ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Dausheyev** Jméno: **George-Mukhammed** Osobní číslo: **381720**

Fakulta/ústav: **Fakulta informačních technologií**

Zadávající katedra/ústav: **Katedra softwarového inženýrství**

Studijní program: **Informatika**

Studijní obor: **Webové a softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Aplikace pro architekty pod OS Android

Název diplomové práce anglicky:

Android application for architects

Pokyny pro vypracování:

Design and implement a prototype of a mobile application for the Android platform that will have the potential to improve a communication between an architect and a client using the augmented reality technology.
1. Analyze requirements of potential users and provide- use cases,- FURPS requirements.2. Design and describe - an application architecture,- user-friendly UI,- class diagram,- a method for a simple sharing and exporting of projects from a personal computer to the mobile application using a server application with the REST API.3. Implement- the designed application with elements of the augmented reality,- the export of projects from a computer to the mobile application.4. Test the final application from the requirements and UI points of view.

Seznam doporučené literatury:

Will be provided by the supervisor.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jiří Chludil, katedra softwarového inženýrství FIT

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **28.02.2017** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: _____

_____	_____	_____
Podpis vedoucí(ho) práce	Podpis vedoucí(ho) ústavu/katedry	Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

_____ . _____	_____
Datum převzetí zadání	Podpis studenta

Czech Technical University in Prague

Faculty of Information Technology

Department of Software Engineering

Master’s thesis

Android application for architects

Bc. George-Mukhammed Dausheyev

Supervisor: Ing. Jiří Chludil

9th May 2017

Acknowledgements

I would like to thank Ing. Jiří Chludil for all the suggestions he made during writing the thesis. Further I would like to thank my parents, sister and friends for support throughout my studies.

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 9th May 2017 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Dausheyev, George-Mukhammed. *Android application for architects*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2017.

Abstrakt

Hlavním cílem této diplomové práce je implementace prototypu aplikace na platformě operačního systému Android, která je schopná vizualizovat architektonické projekty pomocí použití technologie rozšířené reality. Při její vypracování byly použité různé přístupy.

Klíčová slova Android, rozšířená realita, LibGDX, Vuforia, ARToolkit, REST, uživatelské rozhraní

Abstract

The main goal of this thesis is to implement a prototype of Android application for architecture project visualisation using augmented reality technology. Different approaches and tools were tested.

Keywords Android, augmented reality, LibGDX, Vuforia, ARToolkit, REST, UI

Contents

List of Figures

List of Tables

Introduction

Progress in technologies allows advancement and automation of different fields of human activity and industry. Smartphones capabilities are becoming more advanced every year and its usage is now an everyday routine. Nowadays trends, such as Big Data, Virtual Reality, and Augmented Reality in combination with the above mentioned habituation inspires people to innovate and create something new and fascinating.
Even though Augmented Reality was born long ago, it still has not gained mass popularity. People are cautious towards trying technology, and this creates boundaries and restrictions. They still dream about widespread usage of AR. Application of AR in architecture is just one interesting use case among plenty of different potential usages of AR in life. Architects will be able to develop various projects with respect to the surroundings. They will know how a building or memorial will fit in a real world environment and match already existing constructions. The task was originally given as an assignment from Kateřina Nováková on the "Cooperation with Industry" portal. As a result of discussions with students of the Faculty of Architecture I understood that the communication with the customer is based on the following steps:

1. Architects models a project in a 3D graphics software products.
2. Model is rendered from different angles and the renders are sent to the customer.
3. The final presentation takes place during the personal meeting. Architects prepare mock-ups and include some final renderings to the presentation.

The above mentioned approach demonstrates a customer's needless dependence on architects. The angle of render, scale and other details are

chosen only by one side. One of the purposes of this thesis is to provide a customer with an ability to view a model the way he/she wants it to be. It will also allow architects to impressively present a project in the final stage of the designing.

Chapter 1

Task Analysis

The aim of this thesis is to design and implement a prototype of a mobile application on the Android platform that will have the potential to improve the communication between an architect and a client using the augmented reality technology. The task is divided into several parts and I will comment each part of the task:
Analyze requirements of potential users and provide use cases and FURPS requirements.
The project will be discussed with Kateřina Nováková as a customer to define requirements and describe use cases.

Design and describe: an application architecture, user-friendly UI, class diagram, a method for a simple sharing and exporting of projects from a personal computer to the mobile application using a server application with the REST API
I will design all necessary models for project implementation guided by the analysis results. I will design REST server API as it was dictated by the task. Wireframes, class diagrams, domain model and database conceptual models will be created during this phase.

Implement the designed application (prototype - defined in the beginning) with elements of the augmented reality and the export of projects from a computer to the mobile application.
I will choose appropriate technologies and create application.

Test the final application from the requirements and UI points of view
The application will be tested with and without users. The result of tests will be used for the project improvement.

Chapter 2

Analysis

The next chapter describes the project analysis which has to be carried out before the start of the implementation process. Analysis is a very important part of the software development. In the end of the chapter a precise project specification, its limitations and requirements will be stated. The task, functional and non-functional requirements were worked up in collaboration with Ing. arch. Kateřina Nováková from the Faculty of Architecture CTU and Jan Vlnas, a student from the Faculty of Mathematics and Physics, Charles University. They suggested to name the application **Ambiant**.
To make the text concise and clear I composed a list of the main terms and definitions.

2.1 The list of terms and definitions

Ambiant is a name of mobile application that will be created in this thesis.

Architect is a person who specializes in planning, designing, and reviewing the construction of buildings. Usually he/she designs the architecture in civil engineering using professional 3D computer graphics programs.

Project is an architectural solution represented as a 3D model and exported in one of the most widespread file formats that includes texture, shaders, meshes and materials.

Client is a person or an organization that orders a project from an architect.

Marker is a pre-determined 2D image, ideally contrasting and printed for the purpose of an effective computer vision based image recognition.

The initial task that I was given is to create a mobile application for Android platform using augmented reality technology. The primary target group of this application are architects or students of the Faculty of Architecture. Kateřina Nováková described the students' expected workflow with the application:

1. Student(s) create a physical model (i.e. from paper and glue) of building surroundings.
2. Put a marker into the appropriate place of the model.
3. Visualize the project in all phases of the development and optimize its appearance with respect to the surrounding.

2.2 Functional requirements

The application should have the following functionality.

F1: Visualization of a 3D model
The application will be able to display the project on the Android device. It should be possible to rotate and scale a model.
Priority: high
Difficulty: high

F2: Visualization of a 3D model on a marker as a part of augmented reality
The application will be able to display the project using video stream from a camera with AR effect. It should be possible to rotate and zoom a model.
Priority: high
Difficulty: high

F3: Opening of a project model from a device memory
The application will be able to open a project from device file system.
Priority: high
Difficulty: medium

F4: Switching layers of a model
The application will be able to change the visibility of layers of a 3D model during the observation process.
Priority: medium

Difficulty: high

F5: Adjusting a model to match the marker
Marker is an real world image. Model should be able to change scale and rotation towards the marker to fit it. Modified parameters will be possible to transfer from one device to another.
Priority: medium
Difficulty: high

F6: Taking a photo and video of a model
Capturing a photo or video during a model observation process in augmented reality. Saving the result on the device memory using device capabilities.
Priority: low
Difficulty: high

F7: Using a custom image as a marker
A marker is a image that fulfills requirements dictated by the image recognition system. Application should be able to define new image as a marker.
Priority: medium
Difficulty: depends on the library

F8: Import a model from a popular cloud file hosting
The application should be able to load a project file from the Dropbox, Google Drive or OneDrive.
Priority: low (optional)
Difficulty: medium

F9: Changing a model in run-time
The application should be able to switch between several models without stopping and exiting the application.
Priority: high
Difficulty: high

F10: Sending a taken photo of a project in AR to the third-party applications
Priority: optional
Difficulty: high

F11: Exporting a project directly to the application

Sharing a project from a personal computer to the mobile Android application in a comfortable way.
Priority: optional
Difficulty: high

F12: Opening the COLLADA (.dae) or some other widespread file format with a layer support.
Collada is a common file exporting format, so the application should be able to work with it.
Priority: medium
Difficulty: high

F13: Change settings of the application.
If application can provide some customizable component, user should be able set parameters using settings menu.
Priority: medium
Difficulty: low

F14: Edit a list of projects in the application.
Application should be able to delete a project from a list.
Priority: medium
Difficulty: low

2.3 Non-Functional requirements

In this section I will describe requirements that tells more about quality, platform, performance, how the system should work rather than directly dictate behavior of the application.

N1: User friendly user interface
The application must have effective, clear and intuitive user interface.
Priority: high
Difficulty: medium

N2: Native Android application
The resulting application must be natively supported on the majority of Android smartphones.
Priority: high
Difficulty: low

N3: Availability
The application should be able to display the model as long as smartphone correctly works. Internet is required only for exporting a project from a computer to the Android device.
Priority: high
Difficulty: low

N4: Performance
The application should provide responsive and smooth UI interaction, so 3D visualization should make an impressions of augmented reality and there will be no lags.
Priority: high
Difficulty: medium

N5: Languages
Application should support Czech, English and Russian languages.
Priority: medium
Difficulty: low

N6: Communication with the server
If the implementation of functional requirements demands a server implementation, the app and a server should communicate through HTTP using REST interface.
Priority: medium
Difficulty: medium

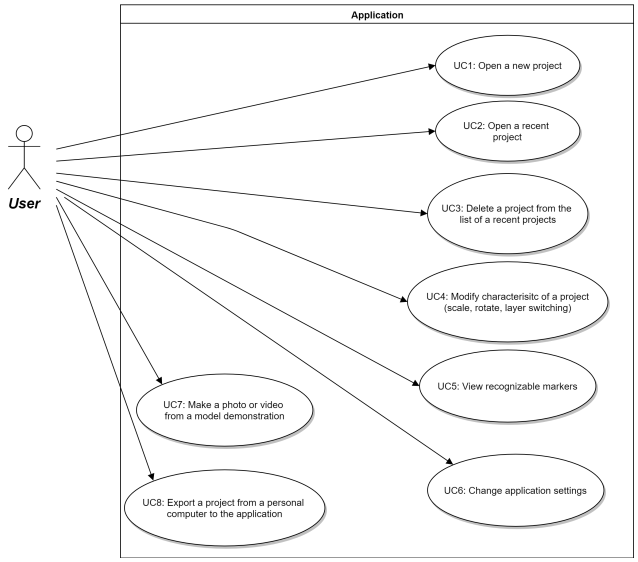

Figure 2.1: Use case diagram

2.4 Use Cases

Every use case represents a discrete task of external user-system interaction [3]. Use case modeling will help us to define a user requirements, will guide me during the implementation phase. Also we can derive test cases from use cases later.
At the end of the analysis we can clearly see that functional requirements fully define the necessary work. In order to avoid a needless duplication the use cases are described briefly and only as an illustrative diagram (see Figure 2.1).

