



**University of
Zurich**^{UZH}

Deploying a Mobile Application for Digital Onboarding

*Nicolas Gordillo
Zürich, Switzerland
Student ID: 17-732-744*

Supervisor: Bruno Rodrigues, Christian Killer, Thomas Bocek
Date of Submission: April 6, 2020

Abstract

Currently the only way to acquire a phone / internet subscription in Switzerland is by contacting with a provider such as Salt, Swisscom or Sunrise and requesting one of the various plans offered to receive a SIM Card with service. The process usually starts with a phone call to the provider or by filling a form within their website. Then, documents have to be signed by the customer and an activation fee of 50 Swiss Francs has to be paid for the new SIM Card. The thesis at hand presents a novel digital on-boarding process to acquire a new mobile subscription. The main goal is to produce an application publicly available to customers on Android and iOS designed to simplify how people acquire new SIM Cards and the amount of money needed to pay for the subscriptions. It is expected to eliminate the paperwork required when creating a new subscription with a provider, remove the costs of ordering a new SIM Card and to offer flat payments without any hidden costs to compete with the big service providers such as Salt or Swisscom. Within the thesis, it is explained how the application should be developed following the latest guidelines of human-interaction to create a designed product easy to use and understand for everyone. The architecture used by the application for the subscription management is explained in detail and the economic impact of the application in the market is analysed and presented within the results of a real world use case of this application.

0.1 Kurzfassung

Die einzige Möglichkeit, ein Telefon-/Internet-Abonnement in der Schweiz zu erwerben, besteht derzeit darin, sich mit einem Anbieter wie Salt, Swisscom oder Sunrise in Verbindung zu setzen und einen der verschiedenen angebotenen Tarife für den Erhalt einer SIM-Karte mit Service zu beantragen. Der Prozess beginnt in der Regel mit einem Telefonanruf an den Anbieter oder mit dem Ausfüllen eines Formulars auf dessen Website. Dann müssen die Dokumente vom Kunden unterschrieben werden und eine Aktivierungsgebühr von 50 Schweizer Franken für die neue SIM-Karte bezahlt werden. In der vorliegenden Arbeit wird ein neuartiges digitales On-Boarding-Verfahren zum Erwerb eines neuen Mobilfunkabonnements vorgestellt. Das Hauptziel ist die Erstellung einer Anwendung, die den Kunden auf Android und iOS öffentlich zugänglich ist und die den Erwerb von neuen SIM-Karten und die Bezahlung der Abonnements vereinfachen soll. Es wird erwartet, dass der Papierkram, der bei der Erstellung eines neuen Abonnements bei einem Anbieter anfällt, beseitigt wird, dass die Kosten für die Bestellung einer neuen SIM-Karte wegfallen und dass Pauschalzahlungen ohne versteckte Kosten angeboten werden, um mit den grossen Serviceanbietern wie Salt oder Swisscom zu konkurrieren. In der Diplomarbeit wird erläutert, wie die Anwendung nach den neuesten Richtlinien der menschlichen Interaktion entwickelt werden sollte, um ein leicht zu bedienendes und für jeden verständliches Produkt zu schaffen. Die Architektur, welche von der Applikation für die Abonnementverwaltung verwendet wird, wird im Detail erläutert sowie die wirtschaftlichen Auswirkungen der Anwendung auf den Markt werden analysiert und im Rahmen der Ergebnissen eines realen Anwendungsfalles dieser Applikation präsentiert.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Thomas Bocek, for his support and feedback during the whole process of this thesis and for his knowledge provided whenever I faced an architecture issue. I would also like to thank Bruno Rodrigues for providing me with guidelines for the thesis and feedback on my progress. Additionally, I would like to thank Professor Dr. Burkhard Stiller for the opportunity to write my thesis at his chair.

Contents

Abstract	i
0.1 Kurzfassung	ii
Acknowledgments	iii
1 Introduction	1
1.1 Motivation	2
1.2 Description of Work	3
1.3 Thesis Outline	4
2 Related Work	5
2.1 Digital transformation	5
2.1.1 Acquiring a new subscription in Switzerland	7
2.2 User Experience Within Mobile Applications	8
2.2.1 User Experience Guidelines Applicable to a digital transformation method	8
3 Design Architecture	11
3.1 Functional Requirements	11
3.1.1 Login / Sign up	12
3.1.2 Solution Presentation	13
3.1.3 Address verification	13
3.1.4 Identity Verification	14
3.1.5 Payment method	15

3.1.6	Sim Card activation	15
3.1.7	Activate subscription	15
4	Implementation	17
4.1	Frontend	17
4.1.1	React Native	17
4.1.2	Login using Social Providers	19
4.1.3	Redux	22
4.2	Backend	24
4.2.1	AWS	24
4.2.2	Killbill Subscription Management	32
4.2.3	Swiss Post Free Address Verification	35
4.2.4	Onfido Identity Verification	36
4.2.5	Funke Delivery System	38
4.2.6	Datatrans Payment Service Provider	38
4.2.7	Zendesk Customer Support	39
4.3	MVNO	40
4.4	End-to-end architecture and API calls	41
5	Evaluation	43
5.1	Mobile application launched for Android and iOS	43
5.1.1	Application statistics	43
5.1.2	System Usability Score	45
5.1.3	User experience questionnaire	47
5.2	AWS Usage	55
5.2.1	Description of costs for the first three months using AWS	55
5.3	Discussion and Findings	56
5.3.1	Issues with the payment method	56
5.3.2	Sim Card being activated after 10+ minutes	57

<i>CONTENTS</i>	vii
5.3.3 Misunderstood publicity and advertisement	58
5.3.4 Identity Verification instructions	58
5.4 Future Deployments	58
6 Conclusions and Future Work	59
6.1 Future Work	60
Bibliography	61
List of Figures	65
List of Tables	67
List of Listings	69
A Installation Guidelines	73
B Contents of the CD	75

