Masaryk University
Faculty of Informatics

Mobile SDK for YSoft SafeQ print and scan management solution

Bachelor's Thesis

Miriam Cabadajová

Brno, Spring 2020

Masaryk University
Faculty of Informatics

Mobile SDK for YSoft SafeQ print and scan management solution

Bachelor's Thesis

Miriam Cabadajová

Brno, Spring 2020

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Miriam Cabadajová

Advisor: Mgr. Juraj Michálek

Acknowledgements

I would like to thank my supervisor Mgr. Juraj Michálek for bringing up such an exciting assignment for this bachelor's thesis as well as for frequent consultations and valuable insights on issues I have encountered. Also, I want to express my gratitude to my friends and family for the endless support.

Abstract

This bachelor's thesis aims to create a mobile SDKs with the functionality that enables the use of the YSoft SafeQ product by partners of YSoft Corporation a.s., for the development of customizable mobile applications native to Android and iOS platforms. The research phase inspects the SDK development methods of companies such as Google and Apple. The findings acquired on business managed SDKs are utilized in the implementation part. The developed SDKs are then validated by creating two sample mobile applications, one for each platform, demonstrating the use of mobile SDKs. The scope of this thesis also includes the gathering of feedback from mobile application users as well as the software engineers.

Keywords

mobile print, Y Soft, YSoft SafeQ, SDK, mobile SDK, iOS, Android, Google, Apple, IPP, Internet Printing Protocol, mDNS, JmDNS

Contents

1 Introduction 1

2 Research about mobile SDKs 3

2.1 *Methodology of research* . . . 3
2.2 *Software development kit* . . . 4
2.3 *Google's SDK* . . . 4
2.3.1 Developer community management . . . 4
2.4 *Developer's guide to Android SDK* . . . 7
2.4.1 Application fundamentals . . . 8
2.4.2 Architectural principles . . . 8
2.4.3 Development guidelines . . . 9
2.5 *Apple's SDK* . . . 10
2.5.1 Developer community management . . . 11
2.6 *Developer's guide to iOS SDK* . . . 13
2.6.1 Application fundamentals . . . 13
2.6.2 App Store Review Guidelines . . . 14
2.6.3 Human Interface Guidelines . . . 15
2.7 *Comparison of Android and iOS SDKs* . . . 16

3 YSoft SafeQ mobile SDK 18

3.1 *Architecture of YSoft SafeQ in mobile printing* . . . 18
3.1.1 Print job delivery via end user interface . . . 18
3.1.2 Print job delivery via Mobile Integration Gateway 20
3.1.3 Mobile Print Server vs Mobile Integration Gateway . . . 20
3.2 *Functional requirements for SDKs* . . . 21
3.3 *The mobile SDKs delivery requirements* . . . 21
3.4 *Service discovery implementation* . . . 21
3.4.1 Local printers discovery . . . 22
3.4.2 Implementation of local printers discovery in Android . . . 22
3.4.3 Implementation of local printers discovery in iOS 22
3.5 *EUI delivery implementation* . . . 23
3.5.1 EUI login and upload general workflow . . . 23
3.5.2 Implementation of EUI workflow in Android . . 23

3.5.3 Implementation of EUI workflow in iOS 24
3.6 *MIG delivery implementation* 24
3.6.1 MIG login and upload general workflow 24
3.6.2 Implementation of MIG workflow in Android . 24
3.6.3 Implementation of MIG workflow in iOS 25
3.7 *Developer's guide for Android SDK* 25
3.7.1 Sample application 25
3.7.2 Implementation 25
3.7.3 Project setup 25
3.7.4 Discovery 25
3.7.5 Login 27
3.7.6 Upload 28
3.8 *Developer's guide for iOS SDK* 29
3.8.1 Sample application 29
3.8.2 Implementation 29
3.8.3 Project setup 30
3.8.4 Discovery 30
3.8.5 Login 31
3.8.6 Upload 32

4 Validation **34**
4.1 *Mobile SDKs* 34
4.1.1 Developer feedback 34
4.2 *Mobile applications* 35
4.2.1 Sample mobile applications 35
4.2.2 YSoft SafeQ Mobile Print application 38
4.3 *Extension of mobile SDK* 40

5 Conclusion **41**

Bibliography **43**

A An appendix **47**
A.1 *Discovery workflow* 47
A.2 *EUI workflow* 47
A.3 *MIG workflow* 47

List of Tables

2.1 Differences between Android and iOS SDKs *16*

List of Figures

3.1 The EUI job upload workflow *19*
3.2 The MIG job upload workflow *19*

4.1 Login screens of sample applications *36*
4.2 Upload screens of sample applications *37*
4.3 The rate of usefulness of the mobile application *38*
4.4 Android active users *39*
4.5 iOS active users *39*

A.1 Service discovery *48*
A.2 The EUI login and upload *49*
A.3 The MIG login and upload *50*

1 Introduction

Mobile technologies have been evolving rapidly in the last decade. As their functionality improves and becomes enhanced, people utilize mobile devices more than ever before. They operate our lives, and one can confidently say that many are incapable of envisioning the execution of the simplest daily tasks without them. However, personal life is not the only sphere affected by this phenomenon. It is fittingly applicable to the business sector as well.

According to Deloitte's Mobile Consumer Survey, smartphone penetration has reached 90% after rising continuously over the last four years, with 95% of adults using them daily [1]. Because of the capabilities smartphones present, some enterprises are taking on actions to create mobile-first and even mobile-only workforce. The demand on the market for mobile applications rises, and they are becoming a truism when supplying a product such as SafeQ provided by Y Soft.

YSoft SafeQ is a workflow platform offering solutions around three main pillars: print management, document capture, and 3D print management. The concept of a mobile application for YSoft SafeQ arises from the idea of higher convenience for the customer, increasing the value of the product. Detaching from the duty of carrying a laptop, the users will have an opportunity to avail the services of YSoft SafeQ using mobile devices such as smartphones or tablets as those will offer sufficient mechanisms.

Y Soft offers SafeQ to its partners, which are represented by companies. Every customer has distinct requirements for the mobile application. A company may want to modify the application according to their needs (own color schemes, logo, unique functionality, etc.). This scenario is hardly manageable as one mobile application could never satisfy the needs of all the customers at once. An apparent solution to this issue would be to give the implementation of the mobile application over to the partners.

However, the software complexity of the YSoft SafeQ makes the implementation of the functionality challenging for third party vendors. The functions require a deep understanding of the product, which is usually preceded by an intensive training. The solution to such a scenario lies in the creation of a software development kit (SDK).

The primary purpose of this thesis is to create mobile SDKs, which exempt the core functions of YSoft SafeQ and enable the development of customizable mobile applications without the need for comprehensive expertise in the YSoft SafeQ product. The targeted platforms are Android and iOS.

The theoretical part of the thesis is divided into three parts. The first part contains the analysis of mobile SDKs produced by companies such as Google and Apple. The emphasis is put on methods that are undertaken by business-managed communities for SDK improvement and extension as well as their approach to handling the user experience of mobile SDKs.

Two mobile SDKs (one for the Android platform and one for the iOS platform) were developed as a form of validation of the research. The iOS SDK was developed using XCode and Swift programming language, and Android SDK was created in Android Studio using Kotlin programming language. The second chapter contains a description of the SDKs implementation details. It also serves as a development guide for the employment of mobile SDKs.

The third part includes validation of the functionality of mobile SDKs. For this purpose, Android and iOS mobile applications were developed. The accentuation is put on the user-generated feedback gathered by the distribution platforms. The last part of the validation contains an evaluation from software engineers who used the mobile SDK.

At the end of this thesis, the appendices are included. They contain sequence diagrams to help understand the implementation details of mobile SDKs.

As the outcome of this thesis, two fully functional mobile SDKs for Android and iOS platforms were developed in the practical part. They are delivered via two sample mobile applications demonstrating the use of mobile SDKs in the development of native mobile applications. The source code of these mobile applications is accessible via a public GitHub repository.

2 Research about mobile SDKs

In this chapter, the focus is put on the comparison of different SDK development approaches, specifically the ones used by Google and Apple.

As the outcome of this thesis, two mobile SDKs are built, which intend to enable the creation of mobile applications native to iOS and Android. They are built on top of SDKs produced by Google and Apple, respectively. These SDKs serve as a model for building the new mobile SDK. It is crucial to examine different methods employed by the companies for SDK development as the development and delivery process requires some extent of regulations for the contribution to the mobile application ecosystem to be adequate.

2.1 Methodology of research

At first, the concept of mobile SDKs is introduced. Then, the proposed approach for comparative analysis of Google and Apple SDK development is based on the following aspects:

- Community management
- Application Fundamentals
- Guidelines for the usage of the SDKs of principles utilized in design and development

The comparison between the methods used by individual companies is conducted on the different techniques utilized for SDK extension and improvement by managing the developer communities.

The fundamentals of platform-specific mobile applications are approximated to help with a better understanding of the development guidelines in a broader context as they target the unique mobile application components.

The findings on guidelines are utilized in the development of new SDKs. Also, the guidelines serve as an inspiration for creating guidelines for the new SDKs.

The results from the research are summarized at the end of this chapter in a comparative table.

2.2 Software development kit

A software development kit(SDK) is a *"set of development utilities for writing software applications."* [2] It incorporates not only the source code but also a collection of libraries, documentation, development guides, and sample code.

Usage of mobile SDKs brings many benefits from the malleability of the functionality to easier maintenance of the application. Their significance is recognized by some of the leading companies for software development.

2.3 Google's SDK

The most popular developer product conceived by Google is inarguably Android Open Source Project (AOSP) consisting of the Android operating system (OS), currently equipped by the majority of all smartphones [3].

An essential component of AOSP is the Android SDK. Crucial for developers, Android SDK renders a collection of libraries and development tools utilized in the production of mobile applications for the Android platform. The role of supplied tools lies in building, running, and testing the Android applications [4, p. 9].

Android is an open-source software stack [5, p. 88], designed so that the contributions are not restricted or controlled by a single central entity. Instead, the product is following a collaborative approach when discussing development. The majority of the Android platform and its documentation, is licensed under the Apache License, Version 2.0 [1].