The Table 2.1 of mapping use case to requirements is provided to verify that use cases covers all functional requirements.

When discussing the project the particular emphasis was made on a user-friendly and attractive UI. Therefore, much attention would be given to this aspect.

Table 2.1: Mapping of Use Cases to Functional Requirements

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
UC1	✓	✓	✓					✓	✓			✓		
UC2	✓	✓	✓					✓	✓			✓		
UC3														✓
UC4				✓	✓									
UC5							✓							
UC6													✓	
UC7						✓				✓				
UC8											✓			

Chapter 3

Technologies

3.1 Augmented Reality

Augmented reality is the technology of overlaying a live view of the world with a computer generated content [4]. With the help of digital technologies we create another dimension, reality that we can interact with. AR is a trending technology, so there is no surprise that many tools and possibilities of its usage are showing up at this moment. Well implemented technology has an impressive effect and is able to evoke rich emotions from users.

However, it can be seen as a trick. From the descriptions of different augmented reality products, we can derive basic constituents and everything becomes to be logical and comprehensible. When using augmented reality on smartphones the following method is applied. System has to detect characteristics of some real-world object with regard to the 6 degrees of freedom: x, y, z coordinates, yaw, pitch, roll. And then modify the 3D model, which is meant to be rendered, using this information. A model in an augmented reality is being drawn over the videoframe, during which the transformation is applied, as illustrated on the Figure 3.1.

The illusion is made the way that this object is attached to some defined point in the real world and is not just rendered like a regular model on the screen. Even though this topic is discussed for many decades, it cannot be said that it has widespread usage. The tools are still being developed and from the results it is obvious that there is a huge potential for the technology improvement.

Nowadays there are well explored technologies and tools for the rendering of 3D objects. However, the biggest complication is precise positioning of real world point with respect to a smartphone.

In the bachelor thesis by Samuel Šušla [5] the basic technologies with

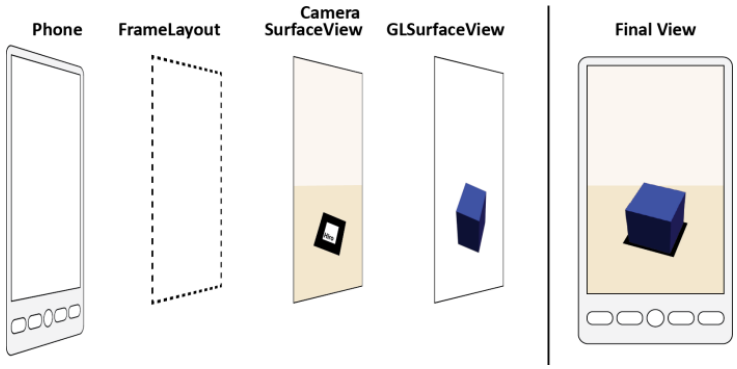

Figure 3.1: Augmented reality realisation on Android using ARToolkit, from the ARToolkit documentation[1]

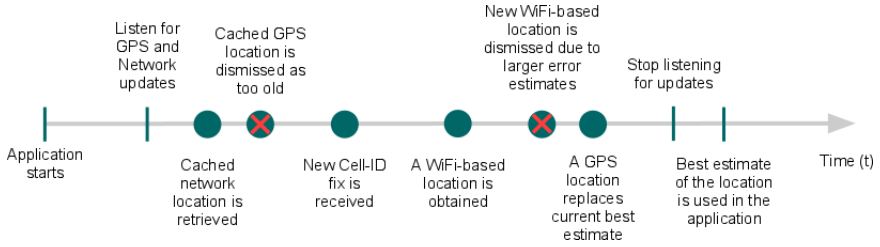

Figure 3.2: Listening to a location updates on Android system, from the Android API Guide [2]

help of which it is possible to create applications with the elements of augmented reality are described well. The results remain applicable. To avoid repetition I am going to describe only two approaches in which the evolvement can be seen and which could help to create the application that fulfills functional requirements.

3.1.1 Global Positioning System

The Android system allows to acquire a user location expressed as GPS coordinates using three technologies that are based on: Cell-ID, WiFi networks information, GPS satellite data. GPS is the most precise one from the above mentioned but it works effectively only on open area. [2]

Obtaining a user location is illustrated on the Figure 3.2.

In theory, it is possible to set GPS coordinates to a virtual object, obtain coordinates on a smartphone and to render an object on the defined point. However, there are two main problems:

1. GPS coordinates gave us the information about the positioning in plane but do not tell the information about the altitude. And con-

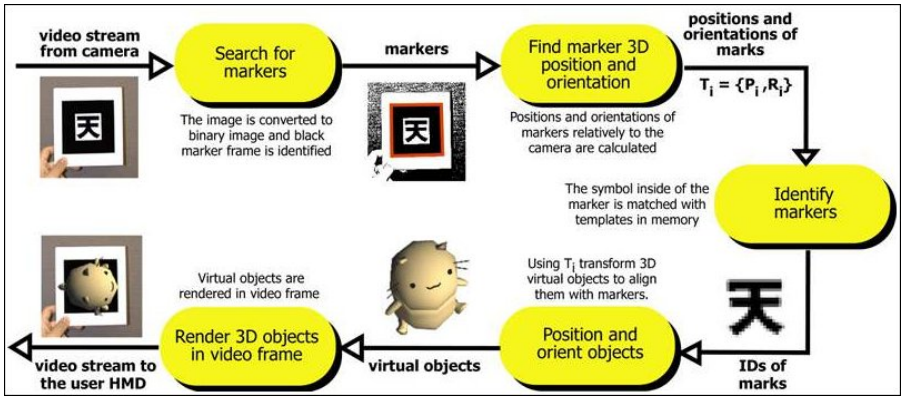

Figure 3.3: ARToolKit augmented reality workflow, from the ARToolkit documentation [1]

sequently it is not easy to calculate the correct pitch.

2. An inaccuracy in gained coordinates (up to 10-15 meters) impedes to achieve the acceptable level of realism. The mistakes in positioning lead to the constant floating of rendered models on the screen [5]. Therefore, the effect of attachment to some point is lost. Inaccuracies can be compensated with the help of ground-based reference stations. GPS technologies with Differential System compensation and RTK (real-time kinematics) allow to gain an observational error up to 2-3cm according to marketing articles [6]. However the price of 600$ did not give the chance to try this system.

3.1.2 Computer Vision Based Augmented Reality

This is the best practice of using AR on smartphones. With the help of computer vision it is possible to teach a program to see a certain type of image: 2D marker, the object of real world (a bottle, a book, etc). As soon as an image, that the system recognizes, appears on a videoframe, the system automatically determines a distance, a roll, a pitch, and sends this information to a developer for the further usage [1]. The Figure 3.3 illustrates this steps.

System functions robustly and fast. It is possible to easily find a great deal of video with the demonstration of capabilities of that approach. The method has an advantage in high speed and accuracy of recognition. Therefore, the result seems to be natural.
One of the drawbacks is the necessity to constantly keep the marker in a vision zone of a phone camera. Moreover, there have to be the following

conditions: decent illumination, visibility and perspective, which is sufficient for the proper recognition. Detailed limitations and environment requirements are described here https://artoolkit.org/documentation/doku.php?id=3_Marker_Training:marker_about.
There is a way to keep the augmented reality effect even when a marker disappears from vision zone. This method is called extended tracking.

3.2 Technical Decisions

3.2.1 Augmented Reality Libraries

I decided not to write another program based on OpenCV because there are plenty of libraries for defining necessary transformations, which encapsulate redundant complexity. I conducted a lot of research and as a result I found the three most widely used libraries. All of them suggest two options of development process:

1. To use native Android SDK, then add a library as the dependency and program in Java. Alternatively use Android NDK in case of performance optimization.

2. To use library as a plugin in Unity game engine, and then create the project as multi-platform program and then build to Android OS.

I decided to choose the first option due to reasons explained in 3.2.2. I have tried each of the chosen libraries and compared by the following parameters:

- The performance and stability of work
- Availability
- The ability to load custom markers
- The ability to load custom models in runtime

For the evaluation of each library the example project of each library was compiled and launched separately. Then within the frames of each project I tried to replace a marker and load a custom object. As a marker I used the black-white Czech Technical University logo (see Figure 3.4).

3.2.1.1 Kudan

It is a proprietary library. At the time of writing this thesis this library provided a free educational license. The actual pricing can be found on the official website `https://www.kudan.eu/`.
The performance and stability of the library are satisfactory. The library easily recognizes complex markers and the recognition is robust.
Custom marker can be created through Windows Toolkit desktop application. The library uses the proprietary file extensions for loading and rendering meshes.

Figure 3.4: CTU logo.

All custom 3D models have to be converted using GUI Window Toolkit desktop application. According to the information in official community, there were toolkits with CLI interface earlier, but at the present time they are unavailable. Developers of the library promise such toolkits in the next release. Due to that it is impossible to implement an automatic export of a projects. Therefore, functional requirement F8 and F11 is very hard to fulfill.

3.2.1.2 Vuforia

Vuforia is a proprietary library. At the time of writing this thesis this library provided a free license for development purposes. In case of publishing an application one-time 500$ fee is required. To fulfill the functional requirement 2.2 it is necessary to purchase the license at negotiable price. In the marketing videos on Kudan official web page one can see that Vuforia loses in work efficiency. However when I independently tried this library I was fully satisfied with the performance and recognition capabilities.
The marker is created with the help of online toolkit. It is necessary for the image to be contain many details and to be contrasting enough.
The library requires the usage of the 3rd party mesh loader and renderer. Therefore, developer is responsible for the file extensions support.

3.2.1.3 ARToolkit

ARToolkit is a free open-source library. The performance and quality of recognition are satisfactory.
Markers has to be in a ARuCO format. Mandatory condition is the square shape, bold black frames and the contrasting black-white image. During the creation of a marker it is necessary to make a calibration of video camera and then to export generated data into a project in binary structure.

The library also does not concern about the loading and rendering an object. A developer is responsible for the file extensions support. In comparison with Vuforia the lack of extended tracking is a disadvantage.

3.2.2 Android Development

Non-functional requirements dictate that application must run on the Android operational system. There are several opportunities how to write a program for mobiles: native development using Android SDK; hybrid development using industry standard web technologies (HTML5, Javascript, CSS3); class-platform engines such as Unity, LibGDX. I chose the first option due to several reasons:

- I have already had the experience in Android native development and I have confidence in using that toolkit. That factor helps to reduce the time spent on the development.
- Users expect that the system will follow their habits. Usage of the Android SDK allows creating UX which is familiar and organic within the framework what coincides with non-functional requirements.
- Superficial web research identified the problem with loading of the objects in Unity during runtime. Also Unity uses another programming language that I do not know. After the labor hours evaluation for learning Unity, I decided to look for alternative ways.
- Developing an application as responsive web page has some limitations. Approach is suitable for developing simple applications that do not require complex technical features or do not have unusual use cases. If we need to put an image over the videoframe in runtime, the native development is advised.