Chapter 1

Introduction

There are many mobile providers in Switzerland offering services such as local or international phone calls and internet subscriptions. The biggest companies are Salt, Swisscom and Sunrise given they own the majority of the infrastructure on which the subscriptions run on [50]. According to the Federal Communications Commission ComCom from Switzerland, Swisscom is the company with the biggest market share with 58.3%, Sunrise the second biggest with 24.9% and then Salt with 16.8% [50]. Smaller companies operate as mobile virtual network operators from these 3 big companies to provide cheaper plans to the customers, companies such as Lebara, Wingo, UPC and Yallo. The entire Swiss market is divided into two parts: the prepaid and the postpaid market. The prepaid plans are the ones which it is necessary to pay for the service before using it. The postpaid is usually paid after usage of the subscription plan and it can only be acquired as monthly plans in every service provider [51, 52, 53].

A new method is proposed which attempts to disrupt the market by offering a new type of subscription alongside the monthly subscription: the daily subscription. In other words, users can now pay for a postpaid subscription in a daily manner via an opt-in / opt-out manner. However, it is not sufficient to offer a daily subscription in the same way as the competition, only through a website [51, 52, 53]. The new method includes a mobile application available for iOS and Android. An application capable of controlling the subscription plans on the Sim Cards in a matter of seconds, developed with the latest guidelines of user experience. Additionally, this method includes an on-boarding process not ever developed by any of the large companies controlling the market share. Within the on-boarding process, users are able to verify their identity, order a free Sim Card (vs 50 francs in every other operator [51, 52, 53]) and add a payment method to speed up acquiring a new subscription plan.

It is necessary to opt for a digital solution to attract new customers. However, it is not sufficient to offer attractive prices or more flexible subscriptions. Nowadays, a good user experience and easy to understand user interface are highly valued by customers who are interested in buying a product like the one described above [56] [54]. By following this approach, customers can decide to opt in or out of a subscription by using only the application installed on their mobile devices.

The attractive part of this new type of subscriptions is the ability to choose between a daily or a monthly subscription. The payment method is prepaid, which means as soon as a subscription is activated, a payment will be charged to the user. To simplify the process of switching providers or getting a new plan, the digital innovation trait is leveraged and the entire on-boarding process is expected to be turned into a digital manner [56]. There should not be any contracts to sign for the users or waiting time until the subscription is processed to be activated. Additionally, the Sim Cards should be free and deliverable within one day after being ordered to enrich the entire process for the users and provide a seamless and easy to understand experience. Even better, the SIM Card can be eliminated from the process if a possibility of installing an ESim[1] (Electronic Sim Card) through the application is feasible. The use and implementation of an ESim will be determined by analysing how many devices are already compatible with this new feature which will be discussed further down.

There are various services which have to be orchestrated in order to provide a mobile subscription to the users. Starting with the mobile application as frontend designed for users to directly interact with. Then, comes the backend which is in charge of communicating the app with every other service attached to it. One of the services is Killbill [43], the subscription manager which handles the phone numbers of each account, which subscription is active for every number and it is responsible of triggering payments as well. Next to this one, is the service for customer support. Zendesk [47] is a powerful front to communicate with users whenever they may reach into a problem with their subscription, payment methods or SIM Card deliveries.

Two services are directly attached to the mobile app. The first one is Onfido [11] identity verification which is needed for users to identify themselves using their passport or identity card. This service is connected to the backend to provide information about the person and the results of the verification. The second service is Datatrans [13], which is a Swiss company dedicated to process online payments. By using this service, the application can directly request users to add a credit card as a payment method or TWINT [12] in a matter of seconds.

1.1 Motivation

This method is created in order to present a novel approach which simplifies the process and paperwork needed when ordering a new phone number from a provider and a subscription attached to it [56]. The current process is a tedious and expensive one, where customers are forced to wait long periods of time until they can start using their subscription. Within this period of time, paperwork must be filled and signed in order to receive a sim card from the provider [71]. It is quite common to find people who are using only prepaid numbers just because they want to avoid the hassle of having a fix subscription with a provider given all the side effects which come attached to them.

A common side effect of a post paid subscription is hidden costs which trigger extra payments at the end of the month. One example of a hidden cost with Salt is deactivating airplane mode while being in a different country than Switzerland. The phone tried to

connect to an antenna without success while deactivating the airplane mode. However, Salt charges customers 12 francs for this attempt to connect while being abroad.

It is also necessary to speed up the process of getting a new number and subscriptions. A guideline needs to be found which converts the entire process into a digital one which can somehow maintain the legal validity of someone ordering a new subscription without any signatures or physical visits to the providers stores [71]. A legal and valid alternative has to be found within the Swiss laws to take advantage of an entire digital experience given the requirements set By Switzerland when purchasing a new mobile subscription [71].

1.2 Description of Work

This thesis proposes a new method where it is possible to order a new number and a subscription within a mobile application as a user. The method consists of two parts, both of them being equally important to achieve the goal of a functional mobile application. The end product of the digital experience should be a mobile application which is usable in today's most iconic mobile operating systems: Android and iOS.

Firstly, it is described how the mobile application should be organized and what are the necessary steps the user must fulfill in order for the contract to be legally valid as a physical signed contract. Each step is explained in detail given the importance of the workflow of the application. Each step must be completely fulfilled before the users can move forward within the app and discovering it's features. The order of the application steps are as described as the following:

1. Create an account using a social login or email / password combination
2. If email / password combination was chosen, the email needs to be verified
3. Order a new Sim Card if needed with a valid Swiss address
4. Verify the user's identity using their passport or identity card (for certain countries)
5. Add a payment method using a credit card or TWINT
6. Scan the Sim Card to receive a phone number
7. Choose the type of subscription, daily or monthly

Each of these steps will be explained in further details with images and screenshots of the expected behaviour.