2.3.1 Developer community management

After being on the market for ten years, the number of active Android devices has risen to a remarkable 2.5 billion[6, at 00:55:52] [2], and the demand for developers in this sphere is ever-increasing. Therefore, it is of great importance to operate such a community in a sophisticated way

1. The license enables the distribution and modification of free software under certain conditions, protecting the owner of the source code.
2. This information comes from a Google I/O conference convened in 2019 as the conference in 2020 was cancelled due to the Covid-19 outbreak.

to enable sharing new information and innovative remarks. Google presents numerous means of developer community management, such as actively contributing to social media and offering programs available all around the world. As a beneficial impact, a person can meet other developers of different experiences and thus gain expertise in new technologies, improve understanding of the previously known ones, and hand over their skills to give back to the community.

Google Developer Groups
There are currently 986 chapters [3] accessible all around the world. The primary role is occupied by the leadership team whose role is to organize events for developers to participate in. They include various talks, meetups, workshops, conferences, and even hackathons. The most significant event organized by such groups is the DevFest, covering multi-product topics (including Android) in a span of a few days by speakers not only from Google but from different companies as well. Anyone can join their local Google Developer Group by going through simple registration steps. If an individual is identified as an expert in any of Google technologies, the upgrade Google Experts Program should be considered. The participation in such a program is evaluated based on requirements testing the competence of submission. An Expert gains access to elite utilities, including invitations to Expert Summits, access to the not-publicly-available communities, and many others.

Developer Student Clubs
Similarly to the Google Developer Groups, students of all programs can join Developer Student Clubs. They provide the benefit of meeting peers with the same interest in technologies and learning together by sharing knowledge during workshops or while creating solutions on a local scope and cooperating on projects.

Conferences
Android conferences are being held several times a month worldwide.

3. Full list of Google Developer Groups is accessed via `https://developers.google.com/community/gdg/groups`.

The full list can be found on the official Android Study Group GitHub, in the conference repository [4].

The largest one convened is the Android Dev Summit, which lasts two days and is being hosted yearly. The event's primary focus is on the latest innovations regarding Android development. The included talks are carried out in an interactive way denoting that the official Android engineering team reacts to the feedback and ideas coming from the attendees. Attending the conference is free of charge [7]; however, a person must be selected by organizers after submitting a registration form. If a person manifests the eligibility to attend, an invitation, including the tickets, is delivered. In an instance, a person is incapable of participating in the conference, a live stream of the sessions is present as well as their recording on the official Android Developer YouTube channel.

Social media management
Google acknowledges the significance of social media management by creating developer accounts on various platforms [5], allowing interactive collaboration, and mitigating the process of resolving issues or designing solutions.

Giving back to the community
When working in a collaborative environment, giving back to the community must be admitted as a substantial practice. On the one side, such members may help with the learning process of others by actively participating in organizing meetups while being susceptible to presenting their ideas and experiences at such events. On another note, working on open-source projects is extremely valuable as well. Android developers may adopt different approaches of contribution to AOSP.

- **Bug Report**
  The process of bug reporting is executed via Google Issue Tracker tool. First and foremost, if a bug is discovered, it must be searched for in the Issue Tracker to avoid potential duplica-

4. Link to the Android Study Group GitHub - conference repository: `https://androidstudygroup.github.io/conferences/`.
5. Twitter, Facebook, Instagram, YouTube, ...

tion. Secondly, the correct category of components related to the case is selected, followed by filling in the provided template. It is very beneficial if the bug report includes as many details[6] as possible, accelerating the fixing process [8].

- **Code Contribution**
  The majority of source code of AOSP is written in Java programming language; thus, the code contributors must follow AOSP Java code style [7] for the aim of code acceptance. Some parts are written in Kotlin programming language, so the use of Kotlin is permitted only in such sections. The submitted patch is reviewed by Gerrit tool. The detailed guide on patch submission is available on the official Android website [9].

2.4 Developer's guide to Android SDK

The Android SDK comprises all of the necessary development tools for application development regarding the Android platform. The SDK is included in Android Studio, which is an integrated development environment designed for Android development [10, sec. 2].

A conventional way of publishing an Android application is via the Google Play Store. First and foremost, using an existing Google account, the registration for the Google Play Developer account needs to be undergone. Only developers age 18 and higher are authorized to do so. After examining and accepting the Developer Distribution Agreement, the registration fee of 25$, which is a one-time contribution, shall be paid [11]. The publishing is executed via Google Play Console.

Google provides a set of guidelines to make the process of developing high-quality mobile applications with the Android SDK more comfortable and more convenient. The developer ought to follow them so that the mobile application can meet the standards as they ensure reliable performance on Android devices.

6. Description of reproducing steps, log files, patch, ...
7. General information on the AOSP Java code style may be found on this link: `https://source.android.com/setup/contribute/code-style`.

2.4.1 Application fundamentals

Android applications are constructed using components, such as activities, services, broadcast receivers, and content providers [12]. Application components are declared in the manifest file. The Android OS uses the manifest file as guidance on how to integrate the mobile application to the actual device, to derive custom workflow, and guarantee consistent user experience. Additional responsibilities declared in the manifest file include [13]:

- features regarding used hardware as well as software components by the application,
- identification of required permissions,
- minimal supported API level,
- API libraries (if the library provider is different from Android).

Apart from code relating to the back-end of the mobile application, Android applications contain files that are gathered in the app resources folder. These files represent the visual assets of an application, and they include XML files used for defining the layout of each activity, colors, animations, images as well as audio and string definition files [14].

App resources are an essential component in development, ensuring application compatibility across devices with different configurations, such as various screen sizes, dimensions, language preferences, and even screen orientation. Android provides a variety of qualifiers [14], which are short strings that simplify the development process. When creating a new resource file, the qualifier is used directly in the name of the file; therefore, the figures can be automatically applied to the desired configuration by the system.

2.4.2 Architectural principles

The individual application components, presented at the beginning of Subsection 2.4.1, are a part of an environment in which they can be destroyed by the OS because of low memory at any moment. Therefore, these components should not be reliant upon each other, and no data should be stored in them.

The separation of concerns principle is followed during the development [15]. For instance, the fundamental task of the activity component is to handle UI and the interactions with the operating system. It is advised to isolate the logic of the core app functionality in classes, to make the application not only manageable but also to minimize problems dealing with the life cycle of the Android application [15].

An example model of a use case addressing user profile management, found in the official Android developer documentation [16], is used for demonstration of an entire process of architectural structuralization suggested for Android development.

2.4.3 Development guidelines

Development guidelines portray specific criteria grouped by the area of concern, and they are further subdivided into several sections, which then partition into guidelines instances, identifiable by a unique ID. Each of these sections is supplemented by test procedures that describe rigorous steps to test the feature. The described criteria should serve as a checklist before the upload of the application to the Google Play Store.

Visual Design and User Interaction guidelines
Google provides a documentation [17], which identifies the main areas that should be taken into consideration concerning the user interface. They cover standard design, navigation, and notifications.

Android developers use Material Design, which is an open-source design system consisting of components that enforce solid development workflow by unifying them under a thorough set of standards. For the convenience of Android developers, the library Material Components for Android is provided with ready-to-use implementations of designed elements.

Compatibility, Performance, and Stability guidelines
The vast number of device models in combination with different versions of the Android OS makes the application susceptible to compatibility issues. If the startup is slow or a response to user interaction takes too long, it conduces to miserable user experience. The same outcome is witnessed when dealing with an unstable product. Specific areas, namely stability, performance, SDK, battery, media, and

visual quality, are included in the documentation [17] to eliminate such issues when complying with criteria.

Functionality guidelines
The proper functional performance of the application deals with the need to fulfill necessary criteria in areas such as audio, UI, and graphics and user/app state [17]. Permissions also play a vital role in ensuring access to intended capabilities. Android Applications should utilize the principle of least privilege. Only the minimal permissions should be requested from the users, the least needed to address the core functionality of the application.

Security guidelines
In addition to data protection laws, Google states a list of criteria [17], which must not be omitted to guarantee users' privacy. Furthermore, the obligation applies to the User Data policy [8], created by Google's Developer Policy Center, as well.

Transparency is a crucial concept to consider when handling sensitive data. The user should always be aware of the usage and collection of personal data. Such disclosure must be clearly stated not only in the Google Play store listing but within the application as well.

2.5 Apple's SDK

The original idea of Apple Inc.'s founder Steve Jobs was to forbid the creation of native iOS applications by third-party developers. Instead, they were expected to create web applications preventing the need for SDK [18]. After the uttered dissatisfaction coming from the outside programmers, the company's reaction was the creation of iOS SDK in 2008, deviating from the initial concept.

iOS SDK is considered a Closed Source software, which implies that developers are not able to modify the source code related to the OS itself in contrary to Android developers. As a consequence, such an approach leads to a lack of community support, which is generally perceived as a disadvantage [19]. However, the closed model

8. The document on Google User Data policy is available at `https://play.google.com/intl/en_us/about/privacy-security-deception/user-data/`.

has benefits that are more substantial to the Apple company. The customers of Apple are less perplexed, with the reason being that there exist unified versions of the OS, securing comparable experience for every user [19].

2.5.1 Developer community management

It must be taken into consideration that a product whose development is not dependent upon direct code contribution, by the developers outside the private spectrum, does not exclude the potential to integrate ideas from them. Hence, handling the third-party developer community in a sophisticated way is regarded as vital and should not be underestimated.

User groups
Becoming a member of user group carries several benefits regarding the improvement of one's developer skills. Reaching out to those with similar interests and sharing acquired knowledge has proven to be a credible technique. Apple brings its developers together by presenting a decentralized system of groups, members of which take the initiative and organize various events and meetups designated to distribute novelties on Apple's technologies. With hundreds of groups worldwide, any developer may find the closest group and start interacting with its members by entering the preferred location into a form found on the official Apple User Group Resources website [9].

Developer forums
For the developers who prefer online communication over personal interaction, the Apple's developer forums serve as a suitable place to submit inquiries and receive feedback from not only other developers but Apple engineers as well. The forums are reasonably organized into categories concerning main Apple's developer technologies with subcategories handling each proposition respectively.

9. The official website of Apple User Group Resources - `https://appleusergroupresources.com/find-a-group/`.

Feedback submission

For Apple company, observations from users hold serious value. Therefore a distinctive team has been assigned for the sole purpose of their resolution. A user can reach a member of Apple's support team by dialing one of the Apple support numbers, chat online, submit feedback via Feedback Assistant, or make an appointment with a Genius [20].