3.2.2.1 Rendering 3D Models

For the 3D objects rendering Android provides OpenGL ES 1.0 and 2.0, with Android framework APIs as well as natively with the Native Development Kit (NDK). Both of them are low-level APIs. For the time economy I decided to use LibGDX as the third party tool for loading and rendering an object. LibGDX guide recommends to use the desktop based utility fbx-conv to convert FBX/Collada/Obj files to a G3DB/G3DJ text/binary format for static, keyframed and skinned meshes, which is optimized for LibGDX. Perfomance of smartphones, file format properties and LibGDX impose some constraints on models[7]:

- The model should not weight more than 10MB.
- The G3DB format supports only 32k (java's max short) vertices per mesh. Using more vertices for a single mesh will result in unpredictable behavior.
- LibGDX supports only JPEG and PNG textures.
- Rotation/scaling the model to fit the coordinate system of LibGDX has to be done for each frame and therefore, it can drop the performance.

3.2.3 Final Technical Decision

Vuforia was chosen as AR library because of built-in extended tracking support and easy integration with LibGDX game engine.
LigGDX will be used for loading and rendering 3D models because it is free, lightweight and has sufficient performance. Moreover, it allows the usage of the specific part of the engine without the necessity to use engine application architecture instead of custom native app architecture.
I chose the native way of writing the application using Android SDK. Following the Android Developer recommendations, Android Studio was used as IDE, and Gradle was used as a build system.
Android OS has been developed for a long time, has 25 API levels and compatibility issues is important question for such complicated platform. Backward and forward compatibility is solved by setting appropriate values of minSDK targetSDK attributes of the gradle build file. On the basis of Google statistics it seems reasonable to choose the applications minimum SDK version (API version) 16 Jelly Bean, which means the application will be supported by 95.2% of devices with the Google Play market installed on them. Details are on the Figure 3.5 At the same time I can use API methods that are not available on the old versions and avoid backward compatibility issues. TargetSDK version is set to the highest possible value, this will ensure forward compatibility of the program. So when API level of device will be higher than targetSdkVersion, then additional compatibility features will be turned on to provide a correct launch of the application. Git is used as version control system. Private smartphone HTC One M7 was used as a testing device.

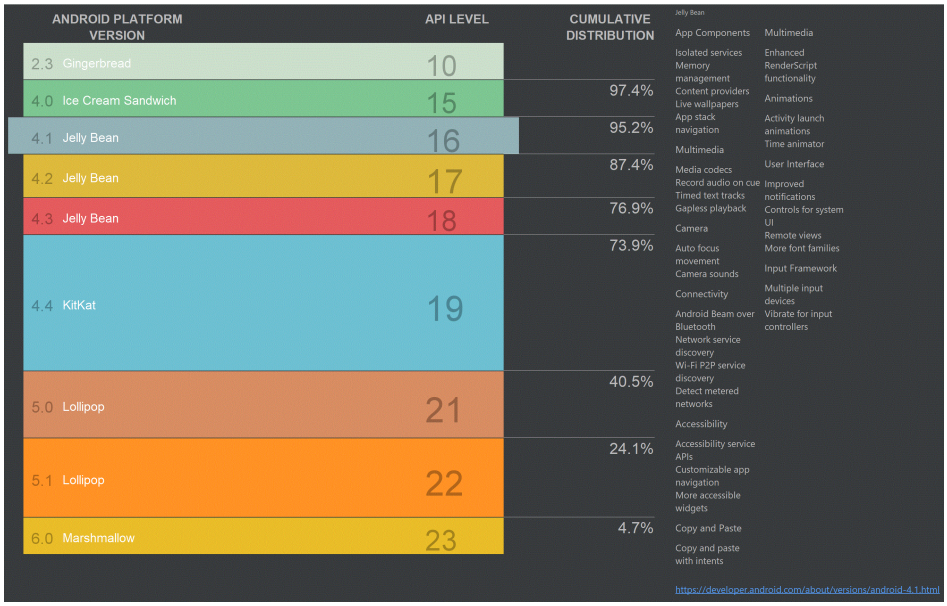

Figure 3.5: Android statistics from Android studio IDE project creation wizard

Chapter 4

Design

4.1 Domain model

Domain model is a visual representation of conceptual classes in a domain [8]. Applying UML notation, a domain model is illustrated with a set of class diagrams in which no operations (method signatures) are defined. Model has to be platform independent.

Business logic of the task is relatively simple and domain is not extensive and complex, so the domain model on the Figure 4.1 does not contain a lot of entities, though help to understand and reduce representational gap between mental and software models.

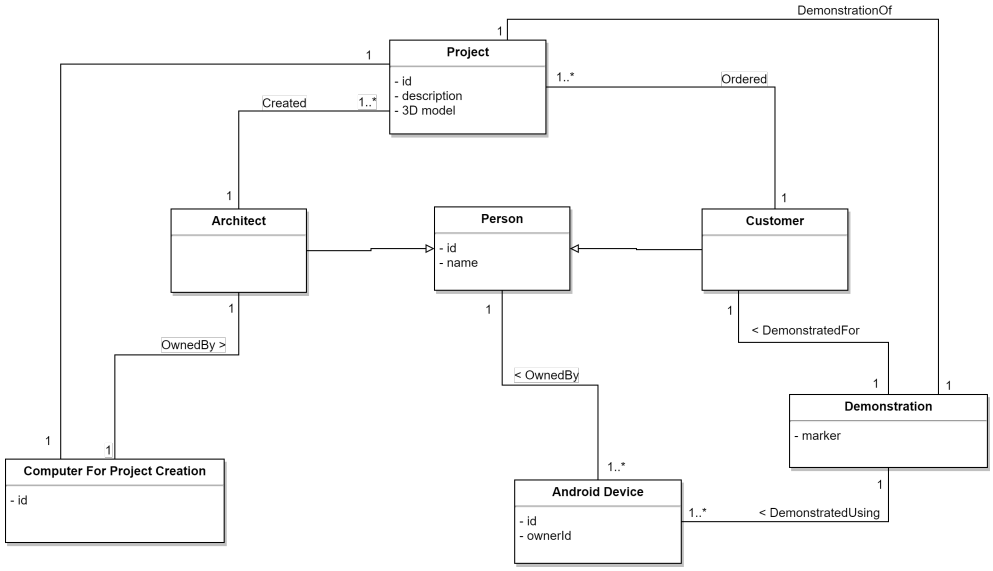

Figure 4.1: Domain model of the application

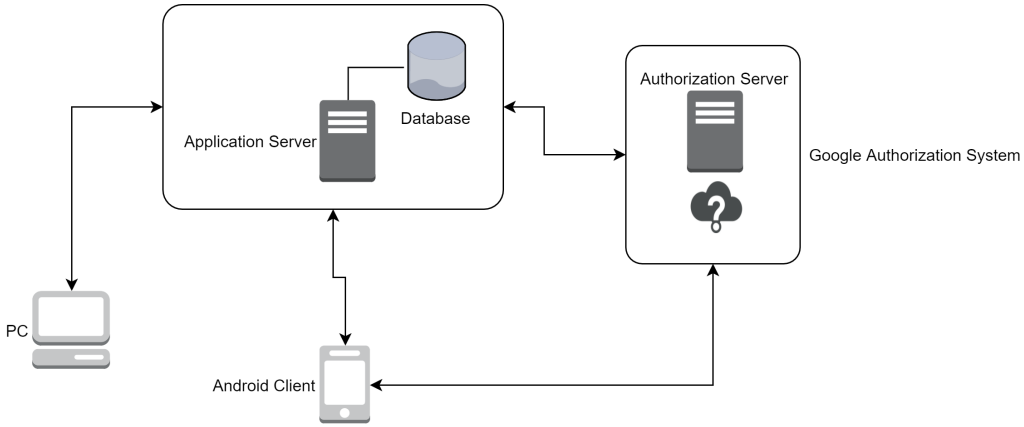

Figure 4.2: General architecture of the application

4.2 Architecture

I decided that the optimal solution will be a client-server mobile application. Server responsibilities are convertation of a project into an appropriate format and exporting projects into the mobile application.
Mobile application will be responsible for primary functionality of the project.
To describe software architecture of the project I use top-down approach and start with a high level of abstraction. Then I will continue with detailed description of server architecture and mobile app architecture.

On the Figure 4.2 we can see 3 main entities: personal computer, server and mobile app. Mobile device will subscribe to updates of certain account. For the implementation of this feature it is necessary to add registration, login and subscription options. Google services will help us with cloud notifications to mobile devices and authentication/authorization if needed.

4.3 Server Design

In the next section I design interface which will be used for the client-server communication and data interchange. Interface will be used mostly for an HTTP based resource access, so REST architecture style is suitable and will be used.

4.3.1 REST

Representational State Transfer is the realization of web-service architecture resource-oriented model. It defines several constraints: client/server

principle, statelessness, cacheability, layered system and uniform interface. Following this constraints will help us to achieve good design, interoperability and scalability[9].
Access to resources on a server should be provided through Uniform Resource Identifier (URI) and permit four basic operations: Create, Read, Update, Delete. These operations perfectly map to HTTP methods GET, POST, PUT, DELETE. Server responds with a standard code of state after the method call with respect to the method semantics.

4.3.2 Designing API

After I had distinguished main entities in the previous section 4.1 I could define the resources which will be available on the server. In order to support the export of the project to selected mobile devices it is necessary to add new entities, such as “account” and “mobile device”. Also we need to provide registration functionality.
After domain model analysis I created design of REST API and described it using Swagger.
Swagger enables creating interface and documentation at the same time, with request-response examples and export of specification in JSON format. Detailed information about parameters and responses is on the disc attached to this document as descriptive JSON source code for Swagger.
Ambiant server API provides functionality of exporting project to the mobile app with an automatic conversion to the internal file format. Project should be uploaded as ZIP archive, which includes FBX/OBJ/Collada files and textures.

List of available paths and operations:

Almost all methods are secured and authorization is needed.

- **Account management**
  - GET
    /accounts/login Signing in to gain access to other methods
  - GET
    /accounts/logout Logs out current logged in user session
  - GET
    /accounts/{id} Returns detailed information about the specific account
  - POST
    /accounts/{id} Create a new account

  - PUT
    /accounts/{id} Update an existing account
  - DELETE
    /accounts/{id} Delete an account

- **Project management**
  - POST
    /accounts/projects/{id} Create a new project
  - DELETE
    /accounts/{id} Delete a project

- **Mobile Device subscriptions**
  - GET
    /account/{id}/devices Get the list of paired mobile devices
  - POST
    /account/{id}/devices Pair new device to this account
  - DELETE
    /account/{id}/devices Delete the paired device from this account

4.3.2.1 Responses

Server will use standard HTTP semantic response codes, see Table 4.1.