Secondly, the backend is explained. Topics are covered such as which is the best architecture for a backend when building a mobile app. The features of the backend is thoroughly detailed and AWS microservices are explained with a serverless architecture. Afterwards, the list of functions included in the backend are shown and the connection to the various

services can be exemplified to understand the complexity of what needs to be orchestrated when providing such a service as this one. A mention to security is included in the guidelines and how protected from outsiders the backend should be.

Additionally, an evaluation of the method is presented providing statistics from the application and from the user interface. An economic analysis was conducted to better understand the advantages and disadvantages of a serverless backend architecture as well as the economic impact of a full digital on-boarding process versus the regular process offered today by different companies.

1.3 Thesis Outline

This thesis is organized as follows: in Chapter 2 describes the different research in recent literature introducing the ESim and the latest guidelines in human interaction for mobile applications. Chapter 3 presents the overall design architecture of the application and its connection to the backend described in this thesis. Chapter 4 continues with in-depth details from the implementation point of view. Chapter 5 presents an evaluation of the architecture chosen for this project versus other backend architectures, as well as the economic impact of having an entire digital on-boarding process versus the existing one. Chapter 6 closes the thesis with the final considerations and future work.

Chapter 2

Related Work

This chapter presents related work in the field of digital transformation. Every work presented below has the goal of transforming a non-digital process into a digital one. [56] describes digital transformation as the use of technology to radically improve performance or reach of enterprises. They mention the need to transform the customer experience, especially in the customer touch points. The digital tools allow customers to save time, while saving the company money [56].

2.1 Digital transformation

In 2012, UBS released their first mobile application for Android devices capable of accessing the bank account [57]. Three years later, in 2015, UBS released the first mobile application for iOS [63]. Since then, constant updates have been released for both platforms being the latest one January 16, 2020. Before having their mobile application, customers could only see their transactions and balance by going directly to the bank or by accessing their accounts via the bank's web page. To fulfill the customer need to access their banking information on the go, a mobile application had to be developed and offered to the public at no cost. The benefits of showing the customers the banking information directly within their mobile phones surpass the effort needed to build the application. The customers find the mobile application helpful, as can be inferred from the 4.7 / 5 star rating within the UBS Mobile Banking application in the Apple Store. This rating is an average from 65k users who have written reviews [63]. Therefore, it is clear a mobile application is the best way to quickly show customers information about their account and services they have previously activated. Instead of using a web page, it is possible to show users information about their phone number within the mobile application. Additionally, information about their subscriptions can also be seen. This approach to show information on the go is taken to build the experience within the method proposed in this thesis.

Due to the rise of services being offered in a digital way nowadays, companies offering mobile subscriptions also opted to offer an overview of their services in a digital manner.

Every major company offers a web portal where customers can access and see their current balance, their subscription plans and how much have they consumed from each service on the current month. Salt, Sunrise and Swisscom offer this portal to every customer subscribed to one of their services [51, 53, 52]. However, to display this information in a mobile application has yet to be developed. The only company who has tried to tackle this issue is Lebara with their mobile application *Lebara Switzerland App* [58]. This application is used for displaying purposes of a subscription, not to interact with it in any way. It is usually slow and it does not have the look and feel of a native application. Only in the latest update (September 2019), it is now possible to add credit to the account via a credit card. Lebara Switzerland App implemented Datatrans as their payment service provider to validate and charge credit cards. Nevertheless, as it is not a native application, Datatrans has to be presented in a web view where the interface is designed by Datatrans and not by Lebara. This clearly shows the need for a native application to maintain the flow of a mobile application and not a mixture which interrupts the user's mental flow of a software system.

Swisscom also has a similar mobile application released on 2018 called *My Swisscom* [59]. It allows customers to see their bills and pending payments, overview of the subscriptions and assistance for technical problems. However, when looking for a new subscription it offers a feature locating Swisscom shops near your location and their opening hours. It does not offer any type of in-app purchase, nor activation or deactivation of services. It is clear the need for an application which allows the customer to opt-in or opt-out of subscriptions without restrictions.

As a result, there is a need of a mobile application to attract customers and keep them engaged. By analysing the methods mentioned before, it is clear the method proposed in this thesis must be delivered for both platforms: Android and iOS. However, just delivering a mobile application is not enough. As seen on the approach taken by Lebara and Swisscom, a mobile application with no significant user experience results in bad reviews from customers and overall dislike. For the method proposed within this thesis, a great deal of effort is spent on delivering a great user experience in every screen presented to the potential customers.

The approach taken by Revolut [64] to present their products and on-board the potential customers is a well organized process within their mobile application. Simple, efficient and easy to understand. Revolut tries to minimize the amount of interactions needed from the user within the on-boarding process. In most of the screens, there is only one button which can be clicked or one input field. The input fields for the address can be filled automatically by choosing the right address when the user starts to type their address. Revolut advertise their on-boarding process as "registering a new account in less than 60 seconds" [65]. The way Revolut presents their on-boarding process to the customer is of great value to the method presented within this thesis. In addition, Revolut allows users to block and unblock their debit cards within seconds and with no struggle at all. The same behaviour is taken to activate / deactivate subscriptions within the method proposed by this thesis. It should not be difficult for a user to opt in or out of a paid service.