Feedback Assistant is a native application targeting developers, bringing a more convenient way of reporting bugs to the Apple company regarding their products. Developers are presented with enhancements such as automatic on-device diagnostics, remote bug filing, the possibility to describe a bug in more detail as well as the ability to recognize more bug statuses [10]. For the reason that the application is limited for users with iOS 12.4 or later, a web-based feedback assistant may be used alternatively if necessary. This service is primarily introduced for members of the Apple Developer Program. For members of Apple Beta Software Program [11] there is a possibility to provide feedback on pre-release Apple software using the same platform. Such membership is free of charge [21] after enrolling the desired device and accepting the Apple Beta Software Program Agreement during the sign-up process.

Conferences

There are dozens of events and conferences targeting the development for iOS worldwide, delivering valuable information from experts in the field.

Apple Worldwide Developers Conference (WWDC) is the most extensive official conference organized by Apple held annually in San Jose. The event serves as an exhibition of the newest technologies unveiled by official Apple engineers, focusing on the software from the developer's point of view. Attendees must undergo a registration process. The registration is subject to acceptance of all terms and conditions of WWDC Registration and Attendance Policy. The visitors are obligated to be members of the Apple Developer Enterprise Program as well as purchase a ticket to WWDC sold for 1599$. In return, the

10. More information on bug reporting may be found at `https://developer.apple.com/bug-reporting/`.
11. Apple Beta Software Program is one of the memberships offered by Apple Inc.

developers will be granted access to different lectures, sessions, and hands-on labs held by professionals for five days [22]. Those who are unable to attend the conference in person, live streaming sessions will be provided online every day of the conference with supplied recordings accessible by the public.

2.6 Developer's guide to iOS SDK

The iOS SDK is used for the development of mobile applications for iPhone, iPad and iPod devices. The XCode, which is an integrated development environment produced by Apple, includes the iOS SDK implicitly when installed. XCode is free of charge; however, the developer is obligated to possess hardware with Mac OS [23].

The application design and development are carried out with aim of application distribution to the Apple App Store. iOS developers must enroll in the Apple Developer Program by registering via Apple ID on the official Apple developer website and paying a charge of 99$ per year. A member of the program gains access to application distribution over the App Store, accesses Apple services arising the value of the mobile application and receives professional advice on any issue regarding Apple development from a specialist [24]. If the development of the application is not finished yet (e.g., trial version, demos, betas), the TestFlight is recommended as a platform for distribution.

Apple maintains significantly higher standards of supplied applications than Google by introducing a comprehensive review process for the App Store admission. The App Store approval or rejection is a partially automated process and is always concluded by a human being. App Store Review Guidelines (2.6.2) and Human Interface Guidelines (2.6.3) are two primary documents according to which it is advised to design and develop mobile applications with the iOS SDK.

2.6.1 Application fundamentals

AppDelegate and SceneDelegate

AppDelegate and SceneDelegate files define protocols that contain definitions of methods that deal with the critical situations in the application life cycle, giving the developer competence to react to them.

Information property list
Information Property List, or Info.plist for short, is an XML file, which contains a dictionary with keys and values which are used by the system to define the the configuration of the mobile application [25].

Storyboards
The storyboards are files used for declaration of visual aspects of the application. There are usually two storyboard files created when a new project is set up. One is used for the definition of the user interface of screens, and the other one is dedicated to the launch screen. The launch screen can be created using the assets folder; nonetheless it is advised to use the storyboard method because it assures broader compatibility, which cannot be achieved by the use of a static image [26]. The use of storyboards is declared in the Info.plist file.

Assets folder
Folder with the *.xcassets* suffix is used for the definition of the graphical content of the mobile application. A developer can create sets of colors, images, app icons, launch images, and many more. The assets are very intuitive to use and create; for instance, the App Icons set is filled with application icons compatible with various screen dimensions using the drag and drop method.

2.6.2 App Store Review Guidelines

App Store Review Guidelines [12] is a document that is used not only as a set of development guidelines for iOS developers but is availed by Apple reviewers as a checklist of required criteria for App Store approval or rejection.

The main areas of concern are safety and performance, followed by business, design, and legal. These areas are further divided into

12. Full document may be accessed via `https://developer.apple.com/app-store/review/guidelines/`.

sections. Each of these sections is examined in detail by Apple reviewers.

The safety of an iOS application is subjected to the Apple Developer Program License Agreement. This PDF document is available to all iOS developers subscribed to the Apple Developer Program. Intentional violation of any agreement point classified as fraudulent action may lead to the removal from the Apple Developer Program of an individual [27].

2.6.3 Human Interface Guidelines

Human Interface Guidelines [13] are a part of Apple developer's documentation, emphasizing native application design for the iOS platform, as well as macOS, watchOS, and tvOS. Three main pillars around which the applications are built are [28]:

- **clarity** in the sense of using unambiguous graphical assets with suitable size and sharpness,
- **deference** to the consumer, excluding his over-saturation employing an understandable interface,
- **depth** of provided functionality, making sure that the user is aware of the context.

Human Interface Guidelines documentation offers an overview of recommended design principles, such as aesthetic integrity, standardized development paradigms of the iOS OS, direct manipulation, the responsiveness of the user interface, etc [28]. The documentation further defines the guidelines for application architecture, user interaction, system capabilities, visual design, and recommendations for interface essentials [28].

The core elements of every iOS mobile applications are bars, views, and controls. The views are components displayed to the user. Bars inform the user about the navigation amongst the views, and they may contain action-starting items such as buttons, switches, text fields, and many more. These items fall into the category of controls, and they not only initiate various operations, but they transmit the information as well [29].

13. The documentation website is accessed via the following link: `https://developer.apple.com/design/human-interface-guidelines/`.

When it comes to the application architecture, the Model View Controller design pattern is accentuated by Apple, assuming that the UIKit is used in the development process [14]. Assigning one of three roles to objects which form an application leads to robust and, at the same time, easily extensible infrastructure. The purpose of a model is assigned to objects defining the computational logic of the application [30, p. 1174]. The view role is occupied by the objects mediating the user interface, and the controller secures communication between the two [30, p. 1174].

2.7 Comparison of Android and iOS SDKs

The table 2.1 depicts the summary of research on mobile SDKs provided by Google and Apple, explicitly the distinction of Android and iOS SDKs.

Criterion	Android SDK	iOS SDK
Software type	open source	closed source
Developer community	organized - developer groups, conferences, code contribution, reporting	organized - user groups, developer forums, conferences, feedback submission
Community contribution	code contribution, bug reporting	members of Apple Developer's Program can submit feedback
Guides	Core App Quality, Material Design	App Store Review Guidelines, Human Interface Guidelines

Table 2.1: Differences between Android and iOS SDKs

The importance of community management, relating to the business managed SDK, has proven to be significant thanks to this research. Both companies recognize their high value. Since the Android source code is open source, Google places more considerable significance on

14. iOS developers use either UIKit or the SwiftUI for user interface development.

the management of the developer community because the developers contribute directly to the system with code.

The support of development on top of SDK should be provided not only by the SDK development team but by the developers using the SDK as well. The programmers may encounter pitfalls at some point during the implementation process. Other members of the community might offer them their solution if they came across the same or similar issues, resolving them more quickly. They may also deliver a vision of features that upgrade the SDK, creating a higher value of the product as a result.

The research on development guidelines has shown that the SDK should be supplied with sufficient documentation as well as issued usage guidelines. Apple introduces a more complex review process for the App Store approval; thus, the regulations for the iOS developers are more elaborated than the ones provided by Google.

The purpose of SDK is to simplify the development process, and it may be achieved by delivering the SDK with a detailed manual on its usage and adoption. This step is as important as the development itself. From the perspective of the developer of SDK, comprehensive documentation helps with keeping a consistent awareness of the implementation amongst the team members.

3 YSoft SafeQ mobile SDK

In this part of the thesis, the YSoft SafeQ product is introduced, mentioning its purpose and explaining the underlying architecture used for mobile printing. Then, the functional, as well as delivery requirements for the mobile SDKs, are defined. After that, the description of the implementation details of mobile SDKs follows. The general workflow of individual features, namely discovery, login, and upload, is identified as well as platform-specific implementation approaches. It is succeeded by the guidelines for developers to use the mobile SDKs in mobile application development.

3.1 Architecture of YSoft SafeQ in mobile printing

YSoft SafeQ is a solution that presents numerous functions related to the management of document printing and scanning. Out of these, mobile print management is accentuated as the core element of the mobile SDKs. The print jobs are delivered from mobile devices to the YSoft SafeQ via one of two channels - end user interface (EUI) or Mobile Integration Gateway (MIG). The uploaded jobs are stored in secure intermediate storage, using server-based spooling until they are ready for release at the designated printer.

The spooling server operates over the network and is capable of receiving, storing, and later forwarding the desired print jobs to the printing device [31]. At first, the print job is received by the spooling server, followed by its placement into the memory. The print jobs are released by authenticating at a single function or multifunction printer through a terminal with YSoft SafeQ installed. When a spooling server receives a request from YSoft SafeQ, the print jobs are transmitted to the desired printer [31].

The described workflow for print job delivery is demonstrated in Figures 3.1 and 3.2.

3.1.1 Print job delivery via end user interface

Definition: *End user interface (EUI) is a web interface allowing users to manage their YSoft SafeQ accounts.*

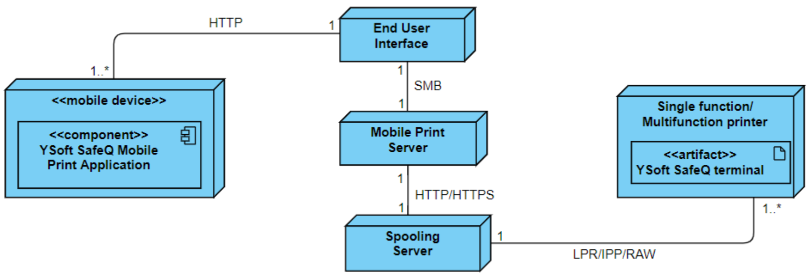

HTTP
1
1..*
End User Interface
1
SMB
1
<<mobile device>>
<<component>> YSoft SafeQ Mobile Print Application
Mobile Print Server
Single function/ Multifunction printer
<<artifact>> YSoft SafeQ terminal
1
HTTP/HTTPS
1
1..*
Spooling Server
1
LPR/IPP/RAW

Figure 3.1: The EUI job upload workflow

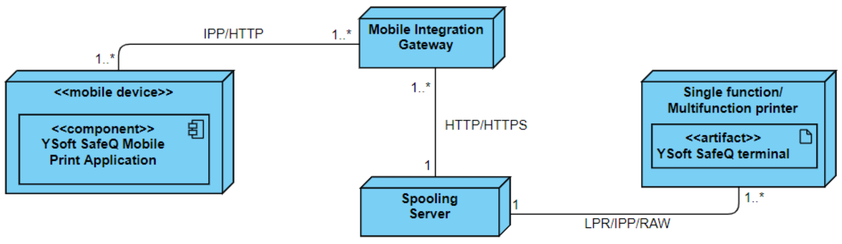

IPP/HTTP
1..*
1..*
Mobile Integration Gateway
<<mobile device>>
<<component>> YSoft SafeQ Mobile Print Application
1..*
HTTP/HTTPS
Single function/ Multifunction printer
<<artifact>> YSoft SafeQ terminal
1
Spooling Server
1
1..*
LPR/IPP/RAW

Figure 3.2: The MIG job upload workflow

From the architectural point of view, a significant component of EUI delivery via YSoft SafeQ shall be Mobile Print Server (MPS). YSoft SafeQ MPS allows users to upload documents from mobile devices such as smartphones and tablets.