Every response has JSON payload in the following format:

```
{
    "status": "success" or "error"
    "type": "method name",
    "count": integer,
    "Result": []
}
```

If everything went well the server returns 200 OK or 201 Created. 200 OK carries result object in the payload. Content of the object varies depending on the request.

Models: JSON objects that are used for the communication

Request and response can contain JSON object as the parameter or result, respectively. Validity and format are described in the following listings.

Table 4.1: Mapping of Use Cases to Functional Requirements

code	message	description
200	OK	The request was processed and returned successfully. Additional data in the payload
201	Created	The new resource was created successfully
400	Bad Request	Problem with the request, such as missing, invalid or type mismatched parameter
401	Unauthorized	The request did not have valid authorization credentials
404	Not Found	URL is wrong, or the requested resource doesn't exist
500	Server Error	Something went wrong
503	Service Unavailable	API is down. Please try again later

```
Account{
    "id":       integer ($int64)
    "name": string
    "pairedDevices": array [
        MobileDevice{
            ...
        }
    ]
    "uploads": array    [
        Project{
            ...
        }
    ]
}
```

```
MobileDevice
{
        "id": integer ($int64)
        "name": string
        "registrationToken":     string
        "imei": string
}
```

```
Project
{
        "id":   integer ($int64)
        "uploadedFilePath":     string
        "date": string ($date-time)
        "convertedFilePath":    string
        "isPublished":  boolean
}
```

4.3.3 3rd party authorization

The implementation of the functional requirements has to be convenient to an end user. For this purpose I need to include features of account management in the design of the server. Account helps to control which devices will receive updates of an account. Devices can be added or removed from the subscriptions list. Smartphones has to subscribe to account updates, so projects can be directly transferred to a mobile device. Which means that device can unsubscribe and user can logout, remove account.
Such functionality can be implemented by ourselves. We can also use 3rd party authorization mechanisms provided by Google of Facebook. That approach helps to save time and simplify the process of account creation. The similar method is used to enable Facebook and Google API authorization mechanism on the server side. We need to create and register an application in the corresponding developers consoles in order to gain API key. From now on we can use public methods of their interfaces.

4.3.4 Server Architecture

It is necessary to keep records about projects, mobile devices, exports and subscriptions on the server. For such purposes I will use database. The conceptual model of the database is illustrated as entity relationship model on the Figure 4.3.

I will use three-layered architecture to implement the server. All requests will be processed in the *RESTResources* class. This class serves as a handler for input requests. In case of complicated business logic and greater amount of endpoints it is worth splitting that class into several rest listeners, though in our case one is enough.
RESTResources class calls corresponding methods in *Services* classes. *Services* are responsible for business logic on the server.
Third layer consists of domain model entities and classes for communic-

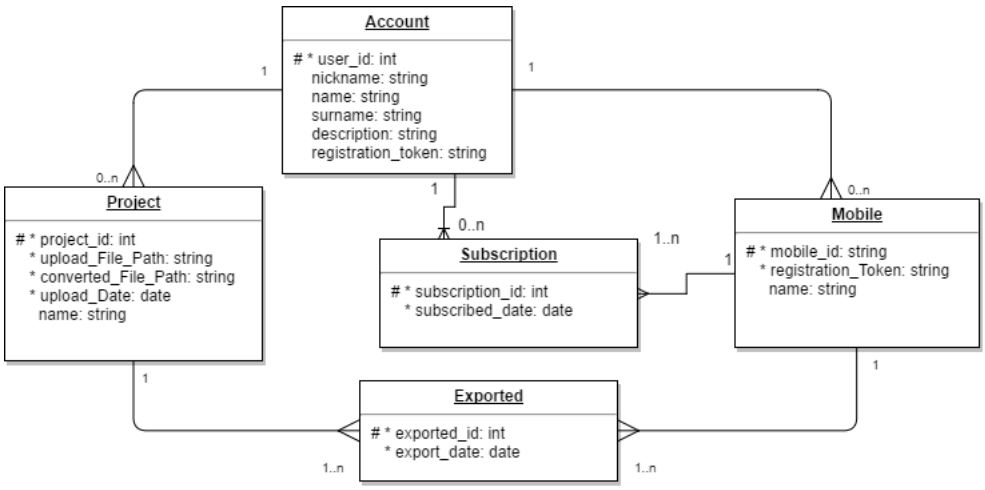

Figure 4.3: Conceptual model of the server database

ation with the database. All database processing methods are incapsulated in *repository* classes. *Model* sublayer contains entities of the domain model.
Relationships are illustrated on the Figure 4.4.

4.4 Mobile Application Design

In the beginning we are going to look at the designing of UI. Then I am going to describe architecture, class design and data interchange inside the system.

4.4.1 UI design

UI was developed as the part of MI-NUR class in winter semester of the 2016/2017 academic year, FIT CTU. In accordance with the requirements of the class the use case analysis was carried out and task graph of the application was developed in cooperation with Artyom Trushin and Radmir Usmanov. Task graph on the Figure 4.5 illustrates relationship between tasks and helps to define what screens need to be created.

Apart from the above mentioned points in the task graph we have to take into consideration platform-dependent moments of interaction with the application including trivial, but very important from the UX's point of view details, such as "return to homescreen" or "return back".
I created LoFi prototype based on the task graph analysis using the Balsamiq mockup application. Wireframe helps to quickly discover the majority of mistakes on at the early stage of the development process. In

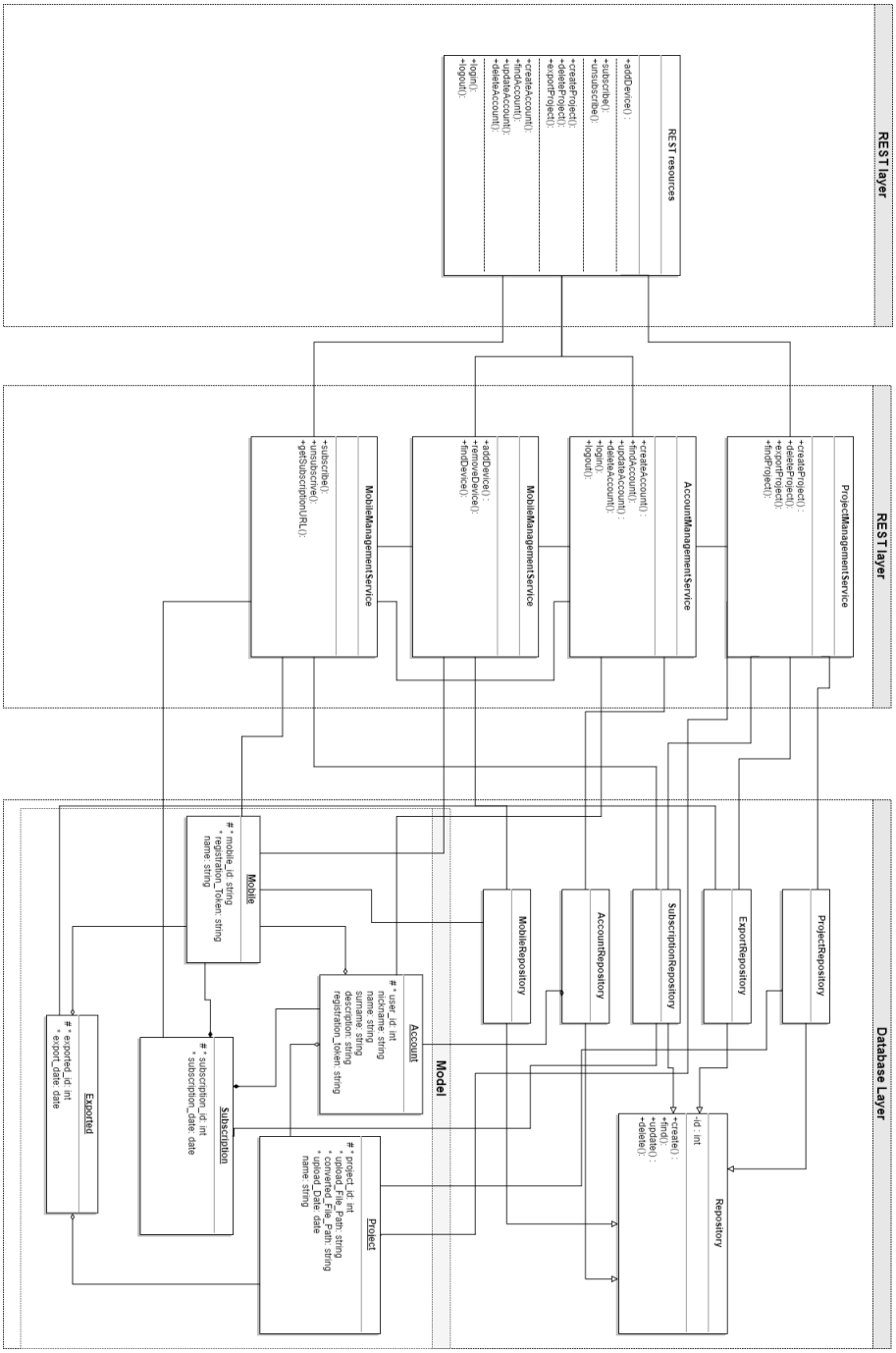

Figure 4.4: Class diagram of the server

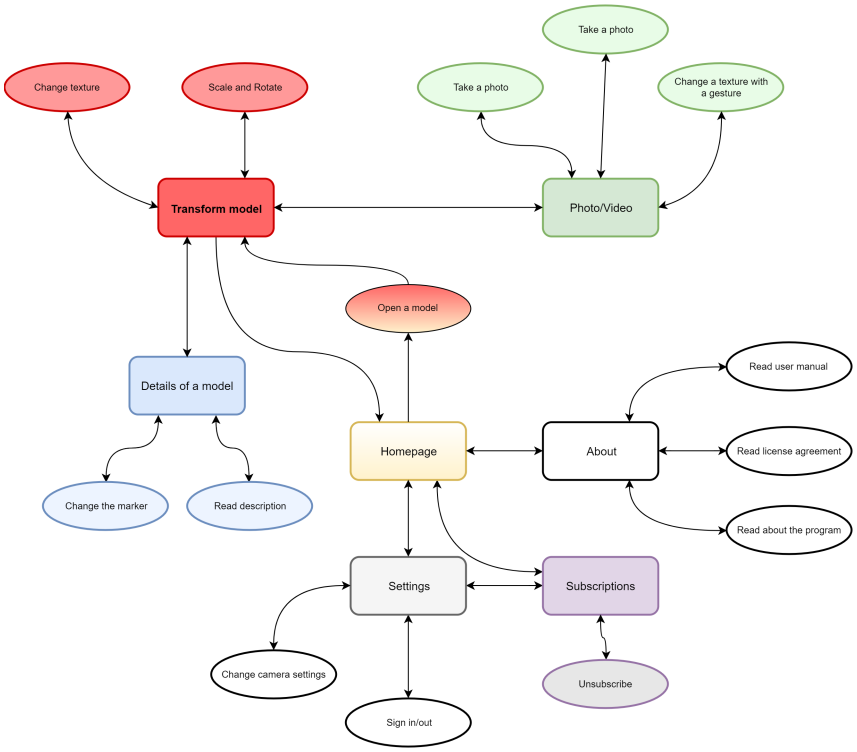

Figure 4.5: Sequential diagram of the export process

combination with the rapid prototyping process the cost of the development decreases. LoFi prototyp in the form of wireframe diagram are on the Figure 4.6 on page 30. All images can be found on the electronic attachment to this thesis on the disc.