2.1.1 Acquiring a new subscription in Switzerland

As of today, it is not possible to acquire a new mobile subscription in a digital manner in Switzerland. Every major telecommunications company offers a section on their website to purchase a new subscription [52, 53, 51]. However, each section only leads to a form which has to be manually filled and a sales agent will contact the customer to move forward with the sale. Mostly, this behaviour occurs given the mandatory identity verification which is done manually by the network providers. According to the Schweizerische Bundesrat, in their category 780 Überwachung (Monitoring), within their compilation 780.11: Verordnung über die Überwachung des Post- und Fernmeldeverkehrs (Ordinance on the Interception of Postal and Telecommunications Traffic), it is obligatory to perform an identity verification check on every new subscription being sold to a customer since March 2018 [71]. Within this compilation, the article 20 can be found which states the need to identify a person using their passport, identity card or foreigner's identity card. Additionally, personal information has to be collected such as:

1. First name and last name
2. Date of birth
3. Type of permit, number and who issued it
4. Address
5. Profession
6. Nationality

Which steps are necessary to acquire a new subscription in Switzerland

The usual steps to acquire a new subscription are the following:

1. The customer must fill in a web form found in a provider's website
2. The customer needs to wait for a confirmation email or a phone call from the sales agent with further instructions
3. The customer must then receive physical documents at their personal or work address and return them signed. These documents can be either sent by post or returned at one of the provider's shops.
4. The identity of the customer has to be verified. Together with the signed papers, a copy of the customer's passport, national card or residence's permit must be attached.
5. 50 francs have to be paid for the new Sim Card which is then received after 3 business days.

6. The service will be then activated after a period of 5-7 business days.

The entire duration of this process is on average 2-3 weeks since the web form is filled. It is clear the need to speed up the process and to reduce the interactions between customer and provider. Not only to reduce costs as a provider, but also to improve the customer's experience while ordering a new subscription. The need exists to remove physical paper-work within the process and processing fees such as the 50 francs charged for the new Sim Card.

2.2 User Experience Within Mobile Applications

The need for good user experiences boomed within 2007 and 2009 [60]. The start of the movement happened because Apple released the first iPhone which now included an auto-rotate sensor, a capacitive screen that allowed multiple inputs while ignoring minor touches [60]. A year later, Android came to play with the G1 phone. Apple also launches the App store with 552 apps available to download [60]. At this point, the trend became clear given the global acceptance for Android and iOS devices. With this acceptance, also developers around the world took interest and began writing code to publish their own applications which could now run within these devices.

Therefore, within 2007 and 2009, research for a good mobile experience was the hot topic. Guidelines were written and best practices were discussed for both platforms while designing, writing and planning user interfaces.

2.2.1 User Experience Guidelines Applicable to a digital transformation method

As explained by [61], it is quite important to maintain a state within the mobile application. Starting with the authentication part, it is necessary to ask a user only to login once. It does not matter whether the mobile application is closed, left in the background or if the mobile phone is turned off, if the application is reopened, the user should still be signed in. Moreover, it is important to keep an state of where the user left off their operation of the mobile application. This is intuitive and explained by [62], a mobile phone might being used by a person while they are walking or taking the public transportation. It is not uncommon users are interrupted during these activities often. They might have to leave the bus to walk home or to switch to another public transport connection. If the mobile application process is interrupted and the users need to restart the whole process all over again, a sense of dissatisfaction is instantly noted from the users side [62].

[55] provides insight on the *feedback* the mobile application should provide to users when an action was completed or triggered. After a user clicks on a button or triggers an action within a screen from the mobile application, visual or haptic feedback has to be sent to the user to reinforce what they just did was processed by the mobile application. [55] illustrates their point by providing tips on what to present when an action is still pending,

successfully triggered or when an error has occurred. The user needs to be informed from every stage of what's happening in the background and needs to be instructed what to do next. If no feedback is provided to the users, they will try to click many times on the same button waiting for something to happen (even though the process might have already started in the background), or the users might even get tired and close the mobile application because they have no clue what is actually happening.

Gronier et al. [54] explains the importance of reducing waiting times within a mobile application. If after triggering an action, the mobile application takes more than 10 seconds to display the next screen or an explanatory message, users will think something went wrong and they need to repeat the action or close the mobile application and open it again. The difference with the feedback guidelines is that in most of the scenarios, a loader or spinner is present within the mobile application screen after the user triggered the action. The problem relays on the spinner staying on the screen for more than 10 seconds, forcing the user to think an error has occurred. To achieve the goal of reducing the waiting time, it is possible to split the tasks the mobile application has to do. As stated by [54], most of the heavy-duty tasks should be done by the backend process communicating with the mobile application. The mobile application should only be a front displaying information received by the backend process. No heavy processing should occur within the mobile application forcing the phone to deliver capabilities and processing power to the application. This helps to reduce not only the battery consumed by the mobile application, but also the waiting times from action to action. Moreover, if possible, the backend process should be hosted in an efficient environment where it is able to handle requests from various users at the time and will not be capped by over-querying or too many requests affected the performance and service delivery blocking the capacity of the backend process to respond in a quickly and swiftly manner.

Every guideline is taken into account while developing the method proposed by this thesis. In order to create a good user experience, a mixture of approaches needs to be implemented. Therefore, it is possible to extract various points from the guidelines presented before and apply them to the digital on-boarding process which will convert a traditional paper-full process to a digital one.

Chapter 3

Design Architecture

In this chapter the structure of the mobile application is explained. The on-boarding steps are presented and the desired flow of the application can be seen with every screen.

3.1 Functional Requirements

The initial requirements for the mobile application were identified, based on related work and the Swiss law mentioned [71], as:

- The mobile application has to run in both Android and iOS devices with only one codebase
- Every step from the on-boarding process must be digital and it has to be fulfilled within the mobile application
- There cannot be anonymous users. Users can sign up by email or by using a social provider
- The mobile application needs to support 4 languages
- The mobile application must be fast, intuitive and simple

In order to make the application attractive for users some requirements have to be met and implemented for a better user experience. At first, it is important to always render a clean and simple screen. It must be intuitive for users to understand what it is expected of them and which information is being communicated. With every screen, users are allowed to discover new bits and parts of the application in a controlled manner. It has been proven by [7] that a burst of information or large texts are disliked by users and do not offer a benefits when transmitting information. For this reason, the information which is sent through to the users has to be divided within many screens. It is necessary to implement a correct order to build the right mental procedure and keep the users engaged during the whole process.