The print jobs are first uploaded to the EUI, then converted to PDF file format via drivers present in MPS and consecutively stored in the spooling server.

3.1.2 Print job delivery via Mobile Integration Gateway

Definition: *Internet Printing Protocol (IPP) is a printing protocol that uses HTTP protocol as a means of communication over the default IPP port, which is the Well Known Port 631. Every HTTP operation is a post method, with the message's "Content-Type" set to "application/ipp" [32, p.18].*

Definition: *Mobile Integration Gateway (MIG) serves as a print job gateway for any authenticated IPP source. MIG also offers support for printing from iOS devices through the Apple AirPrint and from Android devices via Mopria.*

The print jobs can be delivered securely to the YSoft SafeQ from any IPP compatible device via IPP protocol using MIG. As shown in Figure 3.2, there is no need for MPS in the MIG workflow. MIG workflow features simpler architecture in view of the fact that MIG embodies the PDF converter implicitly. The print jobs are processed by print drivers and can be directly stored in the spooling server.

MIG also offers support for printing from iOS devices through the Apple AirPrint and from Android devices via Mopria. Such feature appears as an advantage, but there are some notable arguments against using them with YSoft SafeQ:

- inability to authenticate via OAuth [1],
- inability to use client certificates,
- no support for server discovery differently than by mDNS protocol,
- no support for Mobile Device Management configuration.

3.1.3 Mobile Print Server vs Mobile Integration Gateway

The difference between MPS and MIG is best depicted on an example scenario that involves mobile printing from multiple locations, consider numerous continents. If the job upload via EUI was chosen, then the MPS must be installed in only one of the sites. The MPS is allowed to have only one configuration, hence installing more MPSs

1. OAuth is an authorization standard without using login tokens instead of credentials to avoid sharing sensitive information about users.

would prove to be pointless because more MPSs would only result in replicating the same server. Having only one MPS for multiple locations causes the overload of the bandwidth, slowing down the upload process. On the other hand, MIG solves this issue because each site may deploy MIG with custom configuration.

3.2 Functional requirements for SDKs

The mobile SDKs must render the following features:

- authentication of the user against the YSoft SafeQ server,
- the upload of print jobs to the YSoft SafeQ server via end user interface and Mobile Integration Gateway,
- discovery of server with YSoft SafeQ installed,
- enable the development of native mobile applications for Android and iOS platform.

3.3 The mobile SDKs delivery requirements

The mobile SDKs shall:

- include a detailed documentation on how to integrate the mobile SDK into the project
- be delivered in a form of code snippet via an open-source GitHub repositories as a part of sample mobile applications
- be designed and developed in accordance with the development guidelines for Android and iOS platforms respectively

3.4 Service discovery implementation

The functionality of discovery lies in discovering the URL of the available YSoft SafeQ print server. It deprives the end-user of the mobile application of remembering and entering the complex URL of the server, which is an essential prerequisite for the login to the YSoft SafeQ and subsequent upload of the print jobs.

The MIG is available on port 8050, and the EUI is accessed via port 9443 by default. The algorithm for discovery is designed around the mentioned acknowledgment.

The general workflow is portrayed in Figure A.1. The initial step is to check the value of the entered URL by the end-user, whether it contains *safeq6* substring. If it does not, the entered string is handled as a domain. The *safeq6* substring is, in this case, prepended to the entered URL. The newly created URL is then appended by potential ports (8050 or 9443) at which print servers may be discovered. The new URLs are sequentially checked using a get request until an affirming response of *200 OK* is received. The user is then prompted for the confirmation to use the discovered URL of the server. Otherwise, if the discovering fails to locate the server, the end-user is advised to check the internet connection and try again later.

The

3.4.1 Local printers discovery

The discovery is supplied with the algorithm for locating the printers on the user's local network.

3.4.2 Implementation of local printers discovery in Android

The local printer's discovery in Android implementation is secured by the Java implementation of the multicast DNS library, available through the public GitHub repository.

3.4.3 Implementation of local printers discovery in iOS

A multicast DNS protocol was used for the implementation of discovery functionality, taking inspiration from Apple's Bonjour. The *Discovery* class adopts *NetServiceDelegate* and *NetServiceBrowserDelegate* protocols. The optional methods predefined by these delegates then enable to utilize the functionality of multicast DNS protocol.

3.5 EUI delivery implementation

3.5.1 EUI login and upload general workflow

The workflow of the user's login to the YSoft SafeQ and subsequent upload of the selected print jobs to the EUI is depicted in Figure A.2.

The user provides valid credentials through the login screen and the URL of MPS, either manually or using the discovery function. The get request is then sent to the server represented by the entered URL via HTTPS protocol and obtaining login token by matching *"_csrf\ ".*"* pattern with the request-response. Next, the post request with data such as username, password, and the acquired login token is sent to the same server for credential verification. The upload process is established after the user selects print jobs by sending a get request and determining upload token from the request-response in the same manner as the login token. The binary version of each selected print job is used to build the post request sent to the MPS.

3.5.2 Implementation of EUI workflow in Android

For building requests in the implementation of SDK for Android, an *OkHttp* client was used, instead of the standard HTTP client. *OKHttp* is a third party library, its employment is efficient [2], which saves bandwidth and increases the speed of the request [33].

A persistent cookie jar is added to the client for managing and persisting obtained cookies from get request responses while storing them in the Shared Preferences [3]. The implementation uses a *PersistentCookieJar*, which is a third-party library that uses Shared Preferences for data persistence.

Y Soft issues a self-signed certificate, and given that it is not published by a trusted certificate authority, the certificate is not implicitly trusted by a client. With the intention of handling certificate validation,

2. Efficiency in the sense of diminution of repeated requests due to response caching, connection pooling, use of the GZIP download format,... The *OKHttp* library secures more effective access to the data as well as its writing by creating a shared memory [33].
3. Shared Preferences is a specific file, in which values may be stored under a specific key and later obtained. This file is accessible throughout the entire Android application.

the client is built using a custom class called *CustomTrust* declared in the SDK.

3.5.3 Implementation of EUI workflow in iOS

The network tasks are built using *URLSession* class native to Swift, to satisfy the requirements of login and job upload workflow as the transfer of data over the network occurs. The request is first built using a Swift *URLRequest* class.

As mentioned in Subsection 3.5.2, the MPS uses a self-signed certificate. The authentication of server is handled manually by overriding the *urlSession* method, defined by *URLSessionDelegate* and responding to the *NSURLAuthenticationMethodServerTrust* authentication method.

3.6 MIG delivery implementation

3.6.1 MIG login and upload general workflow

The visual representation of the general workflow of login and upload via MIG is shown in Figure A.3.

The post requests of MIG workflow are based on IPP protocol. Two main divergences from the EUI workflow reside in the login token acquisition and the upload process. First of all, the get request is sent to the MIG. The token, used for both login and upload, is generated from the entered username and password and not from the get-request response. Then, after the successful post request using the resource path of the server URL with a suffix of *"/ipp/print"*, the login process is finalized. The upload part requires only one post request for each selected file to be uploaded.

3.6.2 Implementation of MIG workflow in Android

The implementation uses two classes for building requests: *IppEmptyRequest* and *IppRequest*. They were both implemented by a Y Soft employee as a part of his Master's Thesis on protocols for printing from mobile devices [34].

3.6.3 Implementation of MIG workflow in iOS

The implementation details on used classes are coincident to the ones described in Section 3.5.3, with the requests built in a different manner.

3.7 Developer's guide for Android SDK

3.7.1 Sample application

A sample mobile application, made available to the public via the GitHub, delivers the Android SDK in the *sdk* folder. The application may be started by cloning or downloading the repository and following the instructions in the *README.txt* file.

3.7.2 Implementation

The mobile SDK was developed in Android Studio IDE, using Kotlin programming language. The SDK includes the functionalities separated into classes, namely *Discovery*, *Login* and *Upload* class. The functionality offered by these individual classes is accessed by initializing them according to the rules described in the following subsections.

3.7.3 Project setup

The core functionality is based on sending requests to the servers, and thus specific permission must be added to the application manifest file:

```
<uses-permission android:name="android.permission.
    INTERNET"/>
```

Then, the developer must integrate the *OKHttp* library as well as the *PersistentCookieJar* (both available publically on GitHub) into the project.

3.7.4 Discovery

The initialization of *Discovery* class is required. The first parameter of a constructor for this class is the scope of the *Activity* from which this class is instantiated. The *Activity* must implement the *DiscoveryCallback* iterface declared in the *Discovery* class.

The interface of *DiscoveryCallback* includes three methods which are mandatory to implement:

- *showDialog(title: String, message: String)*
  The purpose of this method is to inform the end-user of the mobile application. The suggested way is to present a dialog with the title and message present in the method's parameters.
- *promptUserForUrlConfirmation(url: HttpUrl)*
  If a server discovery succeeds, this method is called to appeal to the end-user for confirmation or the denial of the identified server URL.
- *hideDiscoveringBtn(flag: Boolean)*
  The interaction of end-user with the screen is disabled in this method.

The second parameter in the constructor is the name of the server, entered by the user through the input text field on the login screen.
Basic discovery algorithm
The discovery is commenced by calling a *discover()* method on the instance of *Discovery* class, which handles the entire discovery process.

JmDNS discovery
If a developer wishes to use the JmDNS discovery from the mobile SDK, there are some additional steps to *Discovery* class initialization, which need to be followed.