In addition, Radmir Usmanov and Trushin Artyom carried out the analysis of similar applications in the Google Play store. Results can be found on the electronic attachment to this thesis.
After this phase I have created HiFi prototype based on this results using Android SDK. In order to ensure that designed UI is effective and user friendly we can choose from two possible testing methods: with and without a user.
Testing without a user is performed by an expert of by a developer. On of the ways to perform this testing is Heuristic evaluation, in which we analyze the design step-by-step following some rules. I think that this is the best option at the early stage, even if it is tolerant to mistakes.

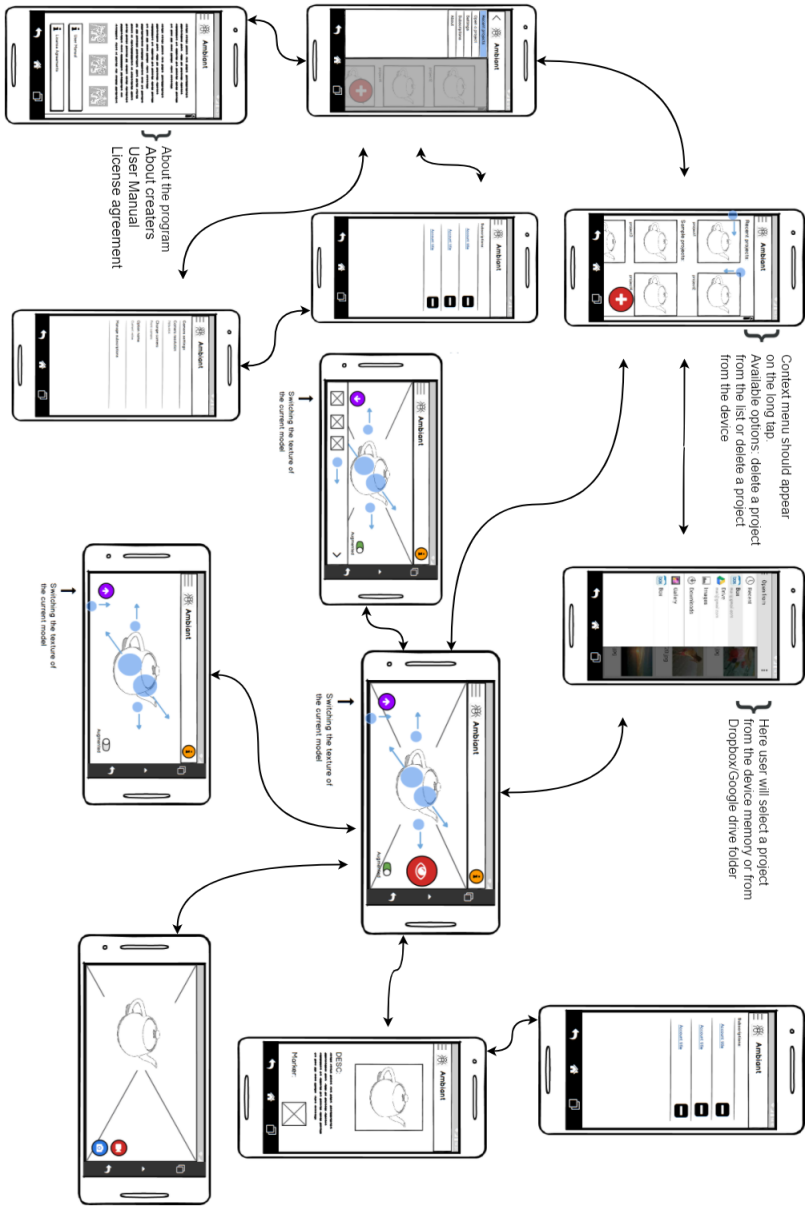

Figure 4.6: Wireframe diagram

4.4.1.1 Design evaluation

As an heuristic I used 10 rules of thumb by Jakob Nielsen[10], therefore, Ambiant mobile app was as well examined from 10 different points of view.

1. **Visibility of system status** *The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.*
   Switching between the screens is mostly instant, application responds without delay. Transition between screens corresponds to the guidelines of Android application design, therefore, does not require explicite illustration-hint.
   When all functionality will be implemented, the following UI elements are to be added:

   - Progress bar on the project opening and during long-time operations.
   - Timer on the screen of video recording.

   If a screen contains menu, the appropriate text item is highlighted. This helps user orientate in the system.
   Importance: 5/5.

2. **Match between system and the real world** *The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.*
   The application suggests where everything is situated and each icon intuitively makes it clear which action it is responsible for.
   The only issue which has been found is that on the project transformation screen the eye icon is responsible for the transition to the demonstration screen. That does not correspond to any common convention and can make users be frustrated. Perhaps it is useful to change the icon.
   Importance: 1/5.

3. **User control and freedom** *Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.*
   Later on the option of deleting the project will be added. In this regard a user will be warned about the consequences.
   Android offers generic return button, so this point is fulfiled.
   Importance: 3/5

4. **Consistency and standards** *Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.*
   Application uses standard platform components, therefore, the application is similar to the majority of android applications.

5. **Error prevention** *Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.*
   Drawbacks were found here. When opening a project it is necessary to add file extensions, file size and validity check.
   Importance: 5/5.

6. **Recognition rather than recall** *Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.*
   If a screen contains menu, the appropriate text item is highlighted. This helps user orientate in the system. The application is simple and clear from this point of the heuristic analysis.

7. **Flexibility and efficiency of use** *Accelerators unseen by the novice user may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.*
   This is only the prototype of the application, and there are not any advanced features.

8. **Aesthetic and minimalist design** *Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.*
   The application fulfils that rule of the analysis. The application contains only relevant and necessary information.

9. **Help users recognize, diagnose, and recover from errors** *Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.*
   There are no error notifications because the application does not contain any complex and important features, which can lead to er-

Table 4.2: Summary of the Nielsen analysis of the user interface

Rule#	Is Fulfilled	Importance (1-5)	Comment
1	No	5	Add progress bar during long-time operations
2	No	1	Change the icon for opening photo/video recording screen
3	No	3	Add warning when removing a project
4	Yes	-	
5	No	5	Add file validity check
6	Yes	-	-
7	Yes	-	-
8	Yes	-	-
9	Yes	-	-
10	No	5	Add user manual

rors. Later with the development of further functionality these notifications can be added.

10. Help and documentation *Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.*
There is no user manual. It has to be added. Importance: 5/5.

Summary

The result of the analysis is summarized on the table 4.2

As a result of further testing with real users we discovered that there is the lack of information about recognizable markers and option to print them, as well as viewing the information about a project from the context menu on the main screen. Corresponding item will be added. All issues have to be fixed.

4.4.2 Architecture

Android makes it difficult to write clean code because of compatibility issues and framework components' behavior. Activity and Fragments, which were initially created for managing and representing the user interface of an application, usually become god objects and contain too much

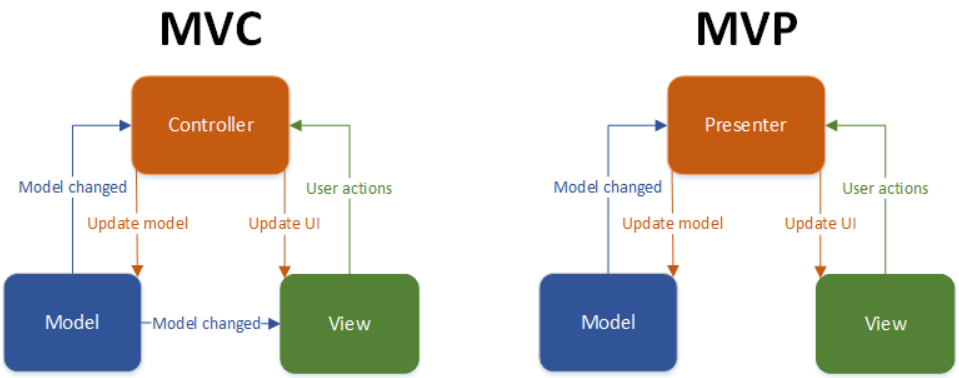

Figure 4.7: Diagram representing MVP layers structure with comparation to MVC

code[11]. I choose MVP architecture for this application, because it is more suitable for Android development due to very tight coupling between Activities and Fragments (which are MVP presenters) and various parts of Android framework[12].

4.4.2.1 MVP

Component relationships are illustrated on the Figure 4.7.

View is responsible for structure and appearance of a view on screen. Android keeps screen layouts in xml files. These xml files are processed by the system and then based on the information from the layout files the system draws elements of graphical user interface: buttons, text fields, images and so on. Every fragment and activity component of Android has corresponding look which is expressed as layout files. All direct work with graphical user interface is extracted into separate classes. The GUI elements manipulating code is placed in this layer and wrapped into convenient methods that are available to the **Presenter**.

Presenter is a mediator between the view and the model. It gets information from the **Model**, returns it in the appropriate format to the **View**, updates model if necessary. Dynamic change of the **View** and receiving of user inputs are also responsibility of the **Presenter**. In our case there are going to be fragments, activities, adapters and also several auxiliary classes to communicate with Vuforia and LibGDX in the **Presenter layer**

Model is responsible for business logic, data acquiring and keeping.

The simplified class diagram in the Figure 4.8 illustrates conceptual implementation of the application using MVP approach. During the implementation there is going to be a necessity to create new classes and utilities. However, this all will follow the MVP approach.

View

We haved defined base class MVPView. In fact, every logical screen is represented by an interface which extends MVPView. Interface exposes public methods to the Presenter. These methods corresponds with a logic of a screen. For example, the screen "About" can expose method `showUserManual()`. Direct GUI elements manipulation is available for this layer classes only.

Presenter

This layer contains Activities, Fragments, adapters, auxiliary classes. They are focused on the logic and functionality. Asynchronous work is handled by activities and fragments. Activity in this case is a logical screen of user interface. The application is divided into 2 activities: *Main Activitity* and *Augmented Reality Activity*. Every activity contains several subscreens - fragments.
Fragments are logical screens, contains relevant MVPView. Connection with the model is held through parent Activity.
Activity is connected to model and controls data related operations, also contains fixes of compatibility issues.