There are screens which are just displaying some kind of information. For example, the subscription prices or the users' email. Other screens are designed to receive information from the customer. For example, input fields asking for the users' address. Within the application, it is possible to change the language to better fulfill a user's experience. The application should be offered in 3 out of the 4 official languages in Switzerland plus English to make it accessible for everyone. That means the application should be offered in German, French, Italian and English. In any point of the on-boarding process, user's are allowed to change their language if needed.

To maintain and fulfill the latest guidelines concerning human interaction and user experience, a designer was found who helped throughout the process of creating the screens and deciding which is the correct order to create a more understandable mental model of the application workflow. This person also mentioned the need of movement within the screens and transition feedback from every navigation within the application. It was pointed out that to highlight the product and the subscriptions, animations had to be implemented to keep the user engaged. Furthermore, a loading animation has to be seen whenever the application is navigating to a different screen or fetching information from the backend. This is aimed to give feedback to the users of their actions. If it's loading, they know it's time to wait and the process is advancing as expected. As it is described by [55], application with no feedback after a user's action, are deemed as defective or rated as a poor user experience in most cases. Additionally, the loading times have to be always below 10 seconds as explained by [54]. If it exceeds 10 seconds, users usually think the application is broken and it needs to be restarted or in the worst case, they just get tired and stop using the application at all.

These requirements have to be fitted within the digital on-boarding process to meet legal, business and marketing constraints and can be seen in Figure 3.1.

3.1.1 Login / Sign up

The first business requirements is no anonymous users. Every user has to be logged in in order to use the application. It is possible to create an account using a social provider like Google, Facebook or Apple. Additionally, users can also create an account with a combination of email and password. However, by using this approach the email has to be verified. Additionally, this requirement is taken from the article 20 from the Schweizerische Bundesrat [71], stating the need of an account for the customer to be verified on.

The input fields are also recommended to have extra features to give them an edge over regular input fields in other mobile applications. A small but intuitive animation was also added to every input field which is triggered when the field is focused and the keyboard opens to start typing text. The font size of the placeholder is reduced and also the position of the label is moved to the top of the input field container.

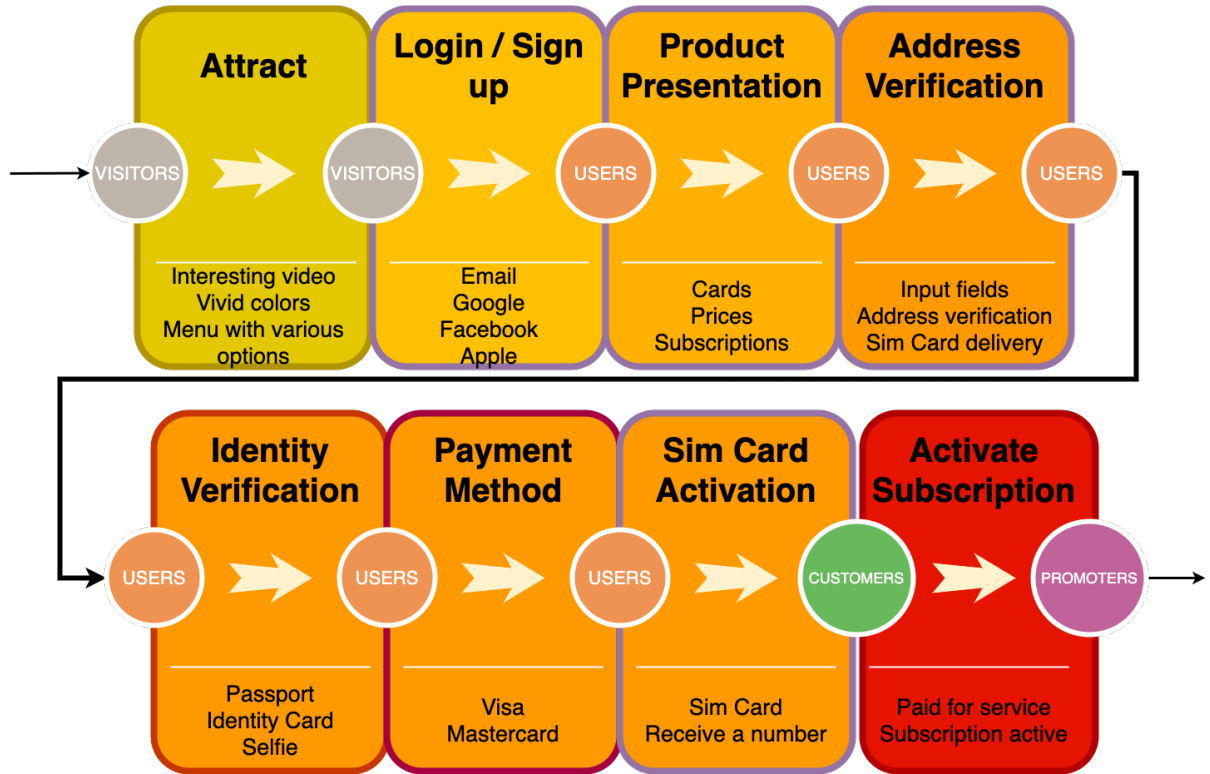


Figure 3.1: Workflow of the method

3.1.2 Solution Presentation

The next business requirement is to present the product in a clean manner, by using the least amount of information needed. As proposed by [8], the available subscriptions are presented in the form of cards. The front part of the card displays some crucial facts about the daily subscription and the back of the card displays information about the monthly subscription. To comply with the guidelines for human interaction and best practices for user experience, animations are to be included within this two cards. To make it understandable the card has to be "flippable", which means using a gesture users can see the back or front of the card with no obstacles or difficulties whatsoever. Following a marketing recommendation, the back side of the card is shown first to the users. This causes the users to see first the monthly subscription and might persuade them to purchase that type of subscription instead of a daily one.