1. A *JmDNS* library is integrated into the project.
2. A permission for accessing the state of the network must be declared in the application manifest file:

   ```
   <uses-permission android:name="android.permission.
       ACCESS_NETWORK_STATE"/>
   ```

3. The variables shown in the following snippet of code are initialized, and the values are then passed to the prepared attributes

of the *Discovery* class. These parameters are used in the JmDNS algorithm implemented within the *Discovery* class. The discovery process should be initiated in the background, preferably on the application startup in the *onCreate* method of managing *Activity*.

```
var discoveryClass = Discovery(this, serverName)

val connectivityManager = applicationContext.
getSystemService(Service.CONNECTIVITY_SERVICE)
as ConnectivityManager

val wifiManager: WifiManager = applicationContext.
getSystemService(Context.WIFI_SERVICE)
as WifiManager

discoveryClass.connectivityManager = connectivity-
Manager

discoveryClass.wifiManager = wifiManager
```

Afterwards, the setup for the JmDNS discovery is complete. The only remaining thing is to call *startIppDiscovery()* method on the instantiated *Discovery* class.

3.7.5 Login

To access the login functionality offered by mobile SDK, the *Login* class must be initialized with the following parameters:

- the scope of the *Activity* (*LoginCallback*),
- the scope of the *Activity* (*Context*),
- URL of the server (*String*),
- the user's username (*String*),
- the user's password (*String*),
- a flag on saving user information (*Boolean*).

The *Activity* of origin (in which the *Login* class is instantiated), must implement the *LoginCallback* interface, which defines four methods:

- *showDialog(title: String, message: String)*
  See the *showDialog* method description in Subsection 3.7.4.

- *showLoginProgressBar(flag: Boolean)*
  The reaction of the UI to the login process is handled in this method.

- *invokeUploadActivity(token: String)*
  Starts the *Activity* managing the upload process.

- *savePreferences()*
  The implementation of saving user's credentials is located in this method.

- *clearPreferences()*
  A deletion of user's credentials occurs in this method.

The URL of the server is either entered by end-user or obtained via discovery using the *Discovery* class, username, and password are received via the login screen. The flag on saving user's information is usually acquired from the checkbox included in the login screen. Saving user credentials is an optional feature, which is not implemented in the sample application.

After successful initialization of the Login class, the *downloadLoginPage()* method is called, taking care of the entire login process. The delivery endpoint (meaning either EUI or MIG) of login is automatically determined according to the server URL.

3.7.6 Upload

The initialization of *Upload* class is conducted using following parameters:

- scope of the *Activity* (*UploadCallback*),
- URL of the server (*String*),
- an array containing paths of files to upload,
- upload token (*String*),
- delivery endpoint (*String*).

The *UploadCallback* interface must be implemented by the *Activity*, in which the upload process is handled. The interface contains three methods, which are obligatory to implement:

- *showDialog(title: String, message: String*
  See the *showDialog* method in Subsection 3.7.4
- *isUploadBeingProcessed(flag: Boolean)*
  Handles the reaction of the UI to the upload process.
- *selectBtnIsVisible(flag: Boolean)*
  Enables and disables the selection of files to be uploaded.

The URL of the server, together with the upload token, is passed from the instance of *Login* class.

The delivery endpoint is a *String* determined from the server URL during the login process.

The initialized instance of the Upload class should call the *handle-Upload()* method, handling the upload process of desired files to the chosen Y Soft's server.

3.8 Developer's guide for iOS SDK

3.8.1 Sample application

The sample mobile application for iOS devices is present in the GitHub repository. The SDK for iOS is introduced in the *YSoft SafeQ SDK* folder. The launch of the application is initiated by clicking on the *.xcodeproj* file. The guidance on how to use the mobile application is available in the *README.txt* file.

3.8.2 Implementation

The SDK was developed in the official XCode IDE, employing Swift programming language. The structure of the SDK for iOS is analogous to the SDK for Android. The functionality is redistributed amongst the *Discovery*, *Login* and *Upload* classes, which must be initialized. Each of the classes includes its protocols, which are Swift's correspondence to interfaces. The protocols ensure the communication with the instance of *UIViewController* class responsible for handling the UI.

3.8.3 Project setup

The developer must add the following key to the Info.plist file [4]:

```
<key>NSAppTransportSecurity</key>
<dict>
   <key>NSAllowsArbitraryLoads</key>
   <true/>
</dict>
```

For the reasons on Y Soft issuing self-signed certificates clarified in Subsection 3.5.2, the HTTP traffic in the requests is classified as unsecured. This key is used for enabling the HTTP traffic in the mobile SDK.

3.8.4 Discovery

The functionality of discovery is accessed from SDK by creating an instance of the *Discovery* class. Its constructor incorporates a *String* parameter denoting the name of the user-entered server name, as well as the scope of the *ViewController* implementing the *DiscoveryDelegate* protocol.

The *DiscoveryDelegate* protocol contains the definition of three methods, which must be overriden in the *ViewController* handling the discovery. The methods are:

- *notifyUser(title: String, message: String)*
  This method notifies the user of the mobile application with the given title and message by presenting an alert dialog.
- *setAllButtons(flag: Bool)* The user interaction with screen elements is enabled/disabled according to the flag present in the parameter of the method.
- *promptUserForURLConfirmation(url: String)*
  This method displays a confirmation dialog with the discovered URL of the print server with YSoft SafeQ.

4. The XML view of the file can be evoked by right clicking on the file in the project navigator, located in the left panel in the XCode, and selecting the *Open As* option followed by *Source Code*.

The last step is to evoke public method *discover()* on the instance of the *Discovery* class. The mDNS discovery is called implicitly within the *discovery* method, not requiring any additional setup.

3.8.5 Login

The *Login* class is initialized with the following parameters:

- server name (*String*),
- username (*String*),
- password (*String*),
- flag for saving credentials (*Bool*),
- the scope of the *Activity*.

The name of the server as well as the username and password, is acquired through the login screen of the mobile application.

There has been created an optional feature of saving credentials. Therefore, a flag for saving the credentials is passed amongst the parameters in the constructor.

The *ViewController* handling the login must implement the *Login-Delegate* protocol methods:

- *showLoginProgressBar(flag: Bool)*
  Reacts to the upload process by modifying the user interface and the end-user's interaction with it.

- *notifyUser(title: String, message: String*
  See the *notifyUser* method in Subsection 3.8.4

- *presentUploadStoryboard(deliveryEndpoint: String)*
  Navigates to a different *ViewController* (the one handling upload) after successful login and passes the *deliveryEndpoint* attribute to the *ViewController*. The server name as well as the token obtained from user credentials shall be transfered to the new *ViewController*.

- *savePreferences()*
  Saves the credentials to the *UserDefaults*[5].

5. UserDefaults is a database available to every user of the iOS mobile application.

- *clearPreferences()*
  Clears the saved credentials from the *UserDefaults*.

The final step to invoking a login process is calling a *handleLogin()* method on the initialized *Login* class instance.

3.8.6 Upload

The class managing the upload process is instantiated by passing a sequence of parameters to the *Upload* class:

- URL of the server (*String*),
- an array of *PrintJob* class instances,
- the delivery endpoint (*DeliveryEndpoint*),
- token (*String*),
- scope of the *Activity*.

The URL of the server is identical to the one used for login to the application.

The print jobs are obtained via the desired file picker. After selecting the files to upload, the array of *PrintJob* objects is created and passed to the constructor. The implementation of a sample mobile application does not contain the file picker because it is one of the features the partners may want to customize.

The delivery endpoint parameter is crucial for the successful upload because the *Upload* class contains the upload algorithm specific for each delivery channel. The delivery endpoint is a *DeliveryEndpoint* enum, which is present in the *YSoft SafeQ SDK* folder as well.

The Android implementation does not contain the delivery endpoint in the form of an enum. Its type is a simple *String* class. The reason behind it is that Android uses a different way of sharing objects amongst the classes. In Android implementation, the *SharedPreferences* were used. This directory enables the storage of only a limited number of supported types of objects [6], making it possible to store the delivery endpoint as *String* only.

Token is obtained from the user credentials during login and is suggested to be passed when invoking *ViewController* responsible for

6. Only primitive data types may be stored in *SharedPreferences*, such as Boolean, String, Int, Float and Long.

upload. This parameter is used in the *Upload* class for the upload via MIG.

UploadDelegate protocol contains the definitions of methods to implement:

- *notifyUser(title: String, message: String)*
  See the *notifyUser* method in Subsection 3.8.4.
- *isUploadBeingProcessed(flag: Bool)*
  Responds to the upload process by disabling the interaction with the UI while uploading the print jobs to the server.
- *selectBtnIsVisible(flag: Bool)*
  Enables the selection of files to be uploaded after the previous upload process finishes.
- *reloadTableview(printJobs: Array<PrintJob>)*
  The *UITableView* showing the selected print jobs needs to be reloaded after the upload, therefore this method should call *reloadData()* method on the instance of a tableview in the upload-managing *Activity*.

To upload the selected files, the developer calls the *handleUpload()* method on the instance of *Upload* class, after its successful initialization.

4 Validation

The idea of mobile SDKs is validated in two aspects. First, the focus is put on the mobile SDKs. Section 4.1 comprises the developer feedback of software engineers on the mobile SDKs. Then, in Section 4.2, a description of the mobile applications built on top of mobile SDKs takes place as a form of verification of the mobile SDKs applicability. Two sample mobile applications, one for each platform, were developed. The SDKs were also used for the development of the official YSoft SafeQ Mobile Print applications. Subsection 4.2.2 includes user feedback on the core functionality of the mobile application and the overall usefulness of the concept. This subsection also issues the data acquired by distribution platforms on active users of the YSoft SafeQ Mobile Print applications. The last section, Section 4.3 outlines possible ways of the future extension of the mobile SDKs.

4.1 Mobile SDKs

The mobile SDKs were developed in conformance with guidelines described in Sections 2.4 and 2.6 and at the same time they meet all of the functional requirements specified in Section 3.2.

4.1.1 Developer feedback

Two software engineers from Y Soft were asked to provide developer feedback on the mobile SDKs. The following paragraphs summarize their insights.

Respondent 1: Senior software engineer
"Overall, the concept of mobile SDK is interesting and a lot of partners would be able to integrate it into their mobile applications. However, the mobile SDKs are delivered as classes within folders, which is not efficient in the production. In my opinion, the mobile SDKs should be delivered as a library, keeping the source code private while giving the partners ability only to use the methods within the mobile SDKs. The current implementation has a potential to be used, but there are still some things (mainly missing unit tests,

delivery form) which need to be handled before publishing within the Y Soft company."

