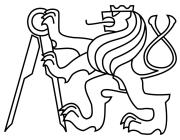
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY



ASSIGNMENT OF BACHELOR'S THESIS

Title:	ElateMe - Android client
Student:	Georgii Solovev
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 Android client for ElateMe.

1. Analyse:
- functional and nonfunctional requirements, use FURPS+.
2. Design:
- a platform specific model, significant sequence diagrams.
3. Implement:
- news feed functionality and its management,
- wish management functionality including wish product suggestion,
- crowdfunding via the FIO bank,
- friendlist management,
- common use cases (login, registration, logout, settings, notifications),
- 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 15, 2017

Czech Technical University in Prague
Faculty of Information Technology
Department of software engineering

Bachelor's thesis

ElateMe - Android client

Georgii Solovev

Supervisor: Ing. Jiří Chludil

May 15, 2017

Acknowledgements

I would like to thank the supervisor of the work Ing. Jiří Chludil and project supervisor Bc. Michal Maněna, for guidance and valuable advice. I would also like to thank ElateMe development team: Yevhen Kuzmovych, Maksym Balatsko, Yegor Terokhin and Gleb Arkhipov.

I want to express my special gratitude to my family for their support during the whole period of study, including writing this thesis.

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 May 15, 2017 ……

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

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

Abstrakt

Služba ElateMe přináší nový způsob řešení situace nákupu vhodného dárku pro své blízké. Je to kombinace sociální sítě s crowdfundingovou platformou a wishlistu. Zaměřuje se na řešení dvou problémů. První - uživatel touží po nějaké věci, na kterou nemá finance. Druhý - přítel uživatele mající takovou společenskou událost neví, jaký vhodný dar věnovat. Uživatel má možnost vytvořit v aplikaci přání, kterým chce buď sebe nebo přítele obdarovat - vyplní popis produktu a jeho cenu. Následně mohou ostatní uživatelé finančně přispívat na dar.

ElateMe je rozsáhlý systém vyvinutý pro web a mobilní platformy pro iOS a Android. Zahrnuje též i backendové části projektu. Cílem teto práce je analyzovat a implementovat klientskou aplikaci ElateMe pro mobilním operačním systém Android. Tato práce obsahuje softwarovou analýzu včetně správy požadavků využívající metodiku FURPS+. Zároveň byly navrženy platformě specifický model, sekvenční diagramy demonstrující funkčnost aplikace, a architektura aplikace. Během vývoje autor se snažil použít nejaktuálnější best practices známe ke konci roku 2016. Aplikace byla otestována uživateli v Usability lab na ČVUT.

Klíčová slova Android, Java, Klientská aplikace, Crowdfunding, sociální síť, FURPS+, MVP, VIPER, uživatelské testy.

Abstract

The ElateMe system introduces a new way to help with purchasing a suitable gift for your friends. It combines a social network with a crowdfunding platform and a wishlist. It focuses on the solving two problems. The first is that the user longs for something to which he has no finances. The second is that a friend of a user that have such a social event does not know what a good gift to give. The user has the opportunity to create a wish inside the application. He can describe the product he wants for himself or for someone else and how much it will cost. Friends, in turn, can donate on it some amount of money.

ElateMe is an extensive system developed for the web and mobile platforms for iOS and Android. It also includes the backend parts of the project. The aim of this thesis is to analyse and implement an ElateMe client for the Android mobile operating system. This text contains a system analysis including requirements in FURPS+ model. Also, there were designed the platform-specific model, the sequence diagrams that demonstrates how the application should work, and the application architecture. During development, the author tried to use the latest solutions of the Android community at the end of 2016. Finally, the application was tested with a usability test in CTU.

Keywords Android, Java, Client application, Crowdfunding, Social network, FURPS+, MVP, VIPER, usability testing.

Contents

List of Figures

Introduction

The ElateMe is a combination of social network and crowdfunding platform. It can be said that it provides crowdfunding capabilities for ordinary users. The user can create his wish, where he describe what exactly he wants, set expiration date, a required amount, and a picture. To make this process more user-friendly was created a search bar, where a user can search through e-shops and choose a product for making a template of the future wish. After it was created, the wish begins to appear in your application friends' feed, and they are able to donate to it by credit card or by bitcoin payment. Login to the application is conducted through Facebook, in this way, all user's data will be loaded on the server automatically during the first login, and therefore no data entry is required at the registration. Also in the application are available some basic features like notifications, friend list management, account management, and wish commenting.

Thesis' task

The task of this thesis is to analyse functional and nonfunctional requirements using Functionality, Usability, Reliability, Performance, Supportability, Constraints (FURPS+) classifying model, design platform-specific model and significant sequence diagrams, and perform usability tests. The following set of features should also be implemented:

News feed functionality and its management: Must be created a feed screen with different news content sent by the server. A feed management means the hiding of a single wish or all friend's wishes, the creating copy from selected wish, and sharing it through the Facebook.

Wish management functionality including wish product suggestion: This task requires to make a wish creation flow including:

- A search screen where a user can find application's suggested wishes sent by server.
- A web page screen, where a user can take a look on it on e-shop site.
- A wish creation screen, where a user sets title, description, required amount of money, etc.

Crowdfunding via the FIO bank: A donation must be created using FIO bank systems.

Friendlist management: This means a creation of friend list screen with possibility to group friends.

Common use cases (login, registration, logout, settings, notifications): Registration, login, and logout must be implemented via facebook social network. Setting on this stage of development will show current user info with a possibility of bank account number editing. Notifications are the screen with the information about last actions related to the user.

GUI (based on a 3rd party design): This requires making a Graphical User Interface (GUI) based on the design of Ing. Jan Hoffman.

Motivation

The main purpose of these subjects and bachelor's thesis was to study and try in practice the systems development life cycle in software engineering and also learn the latest technologies of programming for Android operation system.

Chapter 1

Analysis

This chapter describes an analysis of the EleteMe application. The goal is to identify the main requirements that the system will perform.

1.1 System description

EleteMe's system will follow client-server pattern with a thin client. It will have two parts: server and mobile clients on Android and iOS platforms. This application will use Facebook Software Development Kit (SDK) for registration and login. For the payment transactions will be used FIO bank SDK. In the future, the application will also have the following features:

- Local database, it is required to reduce the load on the server and the amount of mobile traffic required for the application. Also, it will provide minimal functionality in the offline mode.
- Payment in bitcoins.
- Built-in advertising.

In this analysis and design future features also taken into account.

The detailed structure of future Android client and its connection with external interfaces are presented in the component diagram on figure 1.1. The task of this thesis is an implementation of Android component. It will be connected with FIO bank, Facebook and Bitcoin service using appropriate SDK or Application Programming Interface (API). All data will be sent to the main server and received from both - main and advertisement.

Android component will be divided into three layers, where the data layer is choosing how to get or write data: from database or server. The business layer is responsible for business logic and for calling the right component. And Presentation cares only about displaying received information.

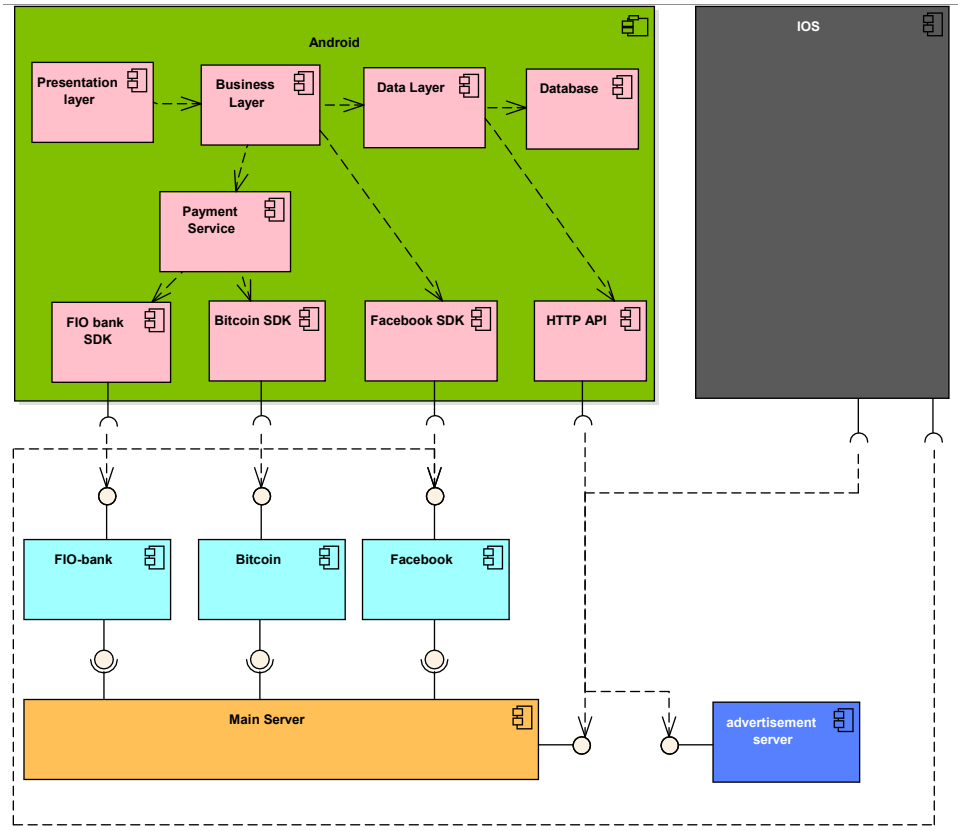


Figure 1.1: System component diagram

1.2 BI-SP1 and BI-SP2 subjects

This thesis is a continuation of the project, which began in Team Software Project 1 subject (BI-SP1) on Faculty of Information Technology (FIT) in Czech Technical University (CTU). The team consisted of two iOS, two Android, one backend programmers and author of this thesis as the architect. There were created the first iteration of platform-independent model, which you can see in Appendix C, and prototype of the Android application. In the beginning of Team Software Project 2 subject (BI-SP2), one of the Android developers switched to the backend, so author became an Android developer on this project. After BI-SP2 second Android developer decided to devote his thesis work to another project, that's why the author continued to write this application alone. The result you can see in this Bachelor's thesis.

1.3 Android platform

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Since its inception, several versions have been created. As of 3rd April 2017, their percentage ratio is shown in platform versions diagram 1.2.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.9%
4.1.x	Jelly Bean	16	3.5%
4.2.x		17	5.1%
4.3		18	1.5%
4.4	KitKat	19	20.0%
5.0	Lollipop	21	9.0%
5.1		22	23.0%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	4.5%
7.1		25	0.4%

Data collected during a 7-day period ending on April 3, 2017.
Any versions with less than 0.1% distribution are not shown.

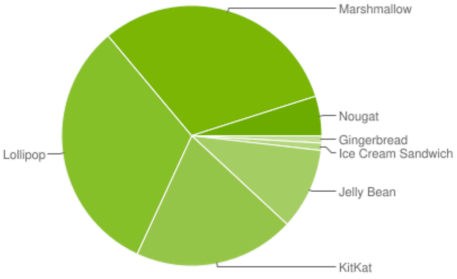


Figure 1.2: Android statistics

Each new version of Android automatically provides backward compatibility for applications running on older versions. The Support Libraries can be used to make the application, developed primarily on the latest version, work on older versions. This will allow using features that were added in new releases.