Model

Model is very extensive and includes several logical subdomains:

- REST Client responsible for the communication with the server
- Database Access classes
- File access classes to read, write and check validity of files
- Domain model classes. There can be several of them for different model representations depending on consumer. For Presenter they can be different from models for Database or REST client.

The Model also contains necessary utilities and classes for asynchronous computation, notifications handlers and etc.
Presenter usually communicates with Model through interacts or repository. They express use cases in model and encapsulate data acquiring.

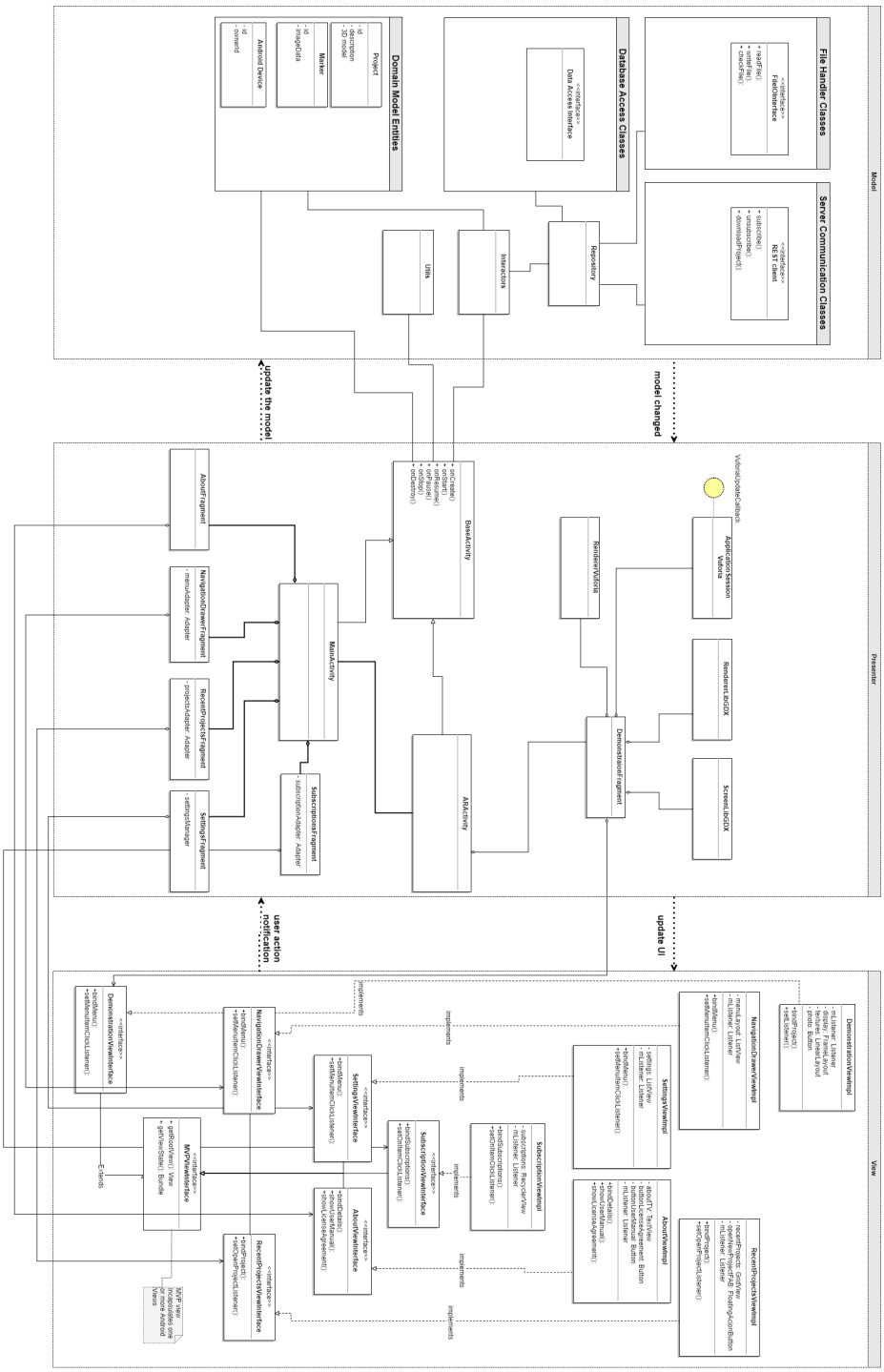

Figure 4.8: Mobile Application Class Diagram

4.4.3 Storing Data

Android offers several ways to store data on the device.

Shared preferences are used to keep key-value pairs. This method is designed to store small amounts of information. I will use shared preferences for settings.

Storing files on the file system is a straightforward method to store more data in various forms. Android offers File API to work with File reading/writing.

A File object is suited to reading or writing large amounts of data in start-to-finish order without skipping around. For example, it's good for image files or anything exchanged over a network.[13]

More complex and structured data can be stored in the relational SQLite database on the device.

I am going to use Android database API to store necessary information. If it will not be enough or suitable, a developer can easily switch to another approach due to layer architecture. With the help of Android API for database manipulations developer can create tables and write SQL requests.

Constants, app images, strings, colors, styles, dimensions are meant to be stored in the resources of the application. This is a special folder that is packed into installation file. Developer can also provide alternative resources for different device configurations or locales, by grouping the into specially-named resource directories. Resources are packed in the APK installation file at the building stage. At runtime system detects configuration and accesses necessary resources using automatically generated resource ID.

String will be provided in Russian, Czech and English languages. System locale will define which language of the application will be displayed.

4.5 Export of the project

On the Figure we can see the illustration of export process.
In the beginning, a user will upload a project to the server.
Then server keeps the project and notifies subscribed mobile application

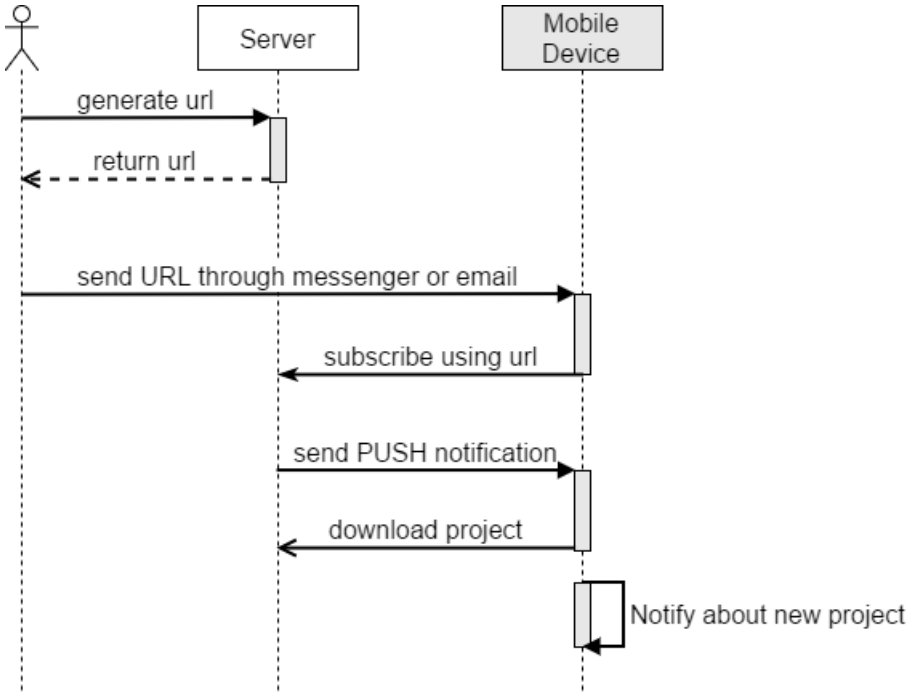

Figure 4.9: Sequential diagram of the export process

about update using cloud messaging. Server send the URL of a project to download.
Mobile app downloads file when it is appropriate and notifies a user about the update using Android UI notifications.

4.6 Subscription

I have designed the following method of subscription.
Subscription is to be allowed only with the permission of an account owner. Server returns automatically generated URL to an account owner on demand. That URL has to be opened on a mobile device. An account owner sends this URL to certain device via email/SMS/messengers. If the application is installed on a smartphone, URL opening is intercepted by the application and device subscribes to account updates.
If the application is not installed, then URL leads to special page with instructions how to install the application and subscribe to an account.
The sequence diagram on the Figure 4.9 is applicable for that use case. In case user wants to subscribe to an account from mobile device, he/she has to enter special code and generated subscription request will be sent to the server.

There are two methods how to unsubscribe:

- Unsubscribe from mobile application settings menu using special registration token.
- Particular device will be removed from a subscription list through web user interface of the server application.

4.7 Cloud Messaging

Firebase Cloud Messaging by Google gives an opportunity to send messages from a server to mobile devices up to 4KB in size[13] Messages are sent from application server to a client application via the chosen Firebase Cloud Messaging connection server, using HTTP or XMPP protocol. FCM servers are provided by Google and responsible for delivering messages to devices. Receiving the message is invisible for device owner.

Chapter 5

Implementation

Guided by the task of the thesis, I had to implement the prototype of the mobile application and provide an option of convenient export of a project, based on the analysis and design. By agreement with supervisor I have partly implemented functional requirements. The application is not complete and ready for publication.

The server was implemented without the feature of registration and multi user support. In all other respects implementation was carried out in accordance with analysis. Therefore, these features are not in the mobile application, too. Server provides simple HTML form for file uploading.

During the implementation of the mobile application I followed the architecture designed in the previous chapter. There are too many classes, so I will describe only few of them. In the “Model” layer the most interesting classes are related to the REST server communication and database interaction.

5.1 REST client

Server client in the Android app was implemented with the help of Retrofit library by Square inc guided by the official documentation. For the communication with the server there is the need for Interface which defines the possible HTTP operations and Retrofit.Builder class instance which uses the interface.

Realization of interface is on the listing 5.1. It defines three endpoins of the server: to subscribe, to unsubscribe and for testing purposes. Annotations before interface methods declarations define which

Listing 5.2: Retro.Builder usage

```
Retrofit mRetrofit = new Retrofit.Builder()
                .baseUrl(BASE_URL)
                .addConverterFactory(GsonConverterFactory.create())
                .build();

AmbiantServerApi mAmbiantServer = mRetrofit.create(AmbiantServerApi.class);

Call<ResponseBody> call = mAmbiantServer.unsubscribe(accountId, deviceId,
    token);
```

HTTP method and path of the request should be used. AccountBriefInfo and ResponseBody classes are model classes which are used to map the JSON response data to java classes. Method parameters can be annotated as @Path, @Body, @Header depending on HTTP parameters placement.