3.1.3 Address verification

The next business requirement is to have the user's address. This is for legal and billing purposes, a user always need to gave an address linked to their account. In this step, the users can decide to order a new Sim Card to the provided address or not. If they do choose to order a Sim Card to be delivered to their address, this is delivered for free. There is no charge for ordering new Sim Cards which is one of the attractive offers of the digital subscriptions management compared to the other big telecommunications providers. They always charge 50 Swiss Francs per Sim Card [51, 52, 53].

The mandatory fields are:

- First name
- Last name
- Street
- Nr
- Postcode
- City
- Name on Post Box, c/o

The information is sent to an address verification API and depending on the response, the user will be able to move forward within the mobile application. This is according to the Schweizerische Bundesrat law, with article 20 [71] stating the need of a valid Swiss address attached to the user account.

3.1.4 Identity Verification

In order to launch a telecommunications service in Switzerland, it is mandatory to include an identity verification within the software system. In this case, the identity verification needs to be a required step within the on-boarding process of the mobile application. To be a valid and legal identity verification, users can identify themselves using their passport or their identity card if they live within the European union. Additionally, the users needs to take a picture of their face to prove who they are.

To accomplish this identity verification, it is necessary to use a third-party service which is specialized in this field.

Given the application needs sensitive data like a picture of the identity card or the user's face, the third-party library is needed to be integrated within the mobile application. More information about the integration of the library will be discussed in Chapter 4 as well as information of the parameters used to verify an identity.

As stated before, the non-digital process requires the service provider to identify the person activating a new subscription. The process consists of sending a copy of the passport to the service provider or personally visiting one of their stores. By implementing the identity verification as a service within the mobile application, it is possible to bypass the manual verification done before in the non-digital process. Additionally, this is an obligatory service imposed by the Schweizerische Bundesrat in March 2018 [71] where every customer needs to be properly identified to purchase a new mobile subscription.

3.1.5 Payment method

It is a business requirement to try to keep the users always with a valid payment method within their accounts. Therefore, a required step during the on-boarding process needs to be the insertion of a valid credit card or Twint [12] as a payment method. Twint is a common payment method in Switzerland. It was developed by a Swiss company aimed to connect bank accounts to small, quick payments. Their functions include sending or receiving money from other people who also have the Twint application installed. Additionally, it is possible to make online payments in a recurring form using the application.

A third-party service is needed to validate the credit card information typed in by the user. The full implementation details and parameters sent to the third-party service will be discussed in Chapter 4.

To keep the same look-and-feel within the screens, a flip animation needs to be also implemented within the credit card view. This is given the credit card also has a front and back side of the card, exactly like the one described in the Product Overview subsection. The information typed in by the users has to be seen written over the credit card as well, in the front and back side.

3.1.6 Sim Card activation

The next step of the on-boarding process can only be completed once the customer receives a Sim Card from the service provider. The Sim Card has to be scanned or the 20 digits number has to be manually inserted into the application to receive a phone number. The phone number will be linked to the user's account and will be able to receive calls. At this point in time, the phone number cannot perform calls and it does not have any connection to the internet until a subscription is bought later on. It is important to note that given the requirement of the Swiss law which says the users must be verified to buy subscriptions for the phone number, the mobile application is stuck in this step until the verification is successful. If the identity could not be verified, the users can decide to undergo another time with the verification or contact customer support.

3.1.7 Activate subscription

The last step of the on-boarding process is to activate the subscription with either a daily or a monthly plan. To make it a bit more interesting to receive a number, an animation was added to the new phone number which the user gets.

To keep the same look-and-feel within the whole on-boarding process, the last step also consist of presenting the front and back card with the information of each subscription. However, in this step the user can already decide to purchase one of the two subscription plans. These will be billed either daily or on a monthly basis. To prevent mistakes, a confirmation popup will appear to ask the user if that action is really the one they intended to perform. This technique comes from the research of [14] where it is stated

a user experience improves much better when it is less prone to errors, especially human errors when a user accidentally purchases an undesired subscription.

Chapter 4

Implementation

In this chapter, the technical implementation of the mobile application is explained. First, the technologies used to build the mobile application are detailed followed by a close-up to the backend powered by AWS microservices and its connection to every service which facilitates the digital on-boarding process of the mobile application.

4.1 Frontend

4.1.1 React Native

Given a key requirement from the application is to run in both Android and iOS platforms, the best cross-platform framework had to be chosen to fulfill the task. The framework that fits the best is React Native [15]. React native is an open-source framework being developed by Facebook. It has not officially been released yet as it is still below the first version, being currently in version 0.61 (January 2020). React Native uses JavaScript as its motor to transform code into native code for Android and iOS. In other words, the code is written in JavaScript using React, which in turn, React primitives render this code to native platform UI.

React Native was also created to tackle a known issue from the other cross-platform frameworks which is the laggy user experience when using the application given the elements are being rendered within a Webview. React Native achieves 60 frames per second [16] and delivers a native look and feel to the mobile applications.

Additionally, React Native is designed to save time to the developers who are using it. First of all, it eliminates the need to have two code bases: one for Android and one for iOS. In many companies, this meant having two different teams working on the same project. However, with React Native one team can control releases for both platforms. It also offers 'Fast Refresh' [15], which allows developers to see their changes on the emulators or mobile devices as soon as the changes are saved on their favorite code editor. This is an extremely useful feature when working with agile methods or directly with the customer who needs to approve how a screen should look like.

It has a few minimum requirements. For example, React Native cannot run on Android devices under Android version 4.1 (API 16). Also it cannot run under iOS devices under iOS 9.0. It also requires a computer running macOS to see the mobile application being developed within the iOS emulator. One workaround to this need is to use their lightweight version of React Native: Expo [17]. Nonetheless, Expo is not used for this project given its lightweight capabilities and is out of the reach of this thesis.