EleteMe was started to develop for Android version 4.4 KitKat (API 19) and higher. As you can see on platform versions diagram, it covers about 90% of all Android devices. At the time the application will be released, this number will only increase.

1.4 Facebook

The user must be logged in to use the application. He can sign in using a Facebook login systems. It allows the user to register without filling in

additional data. All required data server will get from the Facebook account and will send it to the client in the right form.

For implementation is required to create a developer's Facebook account, add a project to "My applications", and add a hash address of the computer, that will build the application. Then a user can sign in through built-in Facebook SDK login screen. The application will receive a Facebook token and will send it to the server, after server's positive response it is done.

1.5 FIO bank and Bitcoin

To make payment in the application will be used two systems to choose from: transfer from a credit card to an account in FIO bank or payment using cryptocurrency Bitcoin and it's payment system. The analysis of the bank is given in the thesis of my colleagues Yevhen Kuzmovych[2] and Gleb Arkhipov[4], and the analysis of the bitcoin payment did Yegor Terokhin[3].

1.6 Requirements specification

For the development of the first iteration of ElateMe is necessary to establish requirements in FURPS+ classifying. The requirements are divided into functionality, usability, reliability, performance, supportability, and constraints.

1.6.1 Functionality

All functional requirements, except login, requires a user to be authorized in the application.

Athorization management

R1 Login via Facebook:. A user will be able to sign in via Facebook if he has not already done this. The first login will be registration in the application.

R2 Sign out:. Authorized user will be able to log out.

Wishes management

R3 Create wish:. The application will allow a user to create and post wishes. The user can set up such parameters as wish's title, description, required amount of money, deadline, and group of friends that will see it.

R4 Friend's wishes:. A user will be able to see the list of his friend's wishes.

R5 Current user's wishes:. A user will be able to see the list of his own wishes.

R6 Contributed wishes:. A user will be able to see the list of wishes for which he had already donated to follow their state.

R7 Show wish detail:. A user will be able to open a wish detail with full information, comments, and the opportunity to donate.

R8 Notifications:. The system will show user information about last actions related to the user, such as new a comment under his wish, new donation, wish completion, etc.

R9 Wish recommendations:. A user will be able to use ElateMe's search to find on the Internet what he wants. Application allows creating a wish from one of search results. Wish's title, description and required amount of money will be filled automatically.

R10 Close wish:. A wish will be marked closed when required money amount is fully gathered or deadline is passed. Also, a wish can be closed earlier on demand of its owner.

Feed management

R11 Show feed:. An application will show the latest news of this application referred to the current user.

R12 Hide wish:. A user will be able to Hide wish from his feed. He won't get information about this anymore.

R13 Hide all this user's wishes:. A user will be able to Hide all wish of some user from his feed. He won't get information about this user wishes anymore.

R14 Copy wish:. A user will be able to create wish from the chosen one. It will have the same title, description, required amount of money, which can be edited.

R15 Share wish:. A user will have an opportunity to share wish through the Facebook.

Comments management

R16 Comment wish:. Users will be able to participate in the discussion under the wish.

R17 View wishes comments list:. A user will see comments from other users on the bottom of wish detail.

R18 Delete comment:. A user will able to delete his own comment.

Users management

R19 Get current user detail info:. A user will be able to get info about himself.

R20 Update current user detail info:. A user will be able to update info about himself.

R21 Create friend group:. In the application user will have an opportunity to combine friends in groups.

R22 Get friends groups:. A user will be able to see his friends groups.

R23 Delete friend group:. A user will be able to delete friends group.

R24 View friends list:. The application will show a list of Facebook friends that are also signed up in the ElateMe.

Donation magement

R25 Donation:. ElateMe supports various payment methods. In this iteration of Android client, users will be able to donate to wishes with a credit card through the FIO bank.

R26 Refund:. In case the money hasn't been collected before the expiration date or due to the fact, its owner decided to close it earlier, all donations will be returned to backers. Refundation happens automatically on the server; the user will be notified.

1.6.2 Usability

R27 Visibility of system status:. "Users should always be informed of system operations with easy to understand and highly visible status displayed on the screen within a reasonable amount of time." [6]

R28 Aesthetics and logicality of the user interface:. The user interface should be clear and intuitive with a consistent design.

R29 Protection from human factors:. The application must check user's input to avoid mistakes, especially when it comes to donations.

1.6.3 Reliability

R30 Accuracy of counting money:. Because this application works with money, it must accurately count numbers, especially decimal.

1.6.4 Performance

R31 Server connection timeout:. Server connection timeout will be set on 2 sec.

1.6.5 Supportability

R32 Expandability:. The application architecture should be convenient for adding new functionality. The previous functionality should not be changed from adding a new one.

R33 Testability:. The application should be divided into separate components according to the functional. Also, must be followed dependency inversion principle.

R34 Localizability:. The system must have a convenient way to localize it. In the case of Android, all string constants should be in a separate file.

1.6.6 Constraints

R35 Native Android application:. The result will be a native application for Android OS.

R36 Android version:. Minimal supported API will be 19. This means that application will support Android 4.4 and higher versions.

1.7 Platform-independent model

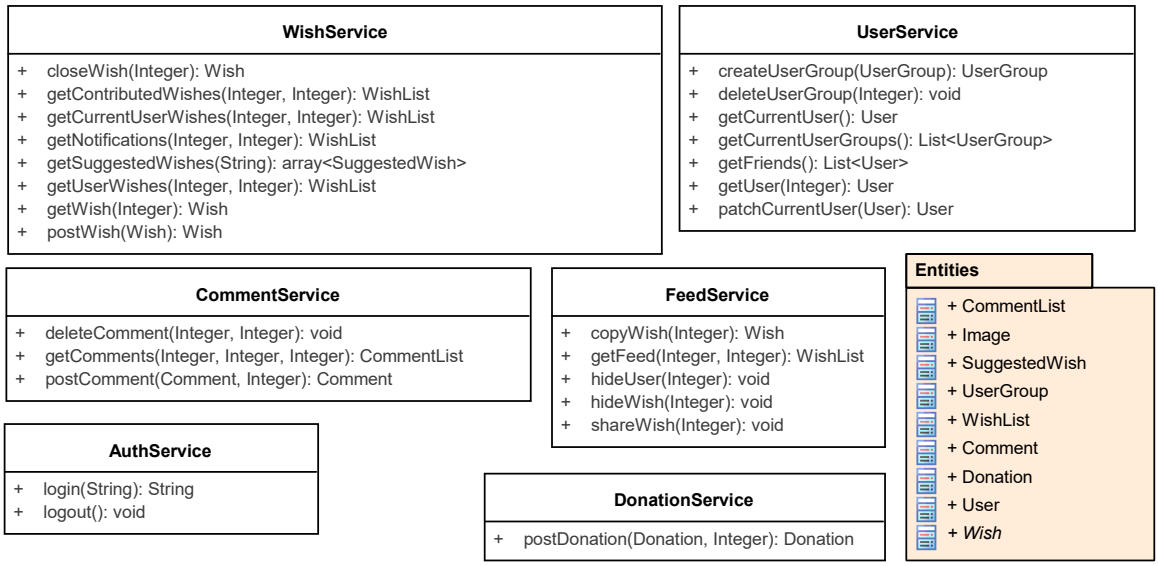


Figure 1.3: Platform independent model

The platform-independent model describes basic business logic methods and entities that will be used in the application. Methods were inspired by functional requirements. This application is essentially a thin client. It means

that model's methods represent server's Uniform Resource Locator (URL)s functionality.

You can see platform-independent model diagram on figure 1.3

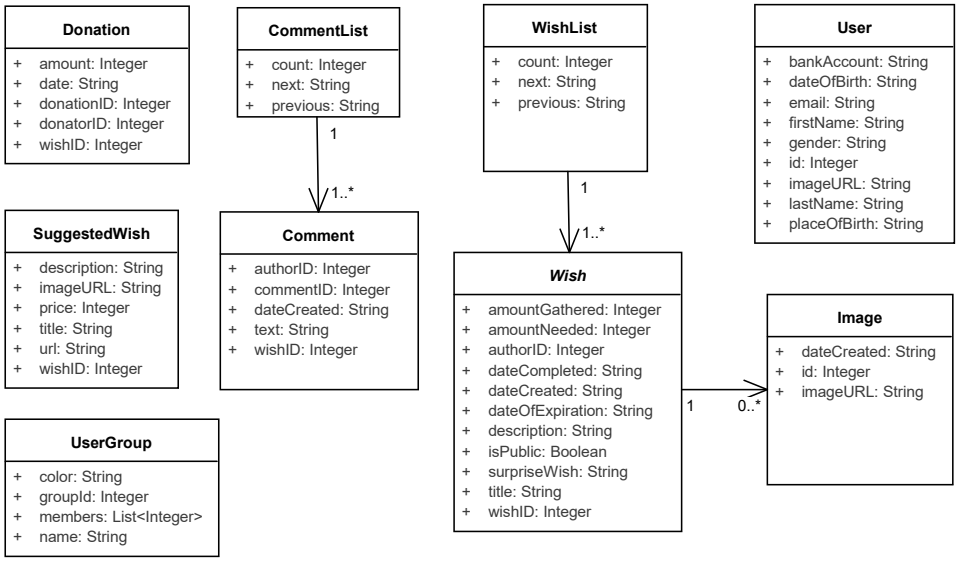


Figure 1.4: Platform independent model's entities

Describing relationships between entities, CommentList and WishList contains within themselves list of Comment and Wishes respectively, and next, and previous URL. That was caused by pagination - a process of dividing large amounts of information into parts, on the server side. Also, Wish contains Image inside itself instead of id to optimize the number of requests to the server. Other entities don't have relationships with each other, instead of this required entity can be called from the server by id in case of need.

You can see platform-independent model's entities diagram on figure 1.4. To see the full picture of the relationships on the server, take a look on the the Yevhen Kuzmovych's thesis [2].

Chapter 2

Design

This chapter contains the design for the EleteMe. It is based on the analyses described in the previous chapter. The goal is to describe how the application will comply with the requirements.

2.1 Wireframes

At the end of BI-SP1 our team created wireframes of basic screens that will be in application clients:

Main screen This is the first screen that user will see after login. It contains feed, my wishes, friends, and settings navigation tabs.

News feed 2.1a: The first opened tab that user will see after login. This screen contains all last news, like current user and his friends' wishes or information that friend got present. In future, there can be a built-in advertisement. Each feed list item will contain user's or wishes image, title and a small description.

My wishes 2.1b: This tab includes current user wishes history. Each wish list item will contain wishes' image, title, small description and progress bar that shows progress towards the assigned amount.

Friends 2.1c: It is just a list of friends. Each item contains friend's photo and his name.

Settings 2.1d: This tab provides current user personal info, friend list management, donation history and payment methods configuration.

Wish detail 2.1e: Wish detail screen contains user's photo, wish title, description, donation button and list of comments.