Listing 5.1: AmbiantServerApi.class

```
package fit.cvut.cz.ambiant.model.server_communication;

import okhttp3.ResponseBody;
import retrofit2.Call;
import retrofit2.http.Body;
import retrofit2.http.DELETE;
import retrofit2.http.GET;
import retrofit2.http.POST;
import retrofit2.http.Path;

public interface AmbiantServerApi {
    // Request method and URL specified in the annotation
    @POST("accounts/{id}/devices/{deviceid}")
    Call<AccountBriefInfoPOJO> subscribe(@Path("id") int id, @Path("
        deviceid") int deviceId, @Body String subscriptionToken);

    @DELETE("accounts/{id}/devices/{deviceid}")
    Call<ResponseBody> unsubscribe(@Path("id") int id, @Path("deviceid")
        int deviceId, @Body String subscriptionToken);

    @GET("https://jsonplaceholder.typicode.com/posts")
    Call<ResponseBody> testAPI();
}
```

Then we need to use a builder class to be able to send requests to our server. Snippet on the listing 5.3are placed in the RESTClient class.

Now we can send a request synchonously with `call.execture()` or asynchronously with `call.enqueue(Callback<T> callback)`

Asynchronous call can be examined on the Figure 5.1

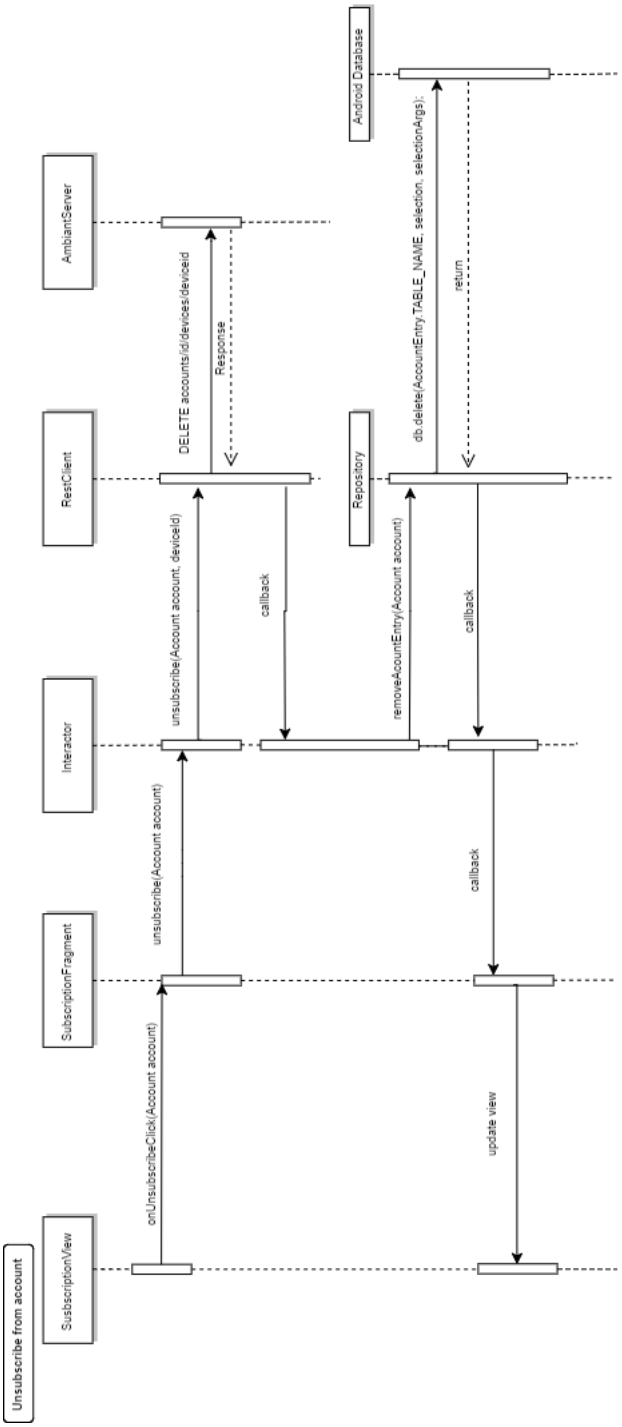

Figure 5.1: Sequential diagram of the export process

5.2 Android database

Operations with database are implemented in the Repository.class, which extends SQLiteHelper and includes DAO (Database Access Object) classes for each table. Repository should be a singleton class during app lifecycle to prevent memory leaks. Each DAO class contains static methods as presented on the listing.

Listing 5.3: ProjectEntryDAO.class createProject method example

```
ProjectEntryDAO.java


    public static long createProjectEntry(SQLiteDatabase db, Project
        project) {
        // Create a new map of values, where column names are the keys
        ContentValues values = new ContentValues();
        values.put(ProjectEntry.COLUMN_NAME_TITLE, project.getName());
        values.put(ProjectEntry.COLUMN_NAME_DESC, project.getDescription())
            ;
        values.put(ProjectEntry.COLUMN_NAME_AUTHOR, project.getAuthor());
        values.put(ProjectEntry.COLUMN_NAME_LAST_OPENED_DATE, seconds);
        values.put(ProjectEntry.COLUMN_NAME_PATH, project.getPath());
        long rowID = db.insert(ProjectEntry.TABLE_NAME, null, values);
        return rowID;
    }

...
```

SQLite database transactions and communication with the application server are long-term operations[13]. By default the whole application functions in a single main thread. The main thread is responsible for sending events to the appropriate UI widgets, including drawing events. If we run long-term operations on the main (UI) thread, the application will cause a noticeable lag in the app. In the worst cases the system can offer to kill the application instance. That is why it is necessary to conduct long-lasting operations in the parallel worker thread.

For the code running in the parallel thread Android offers several options, such as java Thread, Service, Handler and AsyncTask. I chose AsyncTask because it is easy to use and completely satisfies needs. AsyncTask provides developer with four methods:

onPreExecute() this method is being called on the UI thread before the task execution.

doInBackground(Params...) this method is computed on the background thread immediately the previous method onPreExecute finishes. This

step is used to perform background computation that can take a long time. This method can also use publishProgress(Progress...) to publish inform about a progress. These values are published on the UI thread, in the onProgressUpdate(Progress...) step.

onProgressUpdate(Progress...) invoked on the UI thread after a call to publishProgress(Progress...).

onPostExecute(Result) invoked on the UI thread after the doInBackground() finishes. Result parameter is a return value of doInBackground() method.

Here is example of communication with database using AsyncTask.

Listing 5.4: Reading projects from a database using AsyncTask and Repository

```
1  public class RecentProjectsAsyncTaskLoader extends AsyncTask<Void, Void,
       ArrayList<Project>> {
2
3      private final Interactor.RecentProjectsLoadedListener listener;
4      private final Repository repository;
5
6      public RecentProjectsAsyncTaskLoader(Interactor.
           RecentProjectsLoadedListener listener, Repository repository) {
7          this.listener = listener;
8          this.repository = repository;
9      }
10
11     @Override
12     protected ArrayList<Project> doInBackground(Void... params) {
13         return repository.readRecentProjectsEntries();
14     }
15
16     @Override
17     protected void onPostExecute(ArrayList<Project> result) {
18         super.onPostExecute(result);
19         // add example projects
20         Project project = new Project(1,"Combat jet", null, "file:///
               android_asset/example_models/combat_jet/thumbnail.jpg", "Combat
                jet description", true);
21         result.add(0, project);
22         listener.onRecentProjectUpdate(result);
23     }
24 }
```

5.3 Screenshots

Screenshots of HiFi prototype are on the Figure 5.25.3.

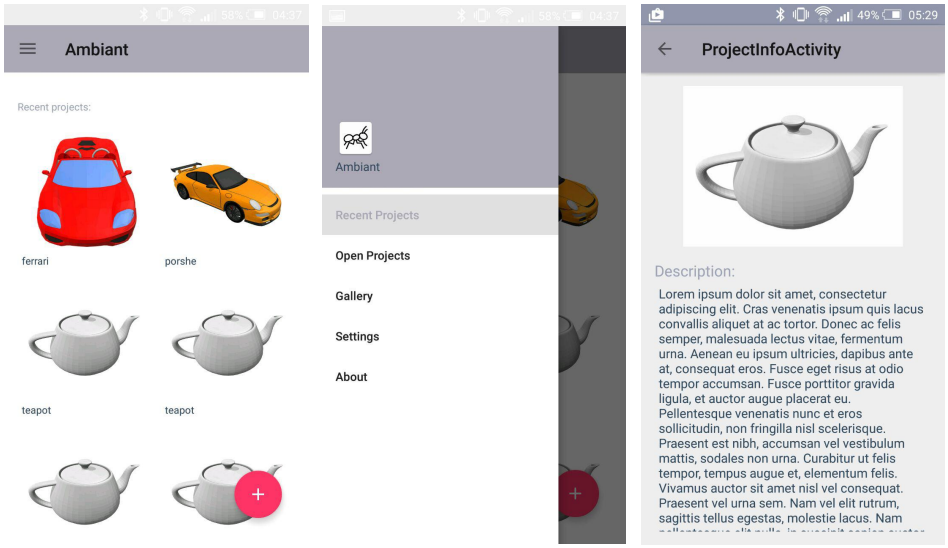

Figure 5.2: Screenshots of Ambiant app

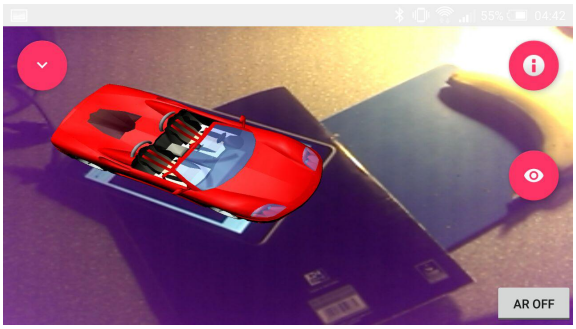

Figure 5.3: Screenshot of Ambiant app

5.4 Deployment

Deployment diagram of the server application and mobile application is on the Figure 5.4. Server machine has to have installed java runtime environment and also machine should provide an access through ssh. Developer must build the server application using gradle command `gradle build`. Then builded jar must be uploaded to a server machine using ssh or any other way. On the server machine jar application can be launched with the terminal command `java -jar ambiant-server-app.jar`.