Respondent 2: Embedded systems developer
"In my opinion, the sample mobile applicatios are a nice prototypes, which show how to implement the upload of print jobs to the YSoft SafeQ from mobile devices using the mobile SDKs. The main issue of the mobile SDKs resides in missing unit tests. Their role is regarded to be crucial for retaining the functionality when updating the SDKs in the future. Another area of concern is that strings present in the code make the mobile SDKs not flexible enough because of the inability to translate them to other languages supported by YSoft SafeQ."

4.2 Mobile applications

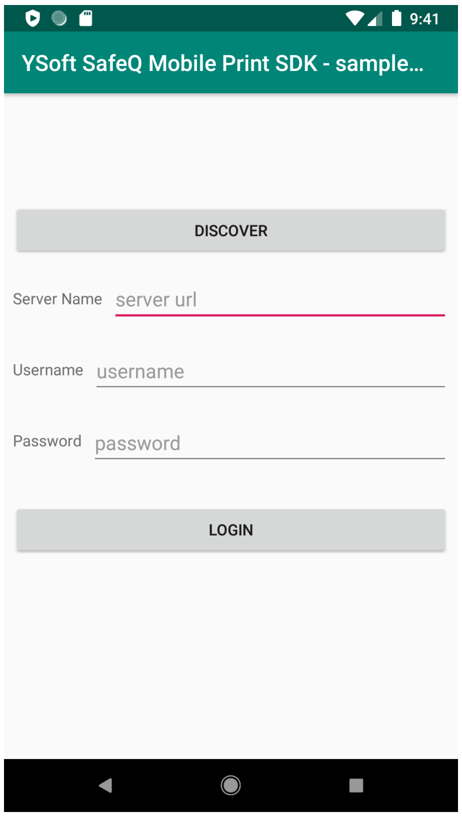
The development of mobile applications on top of the developed mobile SDKs confirms that the mobile SDKs may be used for mobile applications development and simultaneously provide the core functionality of the YSoft SafeQ product for mobile printing within the mobile applications.

4.2.1 Sample mobile applications

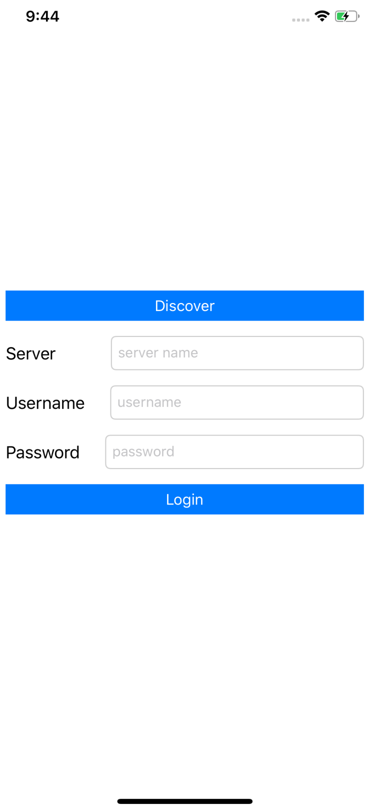
The sample mobile applications are utilized to deliver the SDKs in the designated folders and also to demonstrate the core functionality of the SDKs. A technique of minimum viable product (MVP) is employed. MVP is a method utilized in the software development process, creating a prototype product that comprises the minimum functionality of the concept [35, p.62]. The user interface, as well as a workflow of the provided screens, is described in the following paragraphs.

Login screen

The login screen includes three input text fields for the server name, username, and password of the user as depicted in Figures 4.1a and 4.1b. The credentials are coincident with the ones used to access the YSoft SafeQ product. On the login screen, there are two buttons present.



(a) Android login screen



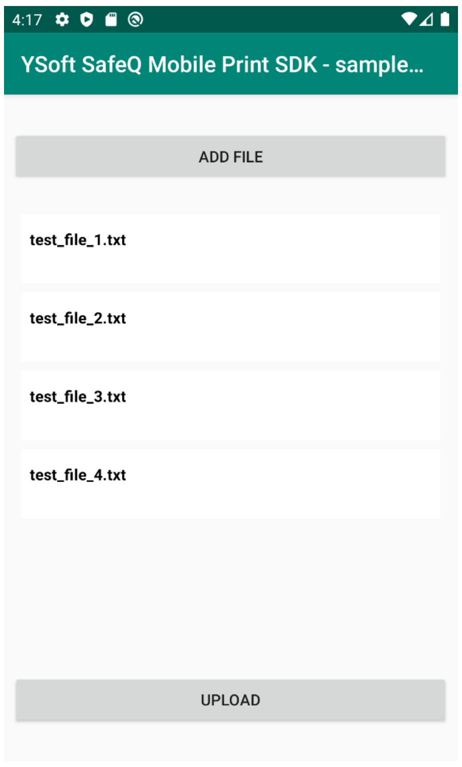
(b) iOS login screen

Figure 4.1: Login screens of sample applications

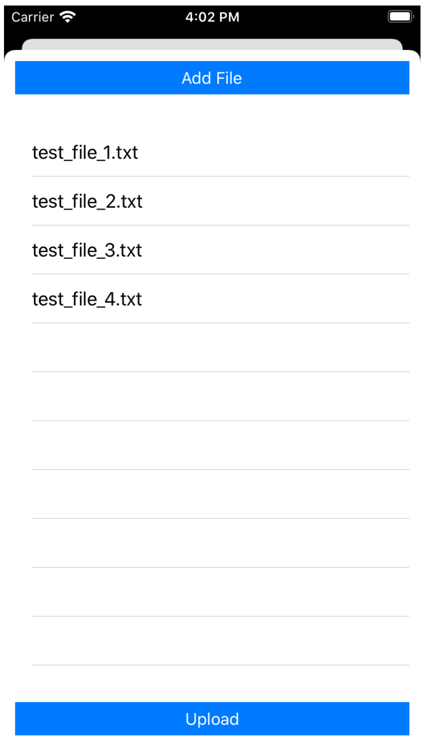
The top *Discovery* button is used for server discovery. The user must enter at least the domain of the server (e.g., ysoft.local) to the server text field for the discovery to be successful. With all of the text fields filled in, the login button may be pressed, which invokes the login process. In case the process of logging in fails, the user is informed about the particular issue that occurred.

Upload screen

The upload screens consist of two buttons and a component displaying the print jobs. The top button titled *add file* adds a generic file to the list of files to be uploaded. The SDKs do not contain file pickers because



(a) Android upload screen



(b) iOS upload screen

Figure 4.2: Upload screens of sample applications

the partners of Y Soft use their custom file pickers, and therefore this feature needs to be implemented according to the partner's own needs. The added generic files are displayed in a RecyclervView in Android (Figure 4.2a) and UITableView in iOS (Figure 4.2b) . After adding the files, the *upload* button is pressed, handling the upload process of the files to the appointed server. The user is informed about the upload by displaying a notification on the upload finish. Users may then choose to select and upload more files. The uploaded files are linked to the account under which it was logged in into the application. They can be displayed at either end user interface or any printer with the YSoft SafeQ terminal where they are also released.

4.2.2 YSoft SafeQ Mobile Print application

The development process of this application falls outside the scope of this thesis. The users of the mobile application were presented with a questionnaire on the SDKs core functionality and the overall concept. The questionnaire was distributed to the YSoft SafeQ employees, specifically to the group of 33 people interested in mobile applications, out of which 12 people were willing to participate in the survey. The received return rate reaches 36%. The answers are summarized in the following paragraphs.

Exactly half of the people participating in the survey have encountered the mobile application version for Android platform and half came across the iOS version.

All of the respondents believe that supplying YSoft SafeQ with a mobile application adds value to the product. The Figure 4.3 depicts the user's judgement on the usefulness of the application processed into a graph. The values assigned to the degree of benefits of the mobile application range from one to ten, with one being the least useful and ten being the most useful.

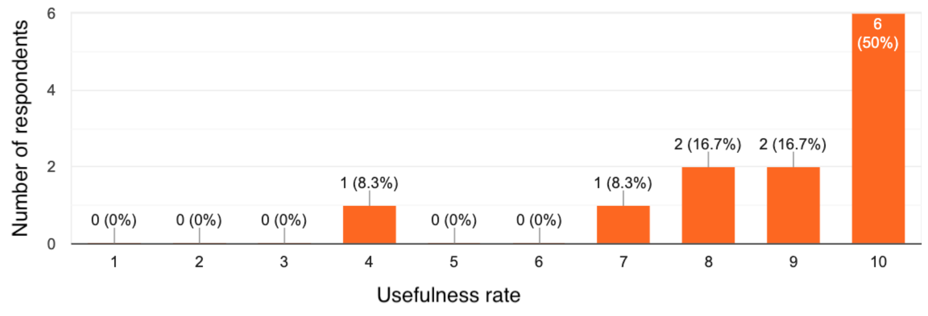


Figure 4.3: The rate of usefulness of the mobile application

The rest of the questionnaire was focused on clarification of the assigned values depicted in Figure 4.3.

The responses to the questions prove that the initial idea behind the creation of the mobile SDKs with utilized functionality has a potential to be used by the customers of Y Soft. The respondents who found the mobile application being the least useful claimed, that the mobile application is lacking some functional features. This creates an opportunity for extension of the mobile SDKs in future development. These features are described in Section 4.3 portraying possibility of additional features of the mobile SDKs.

The Figures 4.4 and 4.5 display the usage statistics of the YSoft SafeQ Mobile Print application after the successful publishing of the applications to the distribution platforms. The graphs were extracted from Google Play Console and App Store Connect, which are the platforms that were used for distribution.