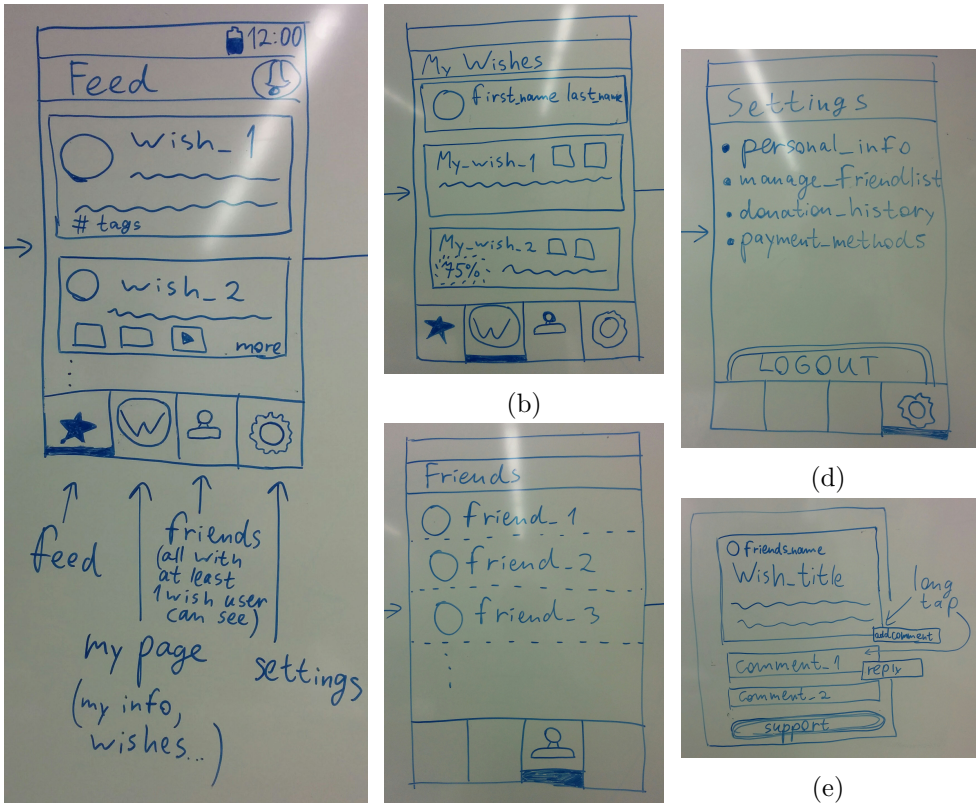


Figure 2.1: Wireframes.

Then the following screens were also discussed:

Notifications: Simple notification screen that will tell the user about last activities related to him.

Wish creation flow: This is a set of screens with which the user can create a wish in the application. It consists of a built-in ElateMe's search through which user can find some templates or create wish manually.

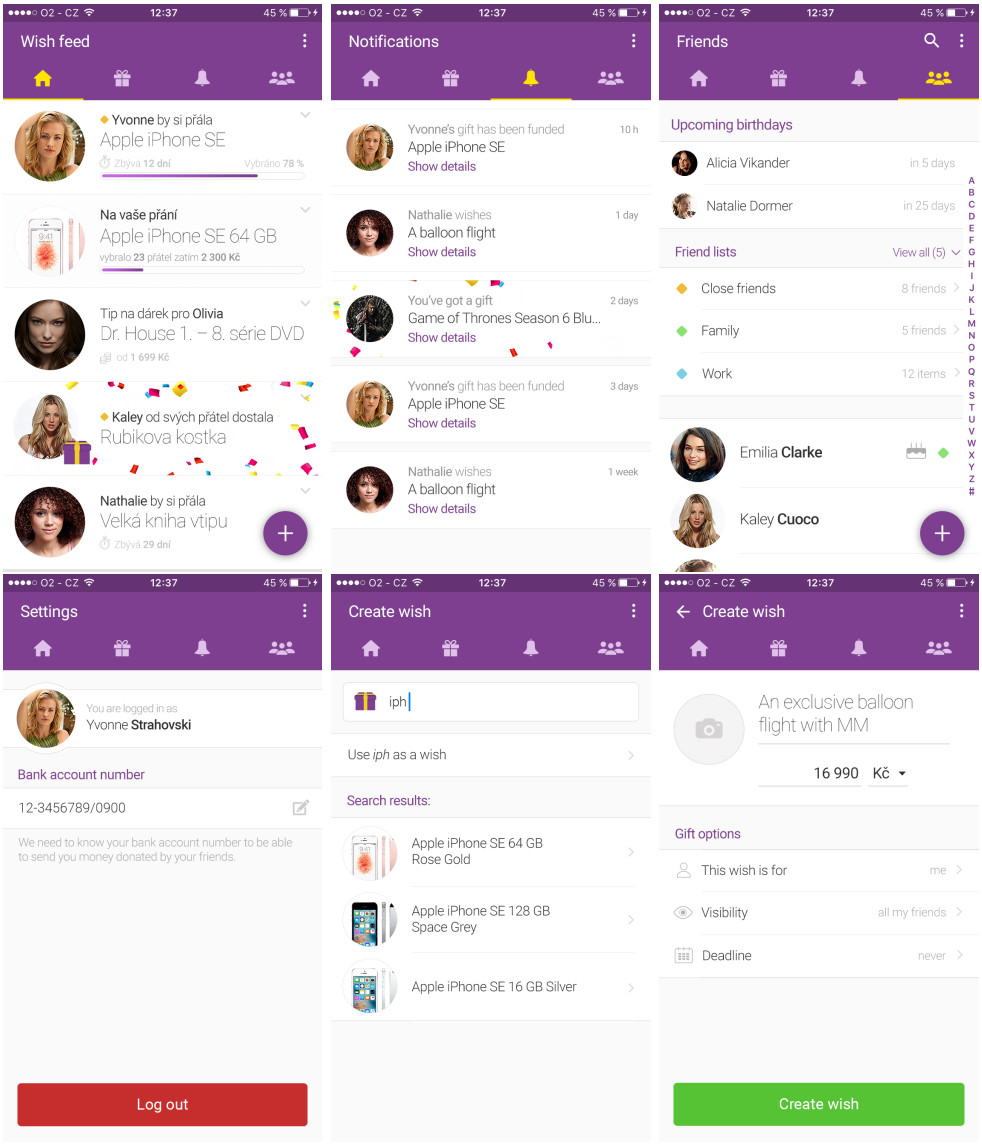


Figure 2.2: Final GUI design.

Then, based on these wireframes and discussions EleteMe's designer Ing. Jan Hoffman did the storyboard. Example you can see on figure 2.2. Full storyboard you can see it in the Appendix B.4.

2.2 Choice of application architecture

The development of the ElateMe began with the BI-SP1. In the framework of this subject, architecture was proposed for future mobile applications on IOS and Android.

The first idea was to realise Model-View-Presenter (MVP) pattern - derivation of Model-View-Controller (MVC) architectural pattern. The model contains all business logic of our application. The presenter retrieves data from the model and formats it for the view. The view displays data and routes user commands to the presenter to act upon that data.

It was originally considered that View will be represented as an eXtensible Markup Language (XML) file and Android's Activity will serve as Presenter. The model will be a singleton object which would download data from the server and provide it to activity. This idea was rejected after a while.

At the end of BI-SP1 was presented a combination of MVP with service oriented architecture. In this case, Android's Activity represents the view, presenter handles user's input and sends data from the model to view and back. The model has access to Parser, Serializer and Transport entities, through which it parses servers data or provides them to the server in a convenient format.

In the beginning of BI-SP2 subject was decided to rewrite architecture for Android OS, based on Android clean architecture pattern[7]. View and Controller saved their functionality, but for the navigation between Activities become responsible Navigator's entity. Model is separated on single use cases represented by Interactor's entity, that cares only about business logic. Data layer becomes more independent than before. In future, it will care about cache and other layers won't know anything about how data was received.

Here is a comparison of these approaches, which explains that choice.

2.2.1 First vs BI-SP1 solution

Basically, the first solution was refused because it didn't follow some basic principles of object-oriented programmings, such as:

Interface segregation principle: In the first solution, the model became one big god-object. This creates the following problem: not decoupled system makes harder to refactor, change, test, and redeploy itself. BI-SP1 solution separates model into services which are responsible for interaction with one type of object (WishService, DonationService, etc). It is a better solution, but still not the best.

Single responsibility principle: Android's activity has to take care about displaying data. In most cases, it can not be avoided, especially when it comes to non-primitive User Interface (UI) logic. So the first solution is faced with the problem that one entity has many responsibilities: displaying data, formatting data for displaying, and handling user's input. BI-SP1 solution solves this problem by dividing presenter into a separate object. Also, first solution's model is responsible for downloading data from the server, JavaScript Object Notation (JSON) parsing and

serializing objects into JSON. It was separated into three different entities by functionality, which were connected by one Service object, that contained business logic.

2.2.2 BI-SP1 vs BI-SP2 solutions

The main problem that was faced in the beginning of BI-SP2 is an absence of good examples of implementation. Furthermore Android have a list of libraries that makes development much easier and clearer. RxJava lets easily implement asynchronous, which needed for connection to the server. Retrofit 2.0 implements HyperText Transfer Protocol (HTTP) client for Android and solves parsing/serializing problems using Gson library. Dagger 2 supports dependency inversion principle which is very important for testing. Was decided to use them, therefore it entailed changes. All these libraries are already in use in Android clean architecture pattern example based on Uncle Bob's clean architecture approach. Because of this, the project began to be based on that solution. It brought the following positive points:

Interface Segregation Principle: Now business layer consists of Interactors instead of models. Interactor represents one use case. So each presenter has access only to use cases he actually needs.

Navigator: This architecture brought a Navigator entity that solves one important Android's problem: data transfer between activities. In Android system it is done by adding extra data during the creation of a new activity and extracting data in activities OnCreate method. The thing is that adding data can not be guaranteed before creation. It can cause wrong data displaying, in case that data wasn't added. Navigator is used for routing from one screen to the next, it makes sure that creation won't happen without required data.

getting rid of Parser and Serializer This implementation lets get rid of Parser and Serializer. Retrofit and Gson libraries take care about this.

2.3 Platform-specific model

The platform-specific model describes the implementation of platform-independent model. You can see it on platform-specific model partial diagram on figure 2.3. This diagram is sufficient to understand the basic principle. The complete diagram is in the Appendix B.5.

Here, as an example, the transformation of the AuthService and WishService. Within the framework of Android clean architecture, all methods have been replaced by classes inherited from Interactor. Interactor has two public methods: execute and unsubscribe. All business logic occurs in the buildObservable method, which is abstract and must be implemented in each child

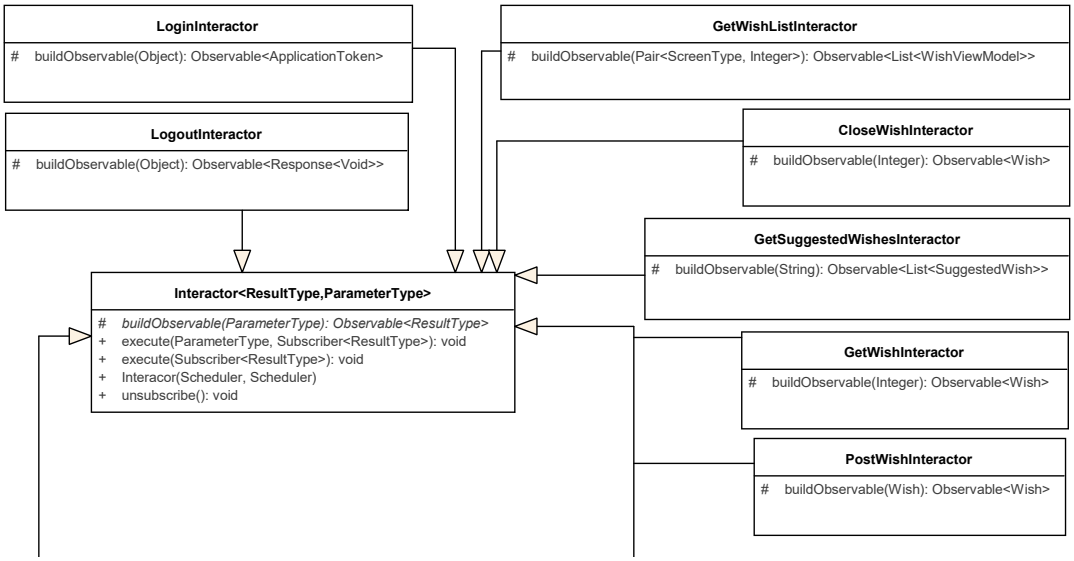


Figure 2.3: Platform-specific model partial diagramm

class. An interactor can be executed with or without a parameter. In second case buildObservable method has Object class as a parameter which will be null.

