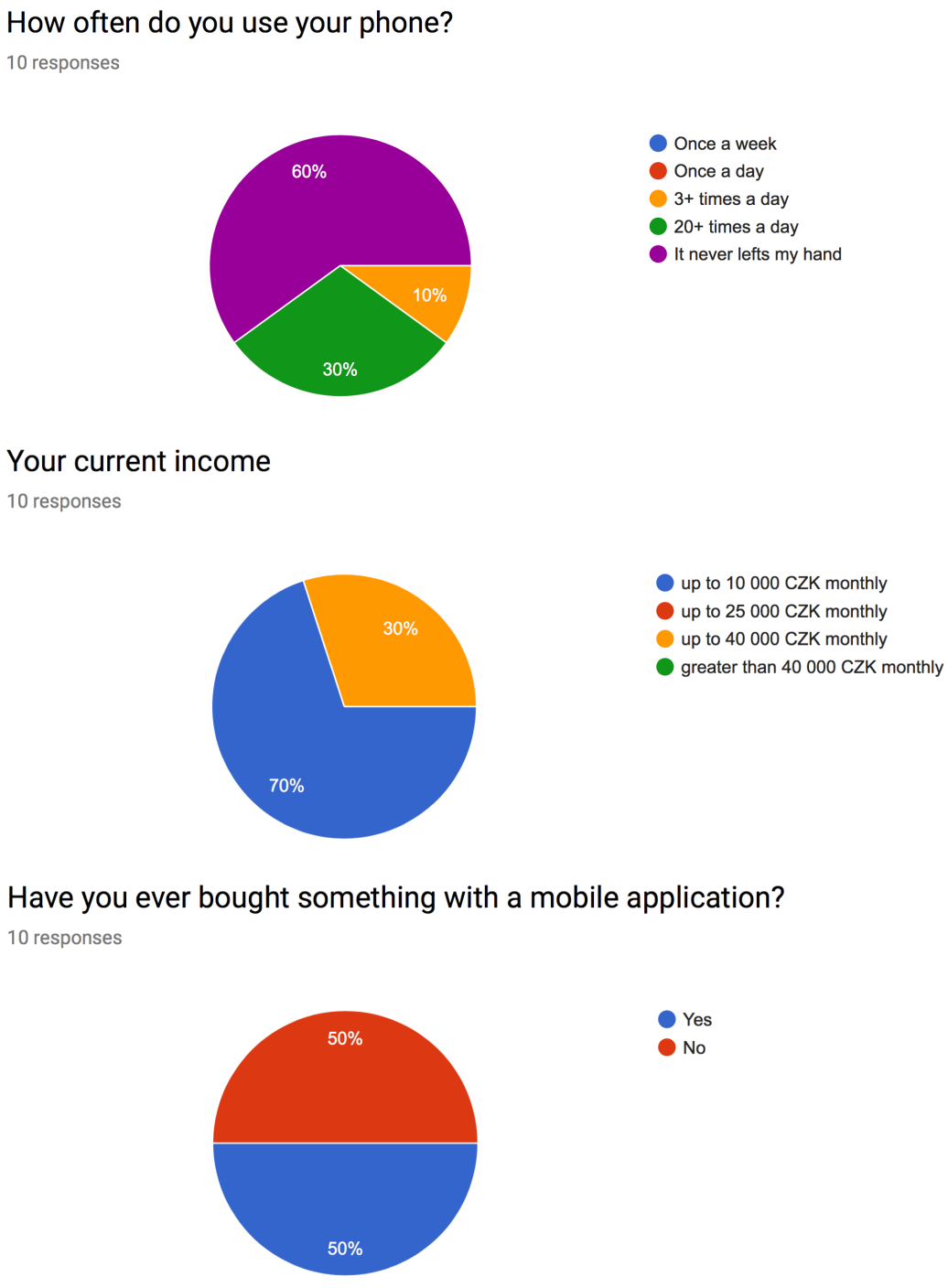


Figure C.2: Pre-test questions 5,2,7

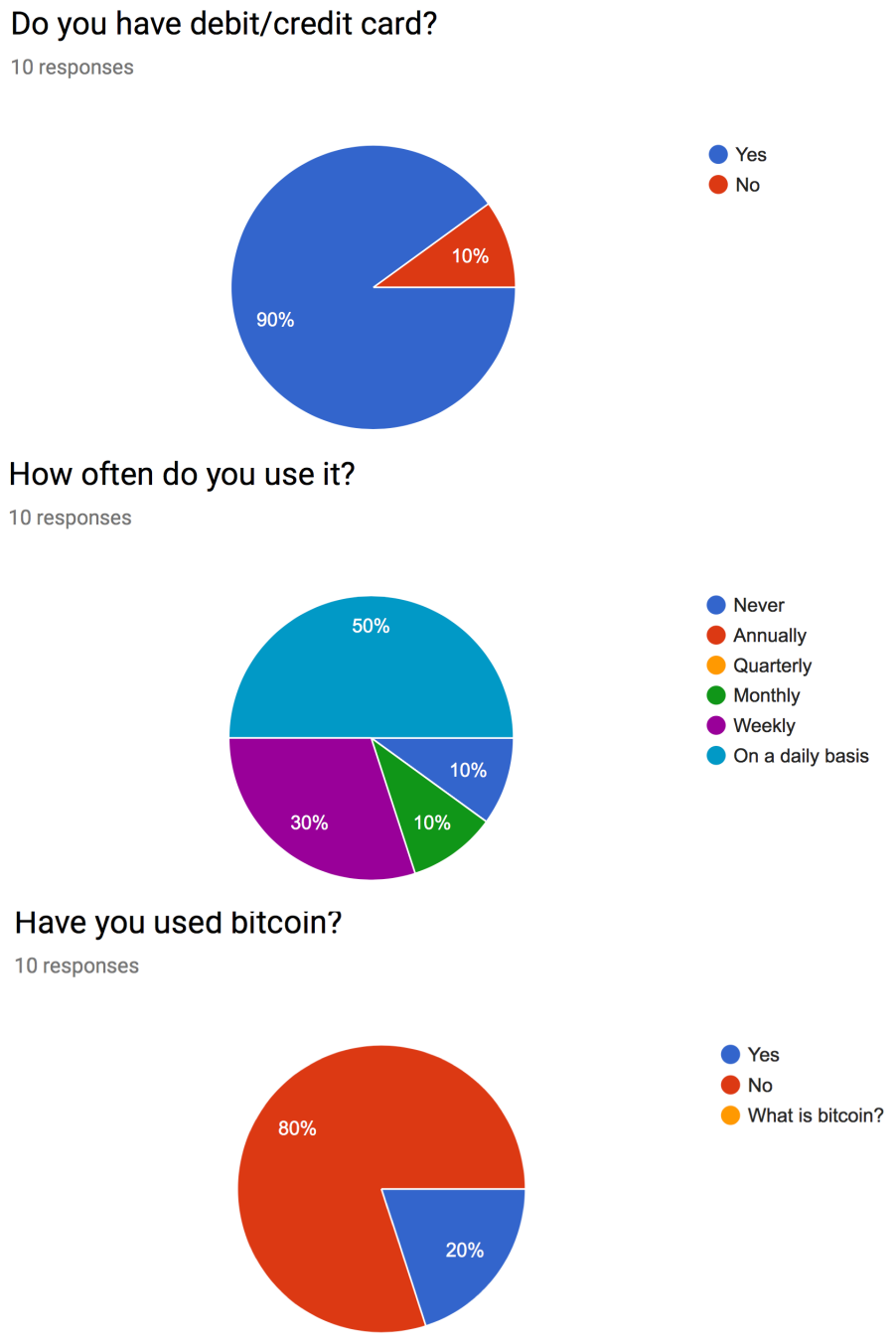


Figure C.3: Pre-test questions 9,10,8

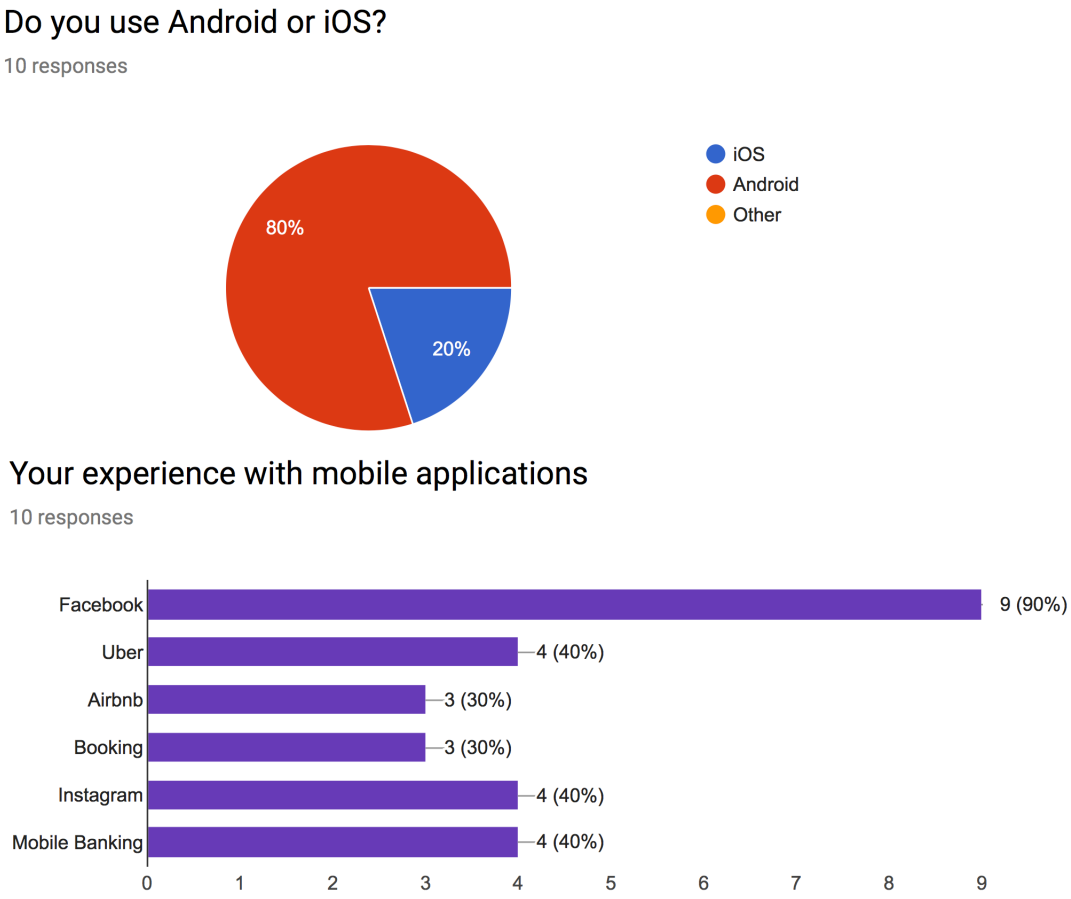


Figure C.4: Pre-test questions 11,6

C.2 Post-test answers

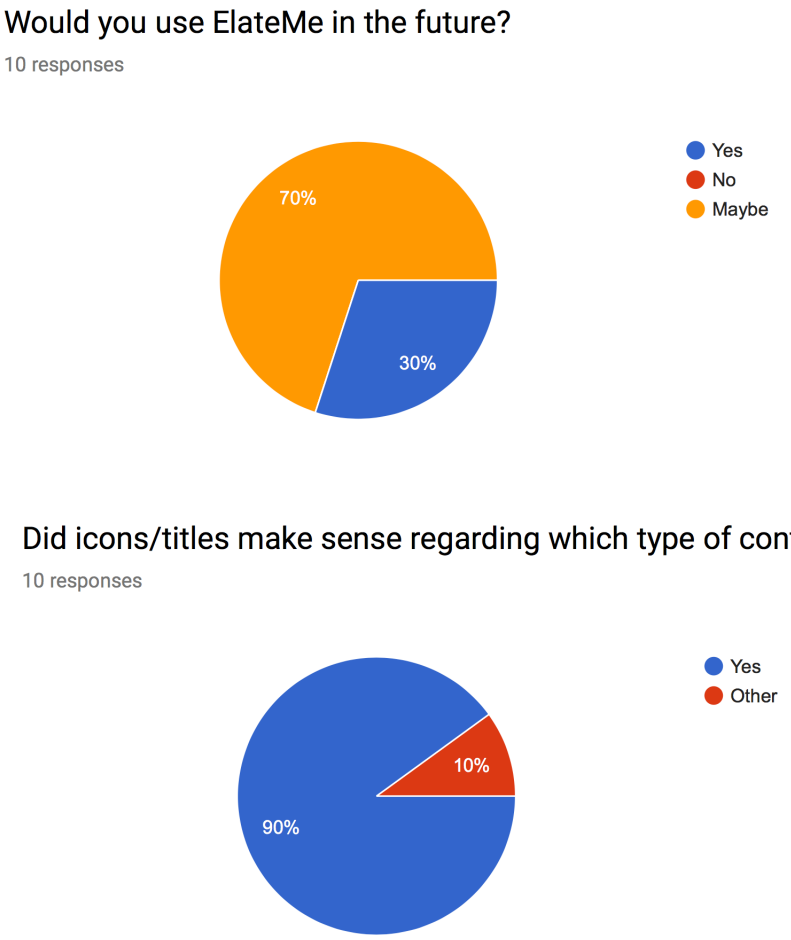


Figure C.5: Post-test questions 1,2

Appendix D

Installation guide

ElateMe application is still in development, so it is possible to launch it only using *Xcode*. Basically, several requirements should be met:

- macOS 10.12 or later
- Xcode 8.3.2 or later
- CocoaPods 1.2.0 or later

Installation from CD

1. Find `client-ios` folder
2. Copy this folder to the convenient location and open it
3. Run `pod install` from command line
4. Run `open ElateMe.xcworkspace` from command line
5. Run ElateMe target in the simulator

Appendix E

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `src` ..... the directory of source codes
  - `client-ios` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format