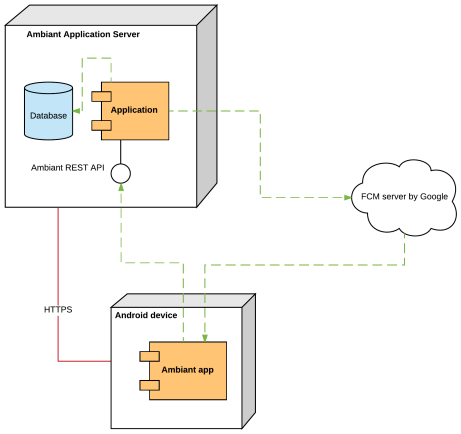

Figure 5.4: Deployment diagram

5.4.1 Installation of Android app

First of all, if the application is not published on the Google Play market, then it necessary to change setting on the mobile device. On smartphone or tablet running Android 4.1 or higher go to *Menu*, select *Settings*, scroll down to *Security*, and check *Unknown sources*. This option will allow installation of apps outside of the Google Play store. Depending on device, warning can popup.
Mobile application is supposed to be build with gradle script. Either gradle console command or android studio IDE can be used for this purpose. In the result there will be apk file in the app module folder. This file has to be transferred to android device via wired connection or any other way. Now it will be able to search for the file location in the My files folder of a device. To install the application, find the APK file, tap it, then select “Install”. And now app will be installed and ready to launch. The launcher icon should appear in an app listing.

5.5 Android Permissions

In order to function user have to grant the following permissions:

INTERNET allows the application to communicate with a server over the Internet.

WRITE_EXTERNAL_STORAGE makes it possible to read/write files using File API. It is needed to download and correctly open 3D models.

CAMERA is a key hardware component for augmented reality part of the application.

ACCESS_NETWORK_STATE allows to optimize network communication, for example to download project using WiFi connection only.

Chapter 6

Testing

Software testing can demonstrate to all interested parties that product meets the requirements that were defined and has satisfactory quality. Moreover, testing can expose bugs and detect a program behavior that varies from the expected. This chapter describes which test techniques were applied and what results were obtained.

6.1 Unit tests

Unit test were carried out to validate functionality of a few components. Tests are composed in a way to verify the initialization correct execution of the specified parts of the unit. For testing purposes I have chosen the following classes: RESTClient, Repository and Interactor. Tests were written with the help of JUnit framework, which is provided with the Android SDK. It also includes a virtual mobile device emulator that can be launched on a desktop OS. The emulator lets test Android applications without using a physical device and change configurations including the memory size, connection speed, *etc.* Tests were launched on the Android virtual device with these specifications:

- CPU: ARM
- Target: Google APIs (API level 16)
- Skin: 768x1280
- SD card: 200M
- emulated camera

Tests were written in accordance with the guide published by Google on the `https://developer.android.com`. Additionally the standard Android logging tools were used to monitor the state of a variables and application workflow.

6.2 Usability testing

Usability is a measure of the quality of a user's experience in interaction with a product or system [14]. In short form, it simultaneously means effectivity, ease of use and user-friendliness. Usability testing attempts to quantify the usability of Ambiant mobile app by observing the people playing with the app for the first time. The test was held in citu (*latin* "on site"). Such tests are time consuming and difficult to organize but the result is very valuable for product development. Tester tried to note several aspects during each test with a user:

- User should find the specific feature or option without mistakes.
- User should find the specific feature or option quickly.
- Does user has any frustration during the app usage.
- General impression from the app.

6.2.1 Test Scenario

For testing purposes the scenario was created on the basis of use cases and implemented features. Every tested person had to complete every task of the scenario step by step. The application was freshly installed and ready to use.

1. User has to open an example project and observe it on the marker using augmented reality.
2. User has to open a project from a device memory and observe it on the marker using augmented reality.
3. User has to examine detailed information about a project.
4. User has to scale and rotate an object during the demonstration.

This scenario was given to 7 people. The detailed testing is described in the Appendix C.

6.2.2 Summary

During the test the next issues were revealed:

- User could not find the information about markers. There is necessity to add marker section to the main menu. Importance: 5/5.
- Add user manual inside the application. Importance: 5/5
- Fix the behavior of the back button navigation. Importance: 5/5
- Add context menu on the main screen, so users can find the information about a model. Importance: 4/5

Conclusion

Task analysis

The aim of this thesis was to design and implement a prototype of a mobile application for the Android platform.
Defined use cases and FURPS requirements can be found in the chapter 2.2.
I created the server architecture for project export, mobile application architecture, including class diagrams and database models. The result can be found in the 4 chapter. User interface creation is described in the section 4.4.1.
The final implementation partially fulfilled the functional requirements, but the prototype is working and can be presented to the customer as a result of agile development iteration. The intermediate result can be discussed with the customer in order to implement high quality final software product.
The user friendly export mechanism was created on the server side, but without additional features such as multiuser support and web application for account management. Mobile application was tested with unit tests and usability test. Tests are described in the 6 chapter.

By my opinion, the given task was fulfilled not perfectly, but acceptably. The project turned out to be more complex than it was expected in the beginning due to several reasons. First of all, Android devices has limitations about 3d object rendering and Android SDK provides only lowlevel APIs for rendering. Additionally, augmented reality tools are in their early stage of development, but it is the price of using cutting edge technologies.

Future improvements

All functional requirements will be implemented. Issues, which were revealed in the test phase, will be fixed. Server definitely needs the web application for account management. A plugin for 3D editing desktop program can be implemented to provide one-click export of a project to mobile device. If effective and reliable way of long distance detecting of 6POF appear in the future, the application can be developed as a very impressive tool for real-time object visualizations in real world surroundings.

Bibliography

[1] ARToolKit. *ARToolKit Documentation, Marker Training [online]*. [cit. 2017-04-21]. Available from: https://artoolkit.org/documentation/

[2] Android Open Source Project. *Location Strategies [online], Android API Guide*. [cit. 2017-04-21]. Available from: https://developer.android.com/guide/topics/location/strategies.html

[3] Sommerville, I. *Softwarové inenýrství*. Brno: Computer Press, first edition, 2013.

[4] Sung, D. What is Augmented Reality? [online]. february 2011, [Cited 2017-04-21]. Available from: http://www.pocket-lint.com/news/108880-what-is-augmented-reality-ar

[5] ula, S. *Building (3D object) visualization using augmented reality on iOS: BACHELOR THESIS*. Praha: VUT v Praze, Fakulta informaních technologií, 2015. Available from: http://hdl.handle.net/10467/63142

[6] Gurylev, V. Reach: 1cm precision GPS [online]. may 2015, [Cited 2017-04-21]. Available from: https://habrahabr.ru/company/intel/blog/258779/

[7] daemontus. Vuforia and LibGDX 3D model renderer [online]. june 2016, [Cited 2017-05-09]. Available from: https://treeset.wordpress.com/2016/06/12/vuforia-and-libgdx-3d-model-renderer/

[8] Larman, C. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Prentice Hall PTR Upper Saddle River, third edition, 2004.

[9] Erl, T.; Carlyle, B.; et al. *SOA with REST*. Prentice Hall, fifth edition, 2013.

[10] Nielsen, J. 10 Usability Heuristics for User Interface Design. january 1995, [Cited 2017-05-09]. Available from: https://www.nngroup.com/articles/ten-usability-heuristics/

[11] senneco. Moxy - realization of MVP for Android with a piece of magic. february 2016, [Cited 2017-05-09]. Available from: https://habrahabr.ru/post/276189/

[12] Zukanov, V. MVP and MVC Architectures in Android. july 2015, [Cited 2017-05-09]. Available from: https://www.techyourchance.com/mvp-mvc-android-1/

[13] Android Open Source Project. *Android Guide*. [cit. 2017-05-09]. Available from: https://developer.android.com

[14] Usability Evaluation Basics. [Cited 2017-05-09]. Available from: https://www.usability.gov/what-and-why/usability-evaluation.html

Appendix A

Acronyms

GUI Graphical user interface

XML Extensible markup language

REST Representational State Transfer

HTML Hypertext Markup Language

JSON JavaScript Object Notation

API Application Programming Interface

AR Augmented Reality

GPS Global Positioning System

SDK Software Development Kit

FURPS Functionality, Usability, Reliability, Performance, Supportability

Appendix **B**

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `executables` ..... the directory with executables
- `src` ..... the directory of source codes
  - `server` ..... application server sources
  - `android app` ..... android application sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format
- `attachments` ..... the directory with electronic attachments
  - `wireframes` ..... the directory with wireframes of the mobile app
  - `serverAPI` ..... server API description in Swagger format
  - `misc` ..... miscellaneous

Appendix C

In citu usability testing

TESTER 1

Name: Timur Bugayevski
Sex: male
Age: 25yo
Occupation: front-end developer
Everyday phone OS: Android
Comment: User had a frustration about the back button behaviour.

Task 1
User asked about a marker. After opening a demonstration screen user did not know about any marker and what he has to perform to observe a model. After providing a marker he was happy to view a model.
Task 2
Everything went fine.
Task 3
Opened the About screen by mistake. Otherwise everything was fine.
Task 4
The task was successfully passed.

TESTER 2

Name: Tokar Zakhar
Sex: male
Age: 25yo
Occupation: Student Fakulty Strojní na ČVUT v Praze. Obor matematická

analýza a matematické modelování.
Everyday phone OS: Windows.
Comment: Frustration about the back button behaviour.

Task 1
Used did not know about a marker. After the help task was solved quickly.
Task 2
Confused about Android default back and home buttons due to lack of experience with android devices.
Task 3
Context menu was expected. Tried every screen and every button. Finally, the task was solved.
Task 4
The task was successfully passed.

TESTER 3

Name: Pozynych Julia
Sex: female
Age: 23yo
Occupation: architect
Everyday phone OS: Android
Komentá: Confused about standard back behaviour.

Task 1
Again, the problem with a marker. Quickly found the information about a marker inside of project description. Wants the feature of printing a marker directly from the app.
Task 2
Successfully solved.
Scéná 3
Successfully solved.
Scéná 4
Successfully solved.

TESTER 4

Name: Karina Usmanova
Sex: female

Age: 27yo
Occupation: Studentka VŠE
Everyday phone OS: iOS.
Comment: back button issue
Task 1
Successfully solved.
Task 2
Confused about what to do. Successfully passed this step with little help.
Task 3
Successfully solved.
Task 4
Successfully solved.

TESTER 5

Name: Vlad Stepanov
Sex: male
Age: 21
Occupation: Student VŠE
Everyday phone OS: iOS.
Komentá: tested person never had an experience with Android smartphones, so he had a problems with navigation.
Task 1
Successfully solved.
Task 2
Tried to open different files, but they were not valid. He suggested to start navigation from defined folder.
Task 3
Successfully solved.
Task 4
Successfully solved.

TESTER 6

Name: Volodymyr Kryzh
Sex: male
Age: 24yo
Occupation: Student VŠE
Everyday phone OS: iOS.

Comment: User had an experience with Android.
Task 1
Could not find an information about markers.
Task 2
Successfully solved.
Task 3
Successfully solved, but commented that it is not intuitively implemented.
Task 4
Successfully solved.

TESTER 7

Name: Alexandr Lustin
Sex: male
Age: 19yo
Occupation: Student
Everyday phone OS: iOS.
Comment: User has a little experience with Android OS.
Task 1
Marker issue.
Task 2
Did not know which button to press. Successfully solved with a little help.
Task 3
Successfully solved.
Task 4
Successfully solved.