Due to the fact that feed, current user wishes, contributed wishes, and friend wishes, have the same user interface at this stage of development, was decided to replace these methods with GetWishListInteracor class. It will get screen type as a parameter depending on which will be called data provider method.

Entities were converted into Java classes with private variables and public setters and getters. In addition to this, WishViewModel and CommentViewModel have been added. They represent the data that will be displayed on view. List of view models is a result of server data mapping in GetWishListInteractor and GetCommentsInteractor. In other cases, there is no need to use view models because of the small differences between data that the server sends and data needed for the view. These view models you can see on figure 2.4.

2.4 Significant sequence diagrams

The sequence diagrams for the key moments are given below for better understanding how the various components of the application interact with each other.

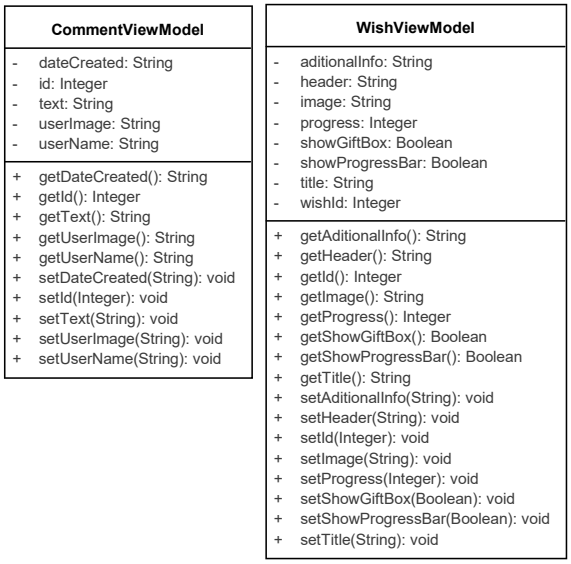


Figure 2.4: Platform specific model partial diagramm

2.4.1 Interaction between layers

Because this application realises Android clean architecture pattern, it is separated into three layers: presentation, business, and data. Also, the presentation is separated on View and Presenter. Figure 2.5 demonstrates communication between entities on the example of displaying screen with current user's wishes. After the creation, the fragment calls the presenter to get the data. The presenter orders the view to display the download progress and subscribes on the interactor - the business logic representer. Within the framework of bachelor's thesis application doesn't have any business logic on frontend clients, so interactors just asynchronously call data providers for data. They can be obtained from the server or in the future versions, from the local database in case of its validation. As soon as the data will be received, it informs subscriber - presenter, which says activity to hide download progress indicator and show data.

2.4.2 Navigator

An important part of the application is also a navigator class. All application logic relating to the switching between activities is placed in this entity. Demonstration of this class functionality is shown on the figure 2.6.

This sequence diagram demonstrates transition between Settings and ChangeBankAccountNumber activities. All other application layers except view were omitted for simplicity of understanding of the scheme.

When the user presses a change bank account button, he calls the navigator

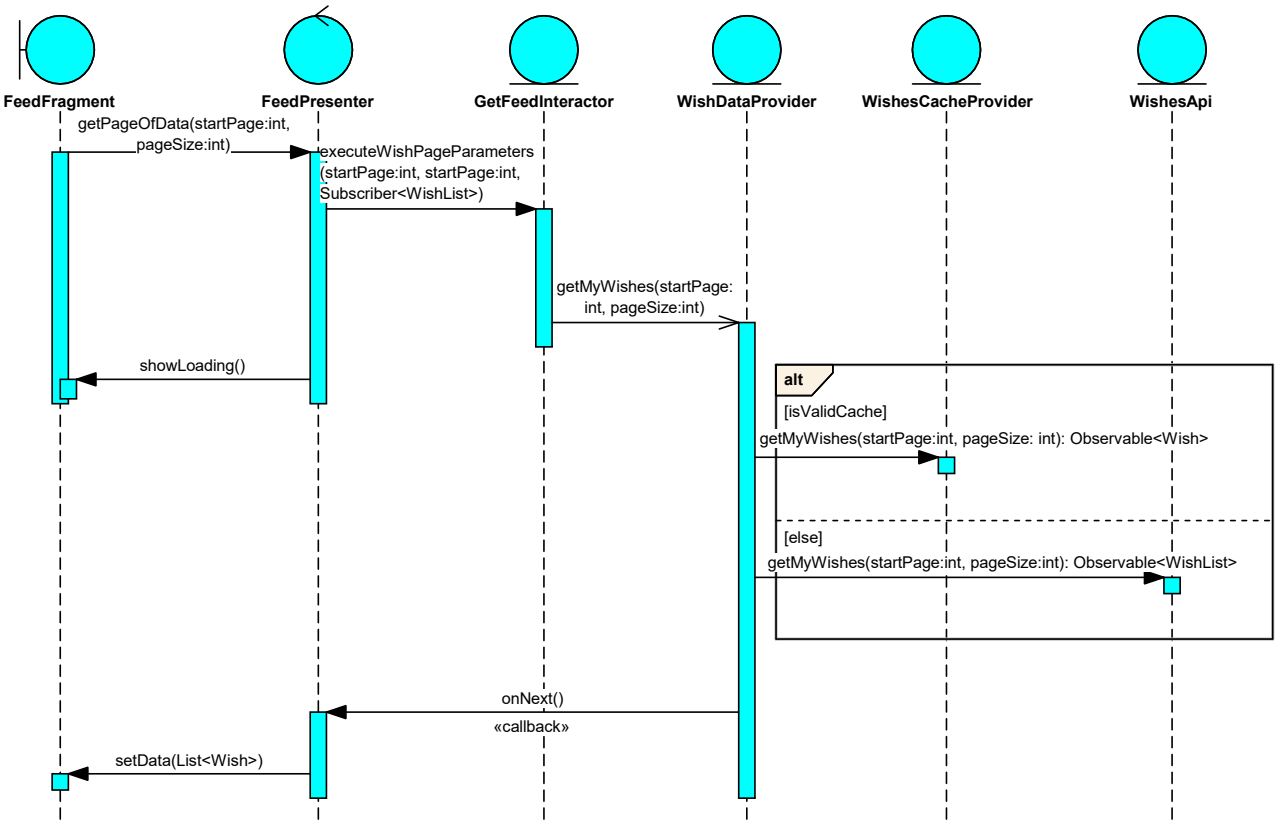


Figure 2.5: My wishes screen sequence diagram

method. Then navigator calls a static method of ChangeBankAccountNumberActivity, that provides activities own Intent[1], and launches Activity. Then after user's bank account edition and confirmation, application loads data on the server and gets updated user entity, which ChangeBankAccountNumberActivity sends back to the navigator. Navigator says SettingActivity to put user entity in intent's extras and finish ChangeBankAccountNumberActivity. This way after destroying ChangeBankAccountNumberActivity and revealing SettingActivity, it will get transmitted actual data, which the user will see.

2.4.3 Create wish

The example 2.7 demonstrates the process of creating wish by the built-in ElateMe's search. All other application layers except view were omitted for simplicity of understanding of the scheme.

[1] "An intent is an abstract description of an operation to be performed. It can be used with startActivity to launch an Activity". [8]

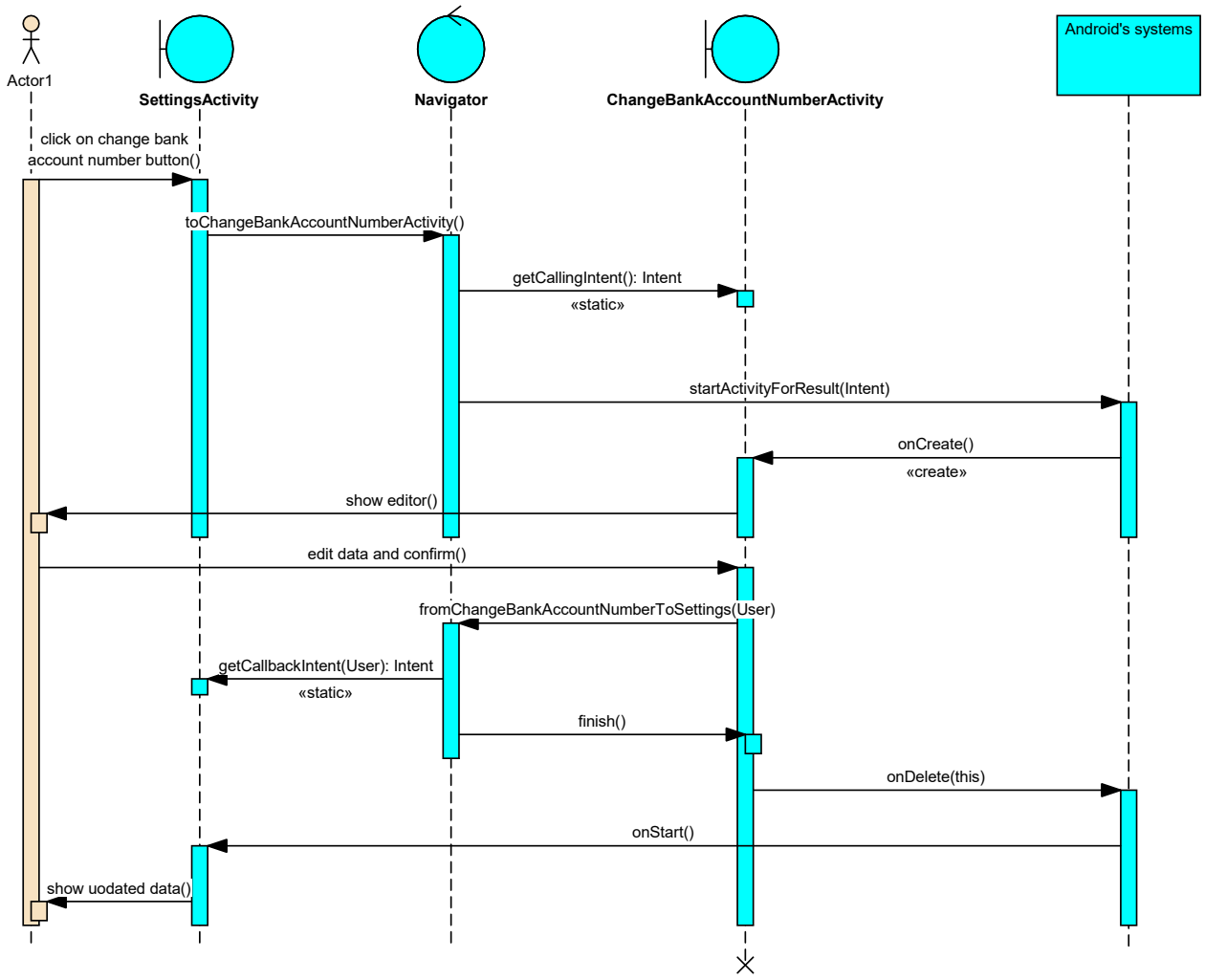


Figure 2.6: Navigator usage example

After the user enters text in the search field, the system will show him a list of suggested wishes. They originally were created on application's server from online shops products. Clicking on one of them will display the web page of the store, where a user can confirm his choice or return back to the search. In first case application automatically fill fields of wish creation like title, description, and price, which user also is able to edit. After confirmation, all data will be sent to the server, and in a case of success, the user will be redirected to the created wish detail page.