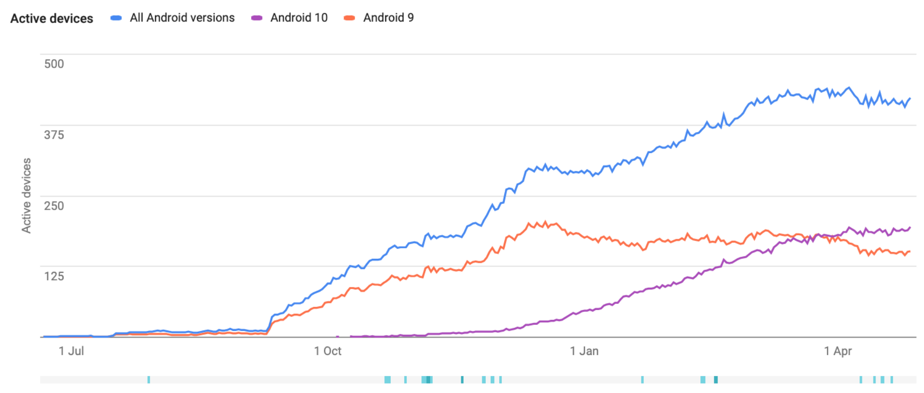


Figure 4.4: Android active users

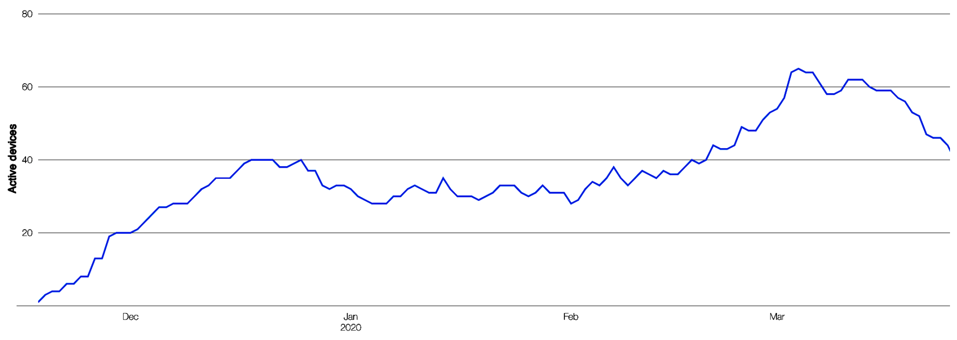


Figure 4.5: iOS active users

The statistics of usage of mobile applications varies between the Android and iOS platform. One reason is the demographics. The propagation of the mobile application focused on the Czech republic and the mobile phone users in this region use mostly devices with Android OS [36]. Another reason is the different release date. The mobile application for Android was developed several months prior to the iOS version, with no issues considering the distribution to the Google Play Store. The mobile application for iOS encountered some problems with performance, making it more difficult to pass the Apple App Store review process.

Despite the inconsistency amongst platforms, the statistics of the usage of the mobile application proves the existing demand for the mobile applications on the market.

4.3 Extension of mobile SDK

The following list incorporates the ideas for extension of the functionality of the mobile SDKs conceived not only by the respondents to the questionnaire described in Subsection 4.2.2, but they were also taken as an input from the partners of the Y Soft company:

- support for printing from home printers using Common UNIX Printing System (CUPS) delivery,
- facilitate the use of client certificates for authentication,
- employ OAuth protocol as an authentication method,
- enable configuration of certain keys[1] by mobile device management provider.

Based on the feedback from developers of the Y Soft company, the mobile SDKs will be supplied with additional unit tests.

1. The configured keys may be, for instance, a URL of the print server or a username.

5 Conclusion

At the beginning, the goal of the bachelor's thesis was defined. The goal lies in the construction of mobile SDKs for Android and iOS platforms which would enable creation of native mobile applications and link the core functionality of the YSoft SafeQ product with the mobile application.

For successful accomplishment of this objective, a certain methodology was conducted. At first, a research was carried out on companies creating SDKs. The newly created SDKs are constructed within a business managed environment of Y Soft company, therefore the first part of research focused on the importance of community management in Google and Apple companies. It has proven to be significant thanks to the research and will be taken into consideration in further maintenance of the mobile SDKs. The research then continued by studying the guides supplied by these businesses in order to ensure consistent and effortless user experience[1] of their mobile SDKs.

The next part of the thesis includes the description of implementation of the mobile SDKs for Android and iOS platforms. They were developed in conformance with the development guidelines of individual platforms. The findings of the research phase on methods of handling user experience by companies were employed in the development process of the mobile SDKs.

The concept of mobile SDKs presented to business partners for the purpose of development of mobile applications was validated by gathering the feedback from Y Soft developers as well as from mobile application users. The mobile SDKs are in the state of being evaluated by the developers of partners, using the mobile SDKs for development.

As a result of this thesis, the YSoft SafeQ mobile SDKs were developed, one for Android platform and one for iOS platform. They fulfill all of the requirements declared in Sections 3.2 and 3.3.

The current state of implementation of the mobile SDKs is sufficient enough to be released publicly to the developer community and they can use it for the development of custom mobile applications. The developed mobile SDKs will be extended by features requested

1. The user denotation indicates the developer using the mobile SDK for development purposes.

by the partners of the Y Soft company in the future and the SDKs will continue to be maintained. The vision for future mobile SDKs management is providing a collaborative environment, in which the developers will have the opportunity to integrate new features to the mobile SDKs via pull requests.

Bibliography

1. LEE, Paul; CASEY, Mark; WIGGINTON, Craig; CALUGAR-POP, Cornelia. *Deloitte's 2019 global mobile consumer survey: Tracking consumers' digital behavior around the world* [survey]. Deloitte, 2019.
2. *Sdk (Software Development Kit)* [online]. Gartner [visited on 2019-04-10]. Available from: `https://www.gartner.com/en/information-technology/glossary/sdk-software-development-kit`.
3. *Smartphone Market Share* [online]. International Data Corporation [visited on 2019-11-26]. Available from: `https://www.idc.com/promo/smartphone-market-share/os`.
4. SIMON, Jonathan. *Head First Android Development*. Sebastopol, United States: O'Reilly Media, 2012. ISBN 1-449-39330-6.
5. ESMAEEL, Hana R. Apply Android Studio (SDK) Tools. *International Journal of Advanced Research in Computer Science and Software Engineering*. 2015, vol. 5, pp. 88–93.
6. *Google Keynote (Google I/O'19)* [conference]. California, United States: Google, 2019. Available also from: `https://www.youtube.com/watch?v=lyRPyRKHO8M&t=3292s`.
7. MCQUILLAN, Sean. *Android Dev Summit 2019 Registration is Open* [online]. Google [visited on 2019-11-28]. Available from: `https://android-developers.googleblog.com/2019/07/android-dev-summit-19-registration.html`.
8. *Reporting Bugs* [online]. Google [visited on 2019-12-03]. Available from: `https://source.android.com/setup/contribute/report-bugs`.
9. *Submitting Patches: For contributors* [online]. Google [visited on 2019-12-03]. Available from: `https://source.android.com/setup/contribute/submit-patches#for-contributors`.

10. SMYTH, Neil. *Android Studio Development Essentials*. 6th. CreateSpace Independent Publishing Platform, 2015. ISBN 978-1519722089. Available also from: `http://solutionsproj.net/software/Android_studio.pdf`.
11. *How to use the Play Console* [online]. Google [visited on 2019-12-05]. Available from: `https://support.google.com/googleplay/android-developer/answer/6112435`.
12. *Application Fundamentals* [online]. Google [visited on 2019-12-05]. Available from: `https://developer.android.com/guide/components/fundamentals`.
13. *Application Fundamentals - Manifest File* [online]. Google [visited on 2019-03-30]. Available from: `https://developer.android.com/guide/components/fundamentals#Manifest`.
14. *Application Fundamentals - App resources* [online]. Google [visited on 2019-04-04]. Available from: `https://developer.android.com/guide/components/fundamentals#Resources`.
15. *Guide to app architecture - Common architectural principles* [online]. Google [visited on 2019-04-05]. Available from: `https://developer.android.com/jetpack/docs/guide#common-principles`.
16. *Recommended app architecture* [online]. Google [visited on 2019-04-04]. Available from: `https://developer.android.com/jetpack/docs/guide#recommended-app-arch`.
17. *Core app quality* [online]. Google [visited on 2019-12-05]. Available from: `https://developer.android.com/docs/quality-guidelines/core-app-quality`.
18. *Jobs' original vision for the iPhone: No third-party native apps* [online]. 9to5 [visited on 2020-01-30]. Available from: `https://ww.9to5mac.com/2011/10/21/jobs-original-vision-for-the-iphone-no-third-party-native-apps/#`.
19. MCDONALD, Tom. *Comparison Between Open Source and Closed Source Software* [online]. New England Systems, Inc. [visited on 2020-01-31]. Available from: `https://www.nsiserv.com/blog/bid/29956/comparison-between-open-source-and-closed-source-software`.

20. KURT, Serhat. *How To Send Feedback To Apple* [online]. macReports [visited on 2020-02-04]. Available from: https://macreports.com/how-to-send-feedback-to-apple/.

21. *Apple Beta Software Program - Frequently asked questions* [online]. Apple Inc. [visited on 2019-05-18]. Available from: https://beta.apple.com/sp/betaprogram/faq.

22. *WWDC19 Registration and Attendance Policy* [online]. Apple Inc. [visited on 2020-02-05]. Available from: https://developer.apple.com/wwdc19/policy/.

23. ADAMSON, C.; CLAYTON, J. *IOS 10 SDK Development: Creating IPhone and IPad Apps with Swift*. Pragmatic Bookshelf, 2017. Pragmatic programmers. ISBN 9781680502107. Available also from: https://books.google.cz/books?id=VuAkvgAACAAJ.

24. *Membership Details* [online]. Apple Inc. [visited on 2020-02-27]. Available from: https://developer.apple.com/programs/whats-included/.

25. *About Information Property List Files* [online]. Apple Inc. [visited on 2020-04-07]. Available from: https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/AboutInformationPropertyListFiles.html.

26. *Launch Screen* [online]. Apple Inc. [visited on 2020-04-07]. Available from: https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/launch-screen/.

27. *Human Interface Guidelines - Legal* [online]. Apple Inc. [visited on 2020-03-26]. Available from: https://developer.apple.com/app-store/review/guidelines/#legal.

28. *Human Interface Guidelines* [online]. Apple Inc. [visited on 2020-03-24]. Available from: https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/.

29. *Human Interface Guidelines - Interface Essentials* [online]. Apple Inc. [visited on 2020-03-25]. Available from: https://developer.apple.com/design/human-interface-guidelines/ios/overview/interface-essentials/.

30. POP, Dragos-Paul; ALTAR SAMUEL, Adam. Designing an MVC Model for Rapid Web Application Development. *Procedia Engineering*. 2014, vol. 69, pp. 1172–1179. Available from DOI: `10.1016/j.proeng.2014.03.106`.

31. KEENEY, Richard A.; LODWICK, Philip A.; SCHOENZEIT, Loren; STEINBERG, John D.; TENENBAUM, Ofer. *Spooling server apparatus and methods for receiving, storing, and forwarding a print job over a network*. US. US Patent, 7095518B1. 2006. Available also from: `https://patents.google.com/patent/US7095518B1/en`.