By using React Native with its full capabilities, it is also possible to write native code when needed. Which means it is possible to write native code for iOS in Swift or native code for Android in Java alongside React. This is a useful feature for this project given it needs to be linked with third party libraries which may not have a direct integration with a JavaScript framework such as React Native. This was the case with two of the services used within the digital on-boarding process: Onfido and Datatrans. It will be explained what is needed to do to connect these libraries in subsection 4.2.4 and 4.2.6.

In addition to the information mentioned above, React Native has a big community of active developers working with it. Its repository in GitHub has around 84k stars (January 2020). The comparison of stars against other well-known cross-platforms frameworks can be seen in table 4.1.

Framework	Year Released	Stars on Github
React Native	2015	84k
Angular JS	2016	56k
Ionic	2013	40k
Native Script	2014	18k

Table 4.1: Comparison of different open-source cross-platform frameworks

Additionally, big enterprises like Instagram, Uber or Amazon Prime Video already use React Native to develop their mobile applications. This has created a big wave of motivation to work with React Native and to write libraries well maintained. One of the most important library for this project is 'React Native Reanimated'. This library leverages a useful feature from React Native which is its declarative components and connects them to the native thread of the mobile device.

There are two ways of running animations in a cross-platform framework. The first and easy way is to create the animations by using the JavaScript thread. In this case, the animation follows the thread limitations and lifecycle. For simple animations, this approach works just fine. However, when complex animations are introduced, the thread is blocked and the animations appear laggy. The second approach is to declare the animation in JavaScript, but have them run on the native thread. Native code will leverage the mobile device's capabilities and will them to render animations within the 60 frames per second which React Native offers in its performance. Reanimated provides a more comprehensive, low level abstraction for the Animated library API, giving much greater flexibility, control and performance [18].

Nevertheless, using this library can be cumbersome and not so straight-forward to learn. To ease the workload of developers, another library was created on top of Reanimated

which greatly simplifies how animations have to be declared without losing the native thread capabilities. This library is called 'react-native-redash' and it is written by William Candon, a French developer currently living in Zürich. This library was used to build every animation which can be seen throughout the digital on-boarding process within the mobile application.

In total, the application built to show the digital on-boarding capabilities uses a total of 96 libraries written by third parties for React Native.

AWS Amplify

AWS Amplify is the bridge connecting every service offered by AWS to the mobile application. It is officially developed for JavaScript, iOS, Android and React Native [20]. The most important feature for the mobile application is the Authentication bridge to AWS Cognito. As discussed in subsection 4.2.1, Cognito is a simple and secure user sign up, login and access control [21]. If a user creates an account within the mobile application, a user will be created in Cognito. Credentials such as session information and token validity is stored within the mobile application by Amplify. Additionally, Amplify is in charge of refreshing user tokens when needed and checking the validity of the user session. For example, if the application is closed and then reopened, the user will expect to land on the last saved point, not having to login again. As seen in Listing 4.1, Amplify will return an authenticated user or undefined if there is none. With this information, the mobile application can redirect the user to the correct place. For instance, if it's a new user and there is no information coming from Amplify, the Authentication screen will be presented. However, if there is information about the user, a request will be sent to the backend asking in which point of the on-boarding process is the current user in.

```
1 import {Auth} from 'aws-amplify';  
2 return Auth.currentAuthenticatedUser();
```

Listing 4.1: Function returning an authenticated user or undefined

Furthermore, Amplify is also in charge of terminating a session by performing a sign out action on the user. As well as actions as forgot password, verify email address and change email address when needed.

4.1.2 Login using Social Providers

By logging in using a social provider or a regular email and password combination, it is possible to create a user on the backend. The service in charge of handling the passwords, sessions and tokens is Amazon Cognito.

However, an interesting point to note is how does the application should behave when a user forgets their password or needs to verify their email account. Amazon Cognito will send out an email to the user with a token to change their password or verify the

account. To keep a desirable user experience and workflow, deep-linking was implemented [2]. Deep-linking is the ability to invoke a mobile application from another application via a uniform resource identifier (URI). The URI will have a predefined format which is the name of the app (`app://`) and is followed by the rest of the content of the URI which will help to identify what to do in each case the application is invoked using this method. This means the user can open the email received on their phone (which is a third-party application used for receiving / reading emails) and click on a button which will open the mobile application again. When the application is opened, the user has the chance to set a new password and login or, in the case of verification, simply login to keep moving forward with the on-boarding process.

Apple login also uses deep-linking to bring back information about the user to the application. A website is opened in the mobile phone web browser and the user has the opportunity to login using their Apple credentials. Once the login is successful, the website uses a predefined URI to redirect back to the application and login the user.

It is important to state that Apple login was not a requirement from a business perspective. Instead, it was a requirement enforced by Apple since November 2019. No application may enter the official App Store unless it includes Apple Login as a login / sign up possibility next to Google or Facebook.

For the other social logins, Google and Facebook, the official SDK is implemented which permit the users to authenticate themselves within a webview inside the mobile application. By using this method, users are not forced to leave the application for authentication purposes. This method has some known pitfalls and is vulnerable to attacks. According to the National Institute of Standards and Technology, an exposed debugging endpoint was found in Google Chrome on Android which allowed a local attacker to obtain potentially sensitive information from process memory via a crafted Intent [3, 4]. For iOS, according to [5] by exploiting the `UIWebView` class, attackers can access the device's filesystem and [6] managed to create a cross-site scripting attack to a `UIWebView` class within the Hack In The Box held in Amsterdam in 2014. For these reasons, Apple is trying to enforce mobile applications to stop using in-app webviews. In a near future, it will only be possible to submit application to the App Store using authentication with deep-linking like the one described previously with Apple login or if the official SDKs use the new and secure class `WKWebView`. However, it is difficult to predict when will the SDKs be updated.