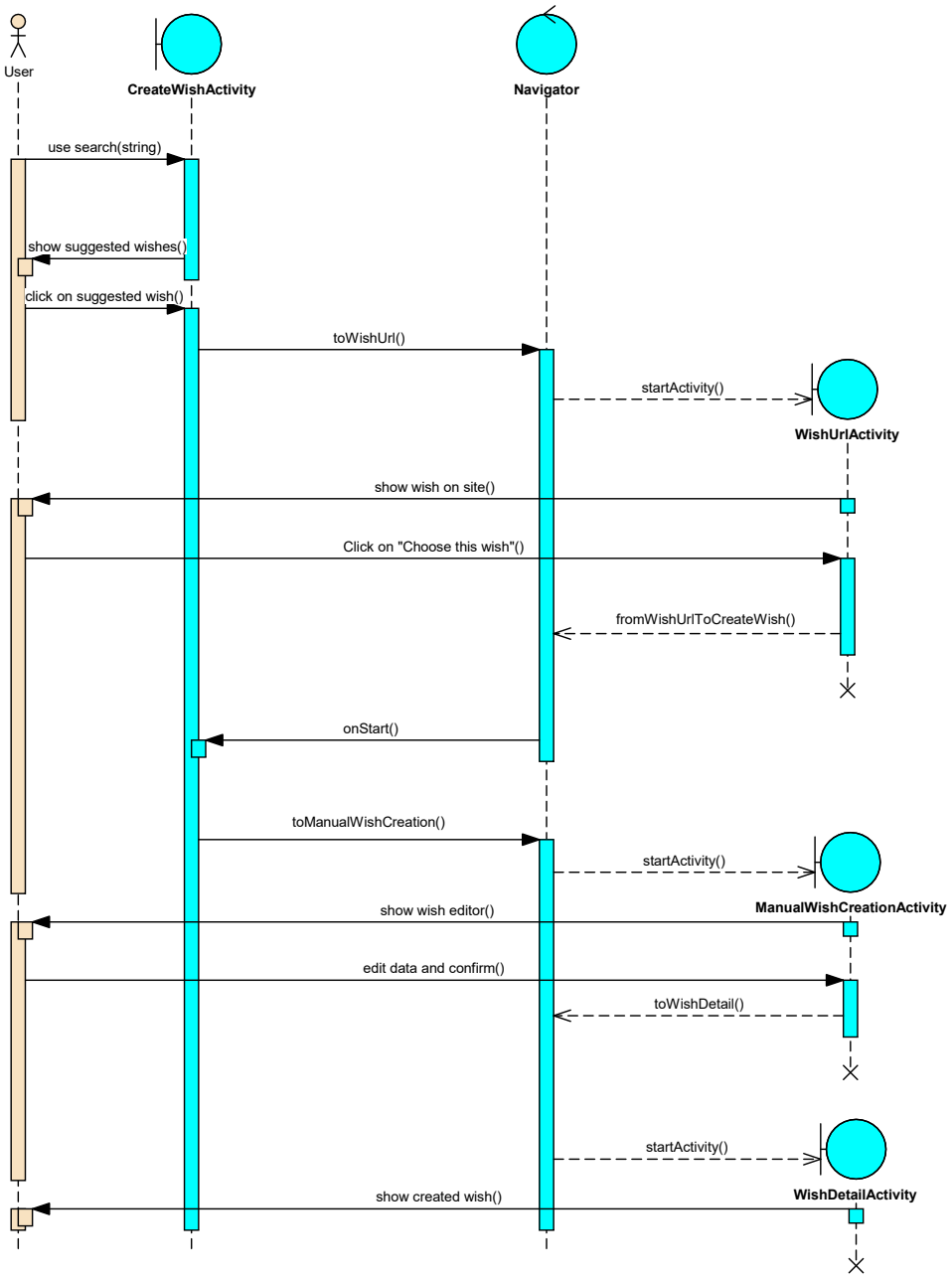


Figure 2.7: Creating a wish from the proposed

Chapter 3

Implementation

This chapter describes the important points in the EleteMe implementation: what kind of technologies were used and how they communicate with each other. Also, it contains installation and user manuals.

3.1 Used technologies

The basic stack of technologies was inspired by the Android clean architecture pattern mentioned earlier. Also, some libraries were added during the development to perform specific tasks.

3.1.1 Android support libraries

Support libraries used to provide new APIs on older versions of Android OS [9]. They were used to add features from the latest versions because minimum API Level required for this application to run is 19 (Android 4.4).

3.1.2 Facebook SDK for Android.

It gives access to the Facebook functionality inside the application. Android client uses the login, logout, and sharing functionality of the social network.

3.1.3 Dagger 2

Dagger is a fully static, compile-time dependency injection framework for Java and Android.[15] Dependency injection is a technique that helps to cover the code with tests. "A dependency is an object that can be used (a service). An injection is the passing of a dependency to a dependent object (a client) that would use it. The service is made part of the client's state. Passing the service to the client, rather than allowing a client to build or find the service,

is the fundamental requirement of the pattern."[16] In this application, this is applied by passing an instance of the service interface to a client by Dagger.

3.1.4 Moxy

"Moxy is a library that helps to use MVP pattern when you do the Android Application. Without problems of lifecycle and boilerplate code!"[12]. It facilitates the creation of an interaction between View and Presenter and also helps to restore the View state after screen rotation. The interaction is based on the dependency inversion principle.

3.1.5 RxAndroid

"RxAndroid is Android specific bindings for RxJava 2."[11] "RxJava is a Java VM implementation of Reactive Extensions: a library for composing asynchronous and event-based programs by using observable sequences. It extends the observer pattern to support sequences of data/events and adds operators that allow you to compose sequences together declaratively while abstracting away concerns about things like low-level threading, synchronization, thread-safety and concurrent data structures."[13]

3.1.6 Retrofit 2

Retrofit makes it easier to connect to the server. It turns HTTP API into a Java interface.[14]

```
public interface GitHubService {
  @GET("users/{user}/repos")
  Call<List<Repo>> listRepos(@Path("user") String user);
}
```

3.1.7 Picasso

Picasso is used for asynchronously downloading and displaying images inside the ImageView in Android. The library stores downloaded images in the Least Recently Used (LRU) cache in Random Access Memory (RAM) and in the internal memory of the device for fast access to the downloaded Pictures. [17] Example of usage:

```
Picasso.with(context).load(IMAGE_URL).into(imageView);
```

3.1.8 ButterKnife

This library simplifies field and method binding for Android views by annotating fields with @BindView and a view ID for Butter Knife.

```
@BindView(R.id.title) TextView title;
```

With the @OnClick annotation, it is also possible to assign the given event method to a click on the appropriate View. [18]

```
@OnClick(R.id.submit)
public void submit(View view) {
  // TODO submit data to server...
}
```

3.2 Component diagram

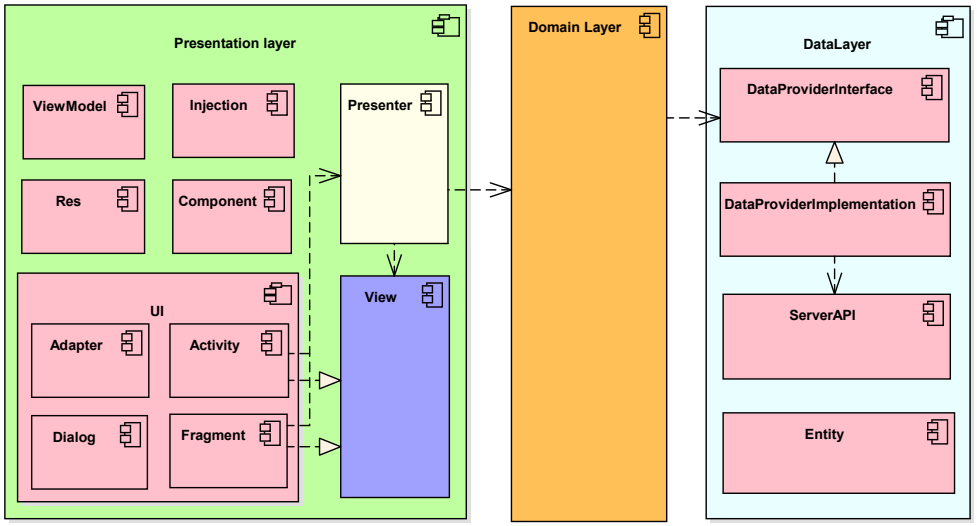


Figure 3.1: Sample of test record

On figure 3.1 is shown a component diagram describing the structural components in the Android application by the end of the development. It will help to understand the written program better. The following is a description of each component individually:
1. **Presentation layer:** This is the top level of the application. As it was said before, the presentation layer is responsible for displaying data obtained from the domain layer and also for processing a user input. It consists of the following packages:
1.1. **View:** It contains all view interfaces, which defines behaviors of Activities or Fragments as method signatures. Presenter gets a reference to the Activity or the Fragment as an interface to fulfill dependency inversion principle.
1.2. **Presenter:** As it was said earlier, presenter sends data from the domain layer to view and handles user's input.

1.3. **UI:** This package contains an implementation of all UI components of the application.
1.3.1. **Activity:** It contains all classes inherited from Activity class from Android library. In simple words, these are application's screens. They realise some of the view interfaces and includes a reference on the presenter.
1.3.2. **Fragment:** Almost the same as Activity package, but classes are inherited from Fragment class from Android library. In simple words, it is a part of the application's screen.
1.3.3. **Dialog:** This package contains classes are inherited from Dialog class from Android library. It has no connection with presenters. If it has some information that should be processed, it will be sent as a call back to the proper view.
1.3.4. **Adapter:** It contains classes inherited from RecyclerView.Adapter. "Adapters provide a binding from an app-specific data set to views that are displayed within a RecyclerView"[19]. RecyclerView is a class for showing a large data set in the limited window. In this application, it displays such data items as a list on feed, my wishes, notifications and other screens.
1.4. **Res:** This is an Android default folder, that contains all resources such as XML layouts for views, images, animations, values, and other.
1.5. **Injection:** This package contain all files related to dependency injection, more precisely to the Dagger 2 library.
1.6. **ViewModel:** It contains ViewModel files, which are used for displaying list items in RecyclerViews.
2. **Domain layer:** This package contains all business logic of the application. It consists of interactors, which represent single use case, as mentioned above. They get data by calling data providers methods and then send it to the presenters.
3. **Data layer:** This layer is responsible for providing data to the domain layer.
3.1. **DataProviderInterface:** It contains all DataProvider interfaces, which defines behaviors of the DataProvider as method signatures.
3.2. **DataProviderImplementation:** This is a realization of DataProviders interfaces. Basically it responsible for getting data from the server (in future also from the local database).
3.3. **ServerAPI:** This is an implementation of server API methods on client side.
3.4. **Entity:** This package contains the main entities that the application operates on.