32. *Internet Printing Protocol/1.1: Encoding and Transport* [online]. Network Working Group [visited on 2020-02-27]. Available from: `https://tools.ietf.org/html/rfc2910`.

33. *OKHttp* [online]. Square, Inc. [visited on 2019-03-01]. Available from: `https://square.github.io/okhttp/`.

34. BARTOŇ, Petr. *Protocols for printing from mobile devices* [online]. Brno, 2016 [visited on 2020-05-08]. Available from: `https://is.muni.cz/th/wyl4x/`. Master's thesis. Masaryk university, Faculty of Informatics. Supervised by Juraj MICHÁLEK.

35. JÄRVINEN, Janne; HUOMO, Tua; MIKKONEN, Tommi; TYRVÄINEN, Pasi. From Agile Software Development to Mercury Business. In: LASSENIUS, Casper; SMOLANDER, Kari (eds.). *Software Business. Towards Continuous Value Delivery*. Cham: Springer International Publishing, 2014. ISBN 978-3-319-08738-2.

36. *Mobile Operating System Market Share Czech Republic* [online]. StatCounter [visited on 2020-04-25]. Available from: `https://gs.statcounter.com/os-market-share/mobile/czech-republic`.

A An appendix

The appendix includes Figures of workflows using sequence diagrams. These diagrams ought to serve for better understanding of the core functionality of the YSoft SafeQ Mobile Print SDKs. Each of them represents a general error-free workflow, which denotes that there is an expected status code of *200 OK* [1] to be received after each request.

A.1 Discovery workflow

The Figure A.1 depicts the basic algorithm for discovering the URL of the available print server with YSoft SafeQ installed. The user must enter the domain of the server for the discovery to succeed.

A.2 EUI workflow

The Figure A.2 serves as a supplement to the implementation part of the login and upload via EUI, for better understanding of the process.

A.3 MIG workflow

The Figure A.3 approximates the login and upload process via MIG delivery channel.

1. Response code of *200 OK* denotes the success of the request.

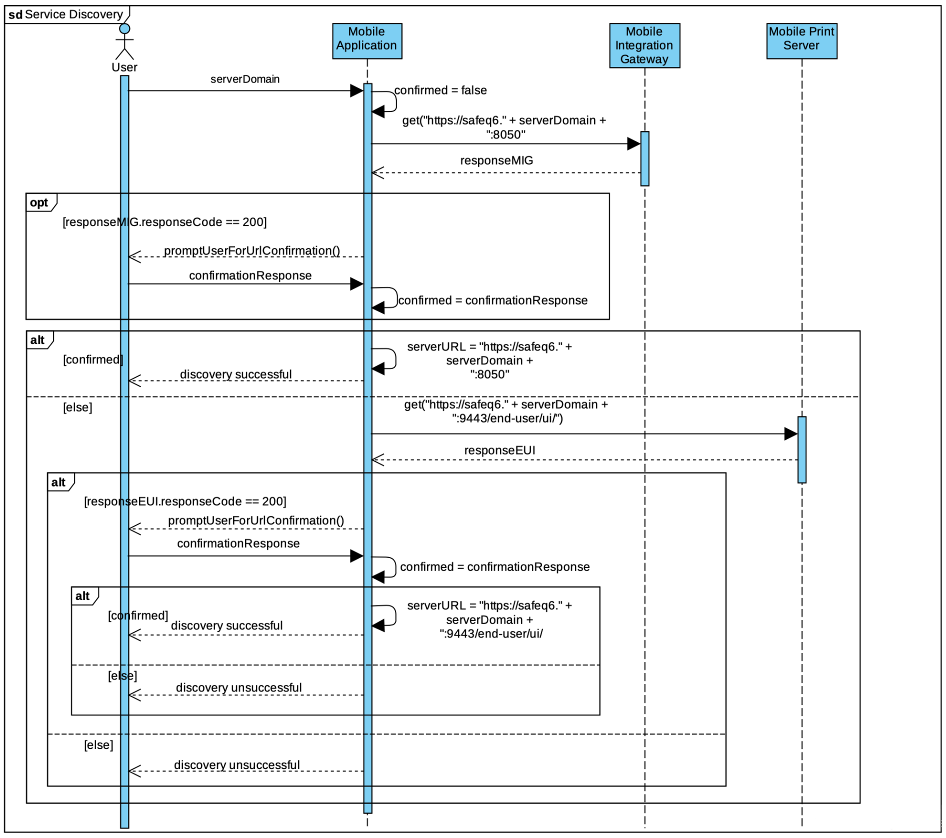

sd Service Discovery
User
Mobile Application
Mobile Integration Gateway
Mobile Print Server
serverDomain
confirmed = false
get("https://safeq6." + serverDomain + ":8050"
responseMIG
opt
[responseMIG.responseCode == 200]
promptUserForUrlConfirmation()
confirmationResponse
confirmed = confirmationResponse
alt
[confirmed]
serverURL = "https://safeq6." + serverDomain + ":8050"
discovery successful
[else]
get("https://safeq6." + serverDomain + ":9443/end-user/ui/")
responseEUI
alt
[responseEUI.responseCode == 200]
promptUserForUrlConfirmation()
confirmationResponse
confirmed = confirmationResponse
alt
[confirmed]
discovery successful
serverURL = "https://safeq6." + serverDomain + ":9443/end-user/ui/
[else]
discovery unsuccessful
[else]
discovery unsuccessful

Figure A.1: Service discovery

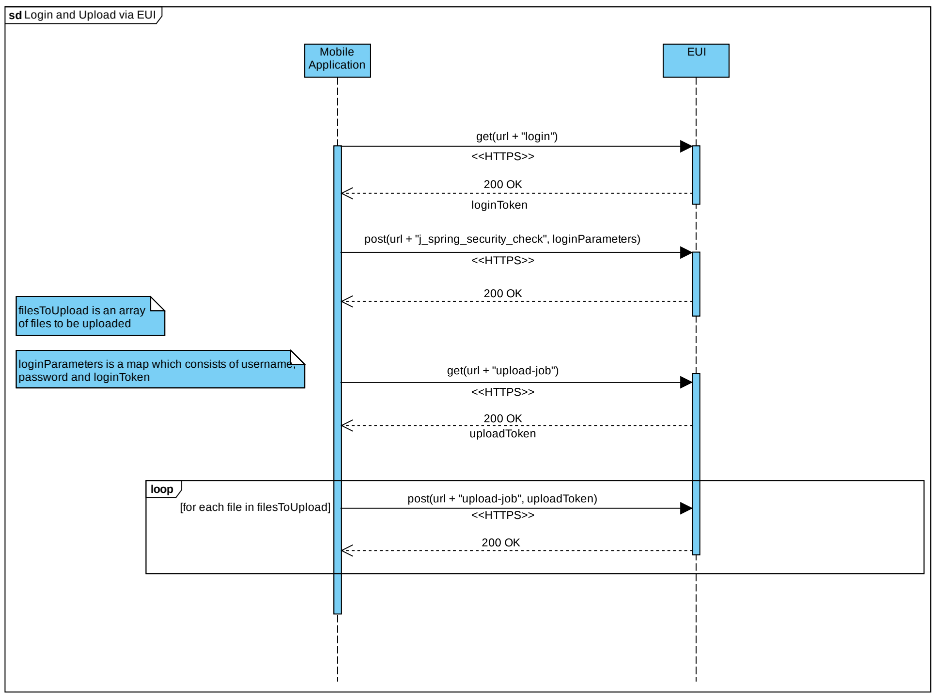

sd Login and Upload via EUI
Mobile Application
EUI
get(url + "login")
<<HTTPS>>
200 OK
loginToken
post(url + "j_spring_security_check", loginParameters)
<<HTTPS>>
200 OK
filesToUpload is an array of files to be uploaded
loginParameters is a map which consists of username, password and loginToken
get(url + "upload-job")
<<HTTPS>>
200 OK
uploadToken
loop
[for each file in filesToUpload]
post(url + "upload-job", uploadToken)
<<HTTPS>>
200 OK

Figure A.2: The EUI login and upload

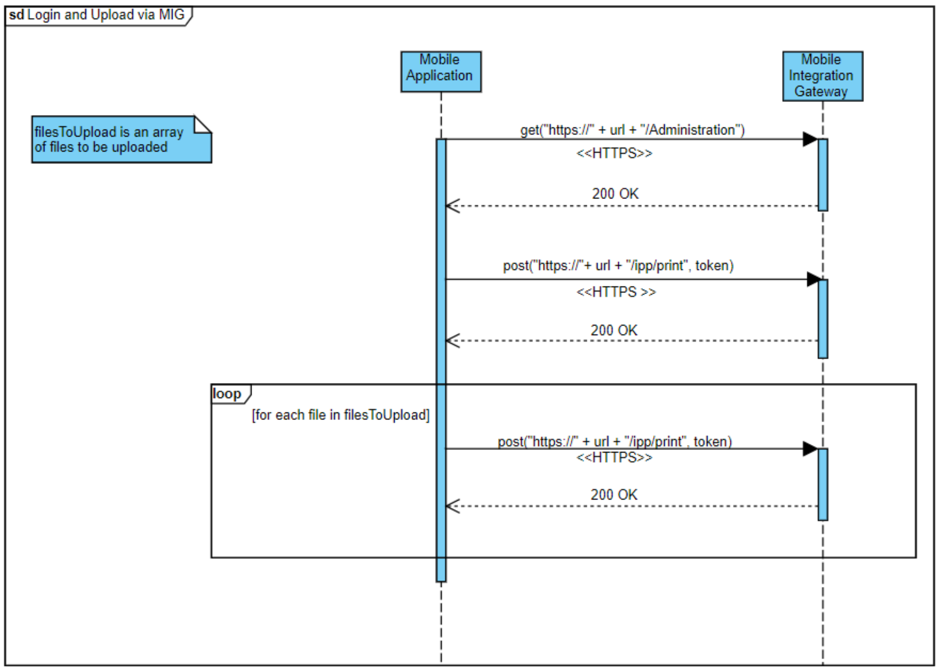

sd Login and Upload via MIG
Mobile Application
Mobile Integration Gateway
filesToUpload is an array of files to be uploaded
get("https://" + url + "/Administration")
<<HTTPS>>
200 OK
post("https://"+ url + "/ipp/print", token)
<<HTTPS >>
200 OK
loop
[for each file in filesToUpload]
post("https://" + url + "/ipp/print", token)
<<HTTPS>>
200 OK

Figure A.3: The MIG login and upload