3.3 Installation manual

While the application is under development, to install this application user need to download installation file with .apk extension and execute it on the

device with Android 4.4 or higher. To install, proceed as follows:

1. Enable installation on unknown devices.
   a) Go to Android device setup.
   b) Select security.
   c) Check the unknown sources item.
2. Connect the device to the PC and copy the ElateMe.apk file to the device.
3. Use the File Manager to find the .apk and run it. Then just follow installations steps.

After a successful installation user's Facebook account must be added to the EleteMe application group on `https://developers.facebook.com/` to have a possibility to log into ElateMe. It can do only someone, who have an administration right in this application on Facebook, e.g. Bc. Michal Manena. After that, you can sign in with a Facebook account.

In future after release user will be able to install the application from Play market and use it.

3.4 User manual

ElateTe has a simple and intuitive interface. But it has a short manual on the first launch which describes the main elements of the application. You can see it on figure 3.2.

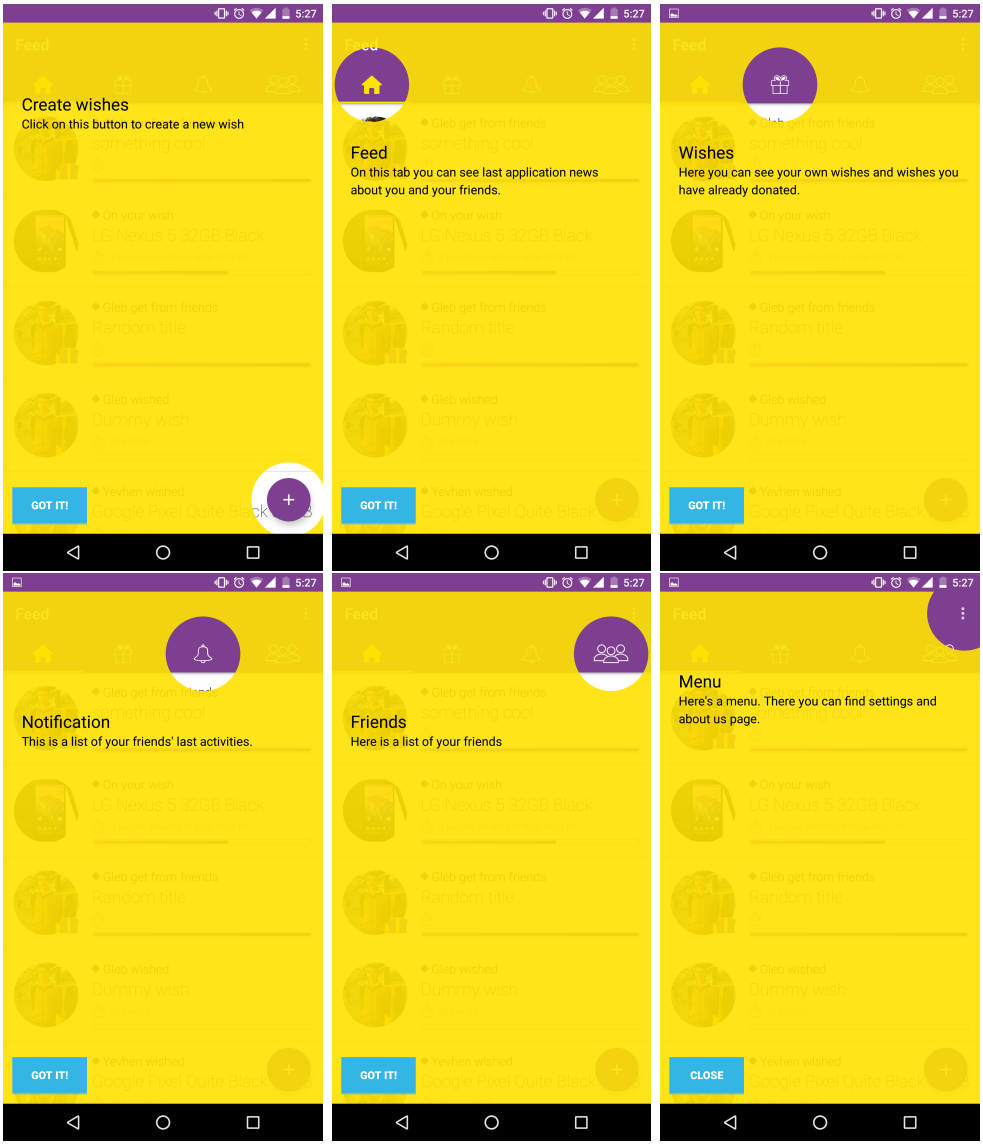


Figure 3.2: First launch manual.

Chapter 4

Usability testing

In the process of applications developing and software developing in general conducts a lot of tests. Appropriate tests can identify errors and disadvantages of the application. It is very important that the overall impression of the application is positive for the user. If the application is unstable and doesn't work as it should, then the user will quickly stop using it. The negative experience of the application can be reflected in its estimation, and it can negatively affect attraction of new users. If the application rating, for example, is low in Google Play, the user will avoid it and will choose an alternative. To solve this problem is recommended to test the application in various configurations (hardware, OS version). It is also advisable to perform usability testing.

"Usability testing is a technique used in user-centered interaction design to evaluate a product by testing it on users. This can be seen as an irreplaceable usability practice, since it gives direct input on how real users use the system."[20] In this case the so-called Hallway testing method was used. For testing were selected random people, who tried to use the application for a given scenario. This can help identify "brick walls", problems so serious that users simply cannot advance, in the early stages.

4.1 Preparation

For user testing was necessary to prepare pre-test and post-test user survey and test scenario. All of these documents are part of the Appendix B.1. Copies in the Czech language filled by the user is on CD in Appendix C.

The pre-test survey is needed to find out basic information about the tester, such as age, earnings, experience in using mobile applications and internet payments. The post-test survey is needed to get feedback from testers. Questions are about whether the interface is understandable, whether they are satisfied with the application and what they would like to change.

4.2 Test case

The main purpose of user tests was to test the clarity and friendliness of the user interface of the application. For testing was created one big script focused on the functionality. The script consisted of some number of steps, such as a creating desire, donation, writing a comment, changing user data, checking for recent notifications, and so on. Thus, the application functionality was tested almost entirely. Steps described what needs to be done, but did not describe how. In this way, the actual situation of using the application was simulated.

4.3 Selection of testers

For testing EleteMe for Android were randomly selected students of CTU FIT. Totally five students took part in the test. The results of their testing are part of the CD attachments in Appendix C and are also discussed in the results chapter.

4.4 Testing

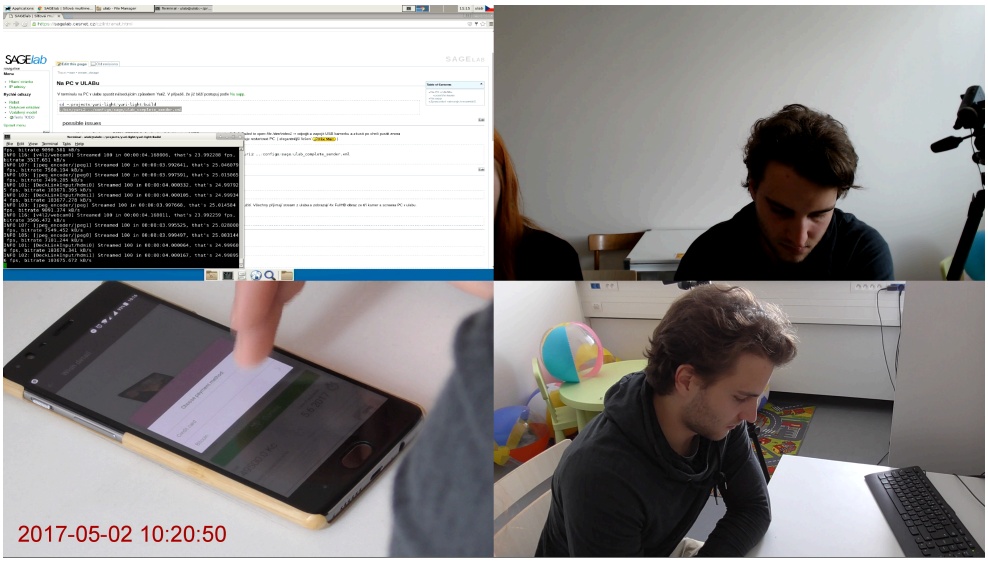


Figure 4.1: Sample of test record

The testing of users was carried out in a specialized testing laboratory at the Faculty of Information Systems. There are three cameras that record the tester and the device. All testing can be monitored in real time in the SAGE laboratory, and later a record is given for the analysis. A sample from the test's record is shown in figure 4.1.

Testing is performed by one user, to which the moderator dictates the steps to be taken. In the beginning, he fills out the pre-test survey. Then the moderator briefly presents the test application and the testing procedure. During the testing, the tester describes what seemed to him unclear or incorrect in the application interface. After all steps are completed, the user fills in the post-test survey.

4.5 Results

User testing was evaluated based on observation of the test and the post-test survey. All testers considered the interface as understandable in general and did scenario very quickly. The peculiarity is that each of them defines themselves as an experienced user and they are all Android users. Probably, additional tests will be needed for less experienced users, but considering that these testers are the main target group of the application, the result can be considered positive.

Of the shortcomings noted by testers and observers, the following can be singled out:

- The lack of a search bar on feed screen.
- Not the best place for a donation button.
- Not a clear location and choice of fonts in the description of feed and notifications items.
- The using of SwipeRefreshLayout[2] was not obvious to some testers.

These problems are subjective and will remain for further consideration by the project supervisor. And in the end, when asked whether they will use this application in the future, the two testers answered in the affirmative and the three did not decide.

[2] SwipeRefreshLayout was added Android support library version 19.1 in the 2014 year. It allows to refresh a page by swiping screen down.

Conclusion

The aim of this thesis was to analyse and implement an Android client for ElateMe. In the analysis was described a look at the interaction between the components of the system from the Android client side, established requirements using FURPS+ classifying model, and was created platform-independent model. Then was designed platform-specific model for the Android operating system, significant sequence diagrams that are describing how components of chosen architecture will interact with each other, and basic wireframes. And after implementation usability testing was successfully carried out.

Tasks fulfillment

Analyse functional and nonfunctional requirements, use FURPS+

In analysis chapter was created and described the functional and nonfunctional requirements in the FURPS+ classifying model for ElateMe Android client that reflect the desired features specified by the project supervisor. All of them described in Section 1.6.

Design a platform-specific model

The platform-specific model is based on a platform-independent model which in turn is based on requirements. The design is described in Section 2.3.

Design significant sequence diagrams

In Section 2.4 described the three most complex examples of interaction in the system, based on which the other can be understood.

Implementation

Almost all required features were implemented. Unfortunately, it was not possible to donate through the FIO bank because the bank did not provide documentation and access to FIO bank API before the deadline. This functionality has been replaced with a mock-up which simulates donation process. The result application can be seen on attached CD in Appendix C.

Perform usability tests in the usability lab

At the end of development, the application was tested by users in usability testing. The full process is described in Chapter 4.

What could be done better

Due to the fact that this project was the author's first attempt to create a serious application architecture, some errors were made during design and implementation. The following conclusions were drawn:

- Using the Android clean architecture pattern probably was an overkill in this application, as well as the use of reactive programming or even dependency injector. The thing is that application is essentially a thin client, so basically it contains only presentation logic when following technologies used more for business logic. Instead of Android clean architecture was enough to use common MVP pattern, instead of RxJava - some simpler frameworks for parallel programming, and Dagger 2 is redundant because this application won't need full Unit test coverage. The main problem which may be caused by this choice of the technology stack is a complicated entry into the project by new developers.
- The Moxy is a useless framework in this project. Because in future all data will be cached in the local database, the activity recreation with current data saving won't make a problem.
- The Dagger 2 used incorrectly by providing dependencies from one big class. In the future, the provision of dependencies should be divided into single responsibility components.

Bibliography

[1] Global market share held by smartphone operating systems from 2009 to 2016 [online]. [viewed 10 March 2017]. https://www.statista.com/statistics/263453/global-market-share-held-by-smartphone-operating-systems/.

[2] Kuzmovych, Y. ElateMe - Backend. Bachelor's thesis, Czech Technical University in Prague, Faculty of Information Technology.

[3] Terokhin Y. ElateMe - iOS client I. Bachelor's thesis, Czech Technical University in Prague, Faculty of Information Technology.

[4] Arkhipov G. ElateMe - iOS client II. Bachelor's thesis, Czech Technical University in Prague, Faculty of Information Technology.

[5] Peter Eeles. Capturing Architectural Requirements, 15 November 2005 [online]. [viewed 10 March 2017]. http://www.ibm.com/developerworks/rational/library/4706.html.

[6] Euphemia Wong. User Interface Design Guidelines: 10 Rules of Thumb. February 2017 [online]. [viewed 24 March 2017]. https://www.interaction-design.org/literature/article/user-interface-design-guidelines-10-rules-of-thumb.

[7] Android-CleanArchitecture. [online]. [viewed 7 May 2017]. https://github.com/android10/Android-CleanArchitecture.

[8] Android documentation. Intent. [online]. [viewed 3 April 2017]. https://developer.android.com/reference/android/content/Intent.html

[9] Android documentation. Support Library Features Guide. [online]. [viewed 1 May 2017]. https://developer.android.com/topic/libraries/support-library/features.html.

[10] Facebook SDK for Android. [online]. [viewed 1 May 2017]. https://developers.facebook.com/docs/android.

[11] RxAndroid: Reactive Extensions for Android. [online]. [viewed 1 May 2017]. https://github.com/ReactiveX/RxAndroid.

[12] Moxy: Moxy is MVP library for Android. [online]. [viewed 1 May 2017]. https://github.com/Arello-Mobile/Moxy.

[13] RxJava: Reactive Extensions for the JVM. [online]. [viewed 1 May 2017]. https://github.com/ReactiveX/RxJava.

[14] Retrofit. [online]. [viewed 1 May 2017]. http://square.github.io/retrofit/.

[15] Dagger. [online]. [viewed 1 May 2017]. https://google.github.io/dagger/users-guide.html.

[16] I.T., Titanium. "James Shore: Dependency Injection Demystified". [online]. [viewed 18 July 2015]. www.jamesshore.com.

[17] Picasso. [online]. [viewed 1 May 2017]. http://square.github.io/picasso.

[18] Butter Knife. [online]. [viewed 1 May 2017]. http://jakewharton.github.io/butterknife/.

[19] Adapter. [online]. [viewed 1 May 2017]. https://developer.android.com/reference/android/support/v7/widget/RecyclerView.Adapter.html.

[20] J. Nielsen, 1994. Usability Engineering, Academic Press Inc, p 165.

Appendix A

Acronyms

BI-SP1 Team Software Project 1 subject

BI-SP2 Team Software Project 2 subject

SDK Software Development Kit

FURPS+ Functionality, Usability, Reliability, Performance, Supportability, Constraints

MVP Model-View-Presenter

MVC Model-View-Controller

FIT Faculty of Information Technology

CTU Czech Technical University

GUI Graphical User Interface

API Application Programming Interface

SDK Software Development Kit

URL Uniform Resource Locator

UI User Interface

JSON JavaScript Object Notation

HTTP HyperText Transfer Protocol

LRU Least Recently Used

RAM Random Access Memory

XML eXtensible Markup Language

Appendix B

Addition materials

B.1 Pre-test survey

1. Your age
   - 12-16
   - 17-18
   - 19-25
   - 26-34
   - 35-50
   - 50+
2. Your current income
   - up to 10 000 kc monthly
   - up to 25 000 kc monthly
   - up to 40 000 kc monthly
   - greater than 40 000 kc 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 lefts my hand
6. Your experience with mobile applications (Put checkmark, if you used it)
   - FACEBOOK
   - UBER
   - AIRBNB
   - BOOKING.COM
   - INSTAGRAM
   - Mobile Banking
7. Have you ever bought something with a mobile application?
   - Yes
   - No
8. Do you have debit/credit card?
   - Yes
   - No
9. How often do you use it?
   - Never
   - Annually
   - Quarterly
   - Monthly
   - Weekly
   - On a daily basis
10. Have you heard about bicoin?

- Yes
- No
- What is bitcoin?

11. Do you use Android or iOS?
    - iOS
    - Android
    - Other / ostatni

B.2 Scenario

- Login via Facebook.
- Look through the tutorial.
- Check notifications.
- Realise that your wish “Pencil” was completed and “Pen” wasn’t.
- Click wishes creation button.
- Enter “iPhone” in the search field to find suggested wishes.
- Select one of the offered items.
- Look through the web page and confirm that it is actually what you want.
- If some field does not suit you, change it.
- Set date of expiration to 15.05.2017.
- Confirm a wish creation.
- Go to “My wishes” screen.
- Find newly created wish and share it on facebook, use long tap to open a pop-up menu with this button.
- Go to the settings screen and set your bank account number to 1234567890/1234.
- Find a wish called “Macbook pro retina” created by your friend “Jan Novak”. There are two options
  - Find it on Feed screen.
  - Go to Friends screen and click on your friend “Jan Novak”. Choose his wish “Macbook pro retina”.

- Open wish detail.
- Donate any amount to this wish.
- Check if donation succeeded or not.
- Leave a comment below.

B.3 Post-test survey

1. Did image/title make sense regarding which type of content was in the tab?
   - Yes
   - No (write what didn't make sense?)
2. Would you use ElateMe in the future?
   - Yes
   - Maybe
   - No
3. What features would you like to see in our application?

B.4 Storyboards

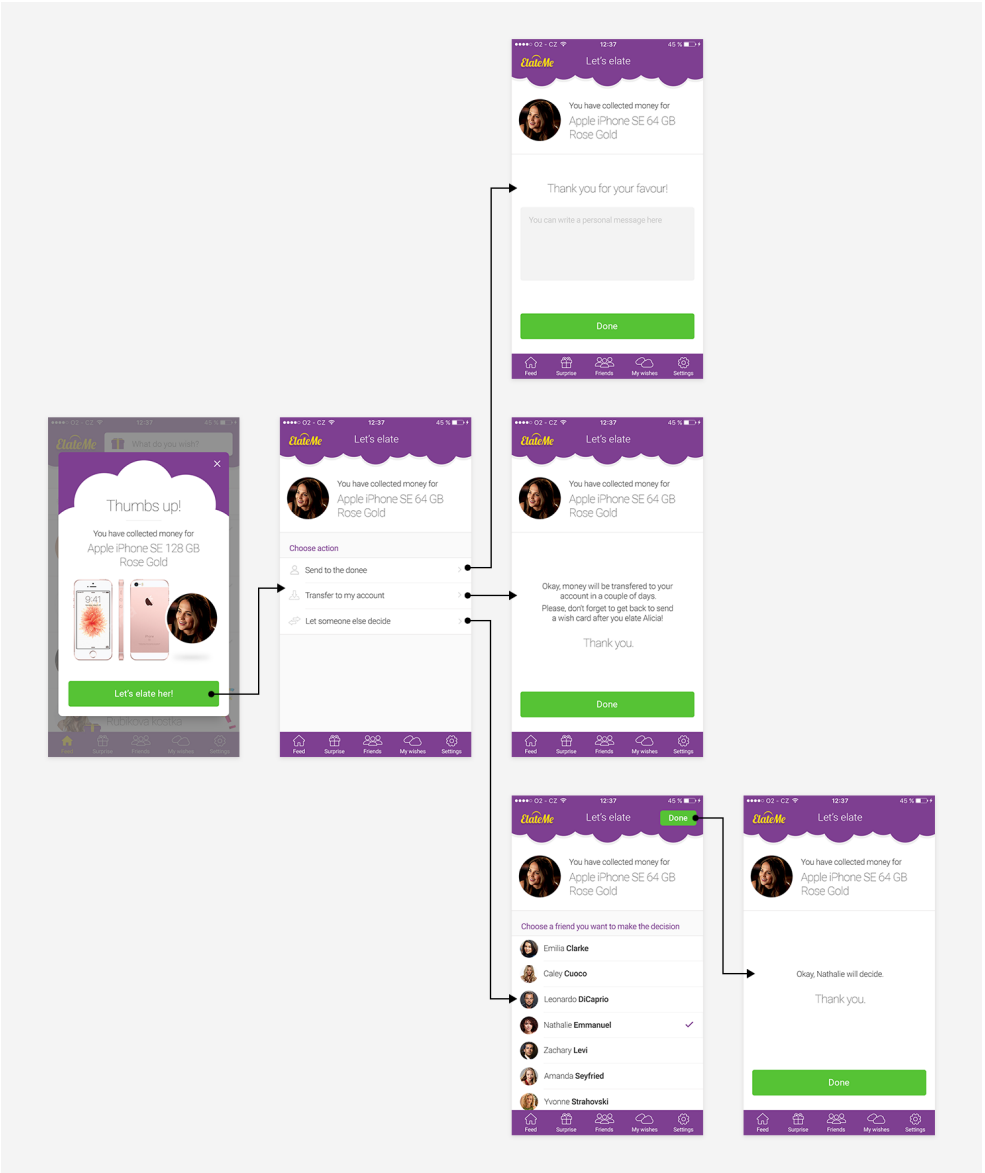


Figure B.1: ElateMe's money collection storyboard

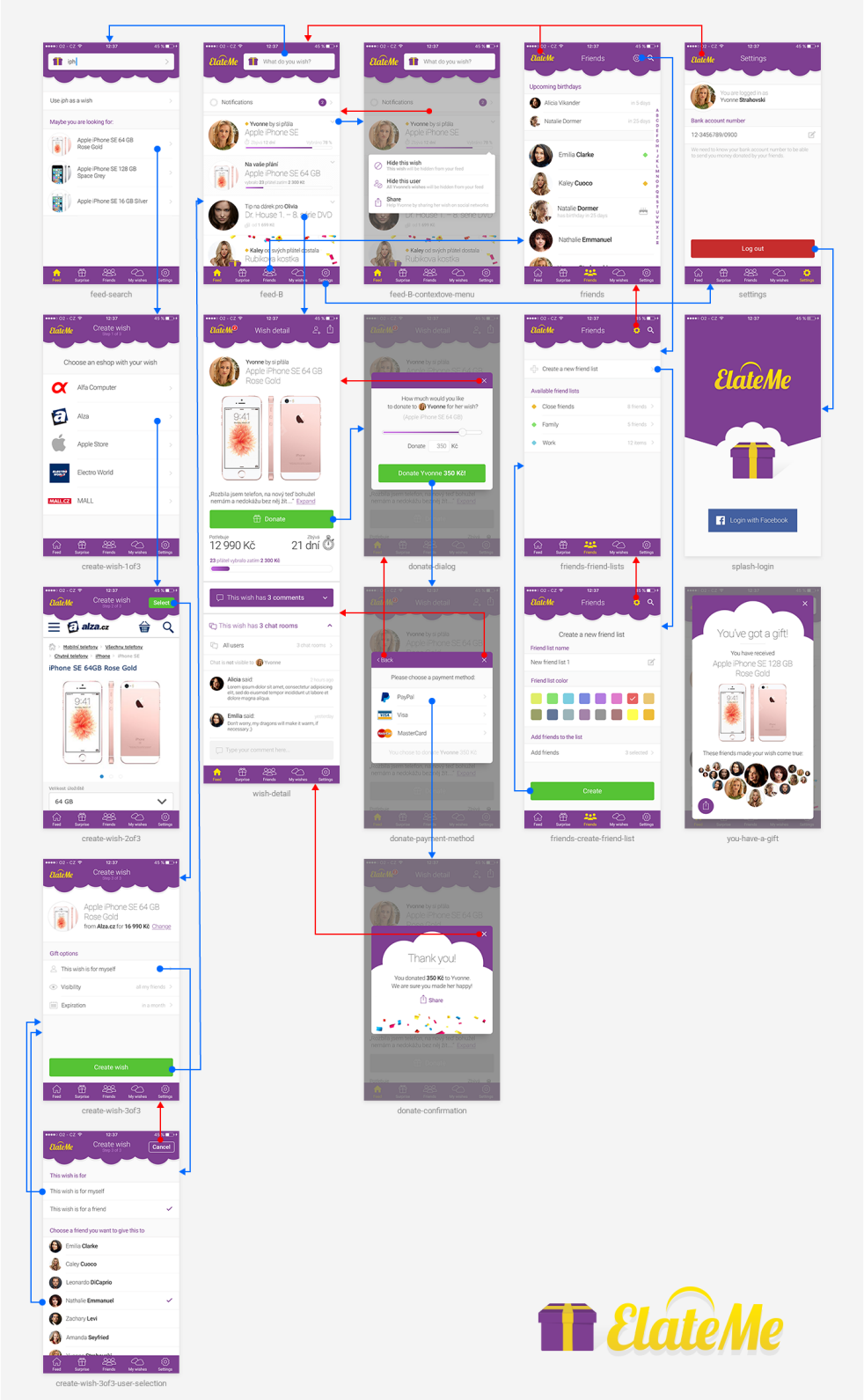


Figure B.2: ElateMe's full storyboard

B.5 Platform-specific model

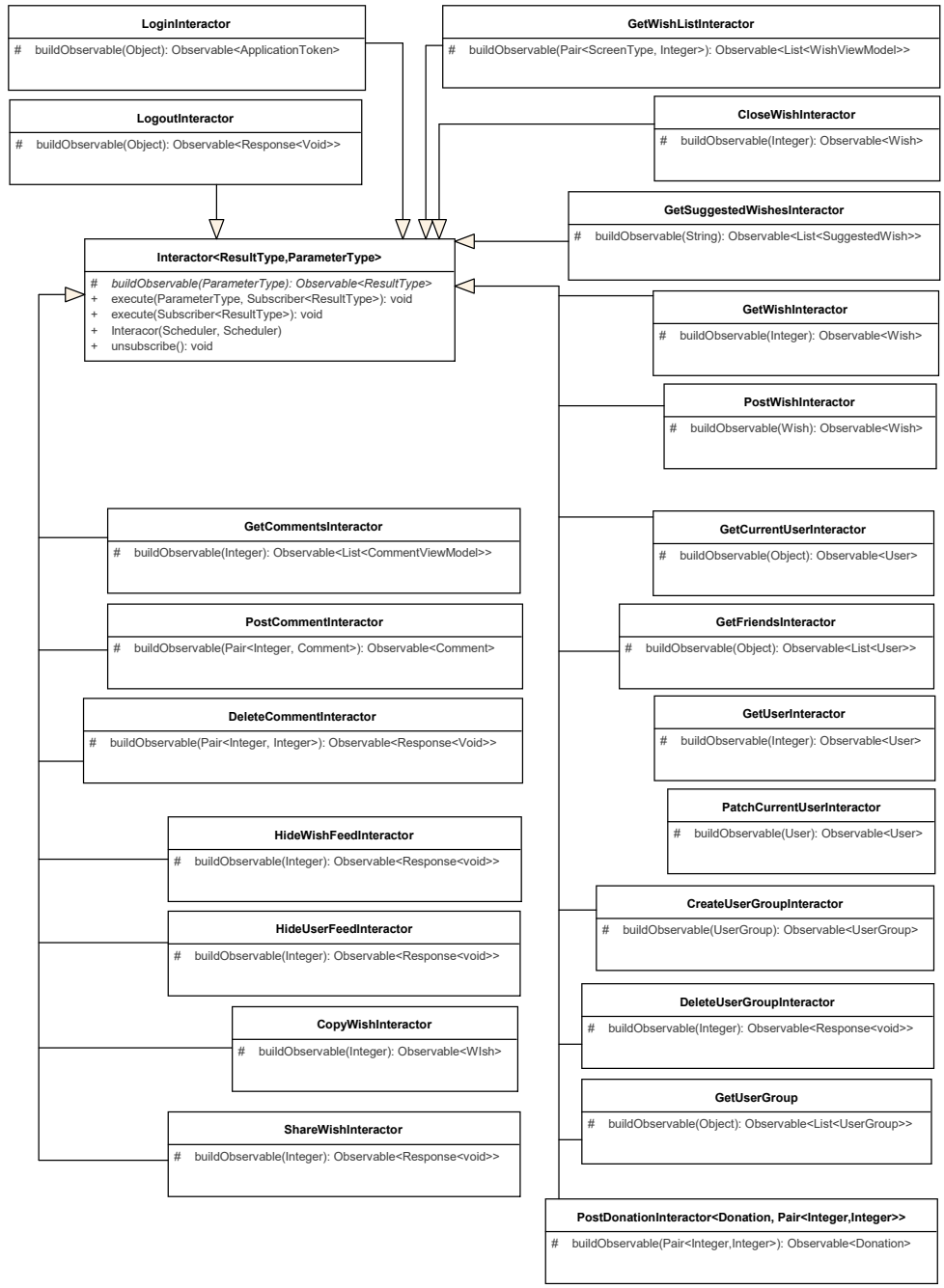


Figure B.3: Complete platform specific model.

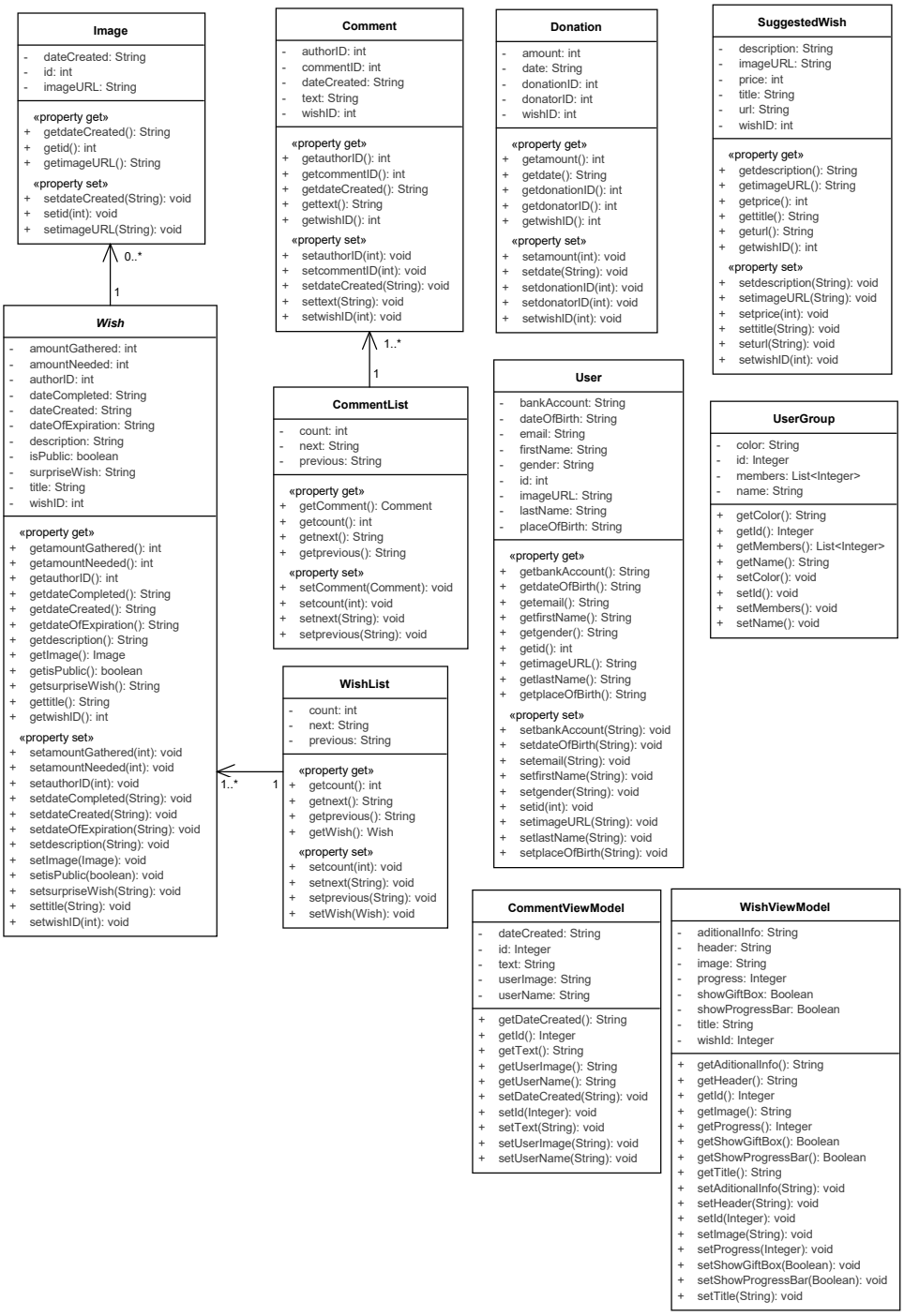


Figure B.4: Complete platform specific model entities.

Appendix C

Contents of enclosed CD

- `root` ..... The root directory
  - `ElateMe.apk` ..... Android installation file
  - `Solovev_Georgii_Bachelor's_thesis.pdf` the thesis in PDF format
  - `src` ..... the directory of source codes
    - `thesis` ..... the directory of LaTeX source codes of the thesis
    - `client-android` ..... the application source code
    - `diagramms.EAP` ..... all diagrams made in Enterprise Architect
  - `attachments`
    - `BI-SP1 architecture documentation.pdf` ..... Architecture documentation at the end of BI-SP1 subject
    - `test surveys` ..... Directory with test surveys user testing