API Requests

Given the available capabilities of the backend, a REST API is used to communicate between the mobile application to the backend. As explained in subsection 4.2.1, it is possible to protect the public endpoints with a layer of authentication. The chosen authentication for the project is a controlled access determined by IAM Permissions [19]. For every request sent from the mobile application to the backend, a helper function is needed to authenticate the requests with the required parameters, injecting them into the request's headers. As seen in Listing 4.2, the credentials are first requested from Amplify as seen in line 14 of the code. By leveraging the information received from Amplify, an object is created containing the secret access key from the user, the access key id and the

session token. Additionally, AWS requires information about the request being sent to match the hash generated on their side. In line 24, it can be seen an object being created with the method of the HTTP request (which can be GET, POST, PUT, DELETE or PATCH), the url being contacted and the data which will be sent to the backend if there is any. Finally, in line 29, the signer class is called by using the method `sign()` which takes all the required parameters and creates a hash using AWS Signature Version 4 [22]. As stated by the Signature Version 4 documentation,

When an AWS service receives the request, it performs the same steps that you did to calculate the signature you sent in your request. AWS then compares its calculated signature to the one you sent with the request. If the signatures match, the request is processed. If the signatures don't match, the request is denied.

(AWS)

The request has to fully match or it will be denied by AWS and no response will be received by the mobile application.

```
1 import Auth from '@aws-amplify/auth';
2 import {Signer} from '@aws-amplify/core';
3
4 export const initializeCredentials = async (
5   url: string,
6   method: string,
7   body?: any,
8 ) => {
9   console.log(
10     'Calling the following END POINT: ',
11     url,
12     ' as a ',
13     method,
14     'with the following parameters: ',
15     body,
16   );
17   let cred = await Auth.currentCredentials();
18   const accessInfo = {
19     secret_key: cred.secretAccessKey,
20     access_key: cred.accessKeyId,
21     session_token: cred.sessionToken,
22   };
23   const serviceInfo = {
24     region: 'eu-central-1',
25     service: 'execute-api',
26   };
27   let req = {
28     method: method,
29     url: url,
30     data: body,
31   };
32   return Signer.sign(req, accessInfo, serviceInfo);
```

```
33 };
```

Listing 4.2: Function authenticating requests

In Listing 4.3, the credentials are being initialized with previously mentioned helper function and then the request is created by injected all the information returned by the Signer class. It is important to notice in line 2, the `getDomain()` function. This method is intended to keep the configuration parameters of the environment dynamic, allowing switching between the different environments (e.g., dev, stage, prod) with no major efforts. The different environments and stacks will be discussed in subsection 4.2.1.

```
1  const creds = await initializeCredentials(  
2    `${getDomain()}/users/${userId}`,  
3    HttpTypes.GET,  
4    null,  
5  );  
6  return fetch(creds.url, {  
7    headers: creds.headers,  
8    method: creds.method,  
9    body: creds.data,  
10  })
```

Listing 4.3: Headers being injected into the requests

4.1.3 Redux

Given the size of the mobile application and it's different components, Redux [23] is needed to keep track of a global state. Redux is a popular (52k stars in Github) open-source library which is intended to manage an application's state. It's goal is to share a state between the different components or screens which might be present in a mobile application. In simple applications, it is recommended to not use Redux or to avoid it until it becomes a necessity given the complexity of the application will rise considerably by using Redux. To quote Pete Hunt, one of the early contributors:

You'll know when you need Flux. If you aren't sure if you need it, you don't need it.

(Pete Hunt)

However, if an application has data which changes over time and it is needed to be shared between many screens, then a single source of truth is the appropriate approach to tackle the problem. In this case, Redux was chosen to be the manager of this single state. The two main components which is shared within the application are:

1. the authenticated user and all the known information about the user such as first name, last name, email, address, among other variables

2. the state where the authenticated user currently is within the on-boarding process

It makes sense to share the user within most of the screens of the mobile application. Within the settings, the user is able to find personal information or their email address. However, the user must also see their numbers and information associated with them which can only be retrieved by passing the user object to many components and screens. It would be too complicated to only query the user at the startup of the application and then manually transmit that object to every screen the user decides to navigate to.

For the second item, the state has to be stored within a global variable to keep track of how the user is advancing within the digital on-boarding process or which one should be the next screen presented.

As explained in Chapter 3, the user must navigate through a mandatory on-boarding process until they land into the on-boarded state, where the on-boarding process is terminated. The on-boarding state can be one of the following:

1. Authentication
2. Offer
3. KYC Document Selector
4. Credit Card
5. Scan QR Code
6. KYC Result
7. Activate Subscription
8. On Boarded

As soon as the mobile application is started, a query will be done to Amplify asking if there is an authenticated user. If not, an initial state will be set for the user starting in the Authentication screen. If there is an authenticated user, the mobile application will query the backend, asking in which of the previously mentioned states the user is at. The response will be saved within Redux and the user will be redirected to the correct screen for them to continue with the experience of the mobile application. This information is needed throughout the entire mobile application. Many screens displays different information depending on which step the user is currently in. For example, the screen containing input fields for the address is reused. It is first used to ask for a delivery address for the Sim Card, while later it is used for the billing address of the user or for the delivery of another Sim Card. It is the same screen presented to the user, but depending on which step they are currently in of the on-boarding process, the screen behaves differently.

4.2 Backend

4.2.1 AWS

The chosen provider to host the entire backend for the mobile application is Amazon Web Services. It provides a gateway for every needed for the digital on-boarding process, security and load-balancing layers as well as good prices per month of usage.

Serverless Architecture

Amazon offers their cloud services in a serverless manner. This means there are no servers to maintain, hardware limitations or computing power unused. The idea behind a serverless architecture is to have a flexible architecture which can be scaled automatically and paid by units of consumption. Instead of paying for an entire server which might be used only at 10-20% of its full potential, it makes more sense to pay for the services and resources used while the service is being requested. To make the offer even more appealing, the services which are crucial for the development of an applicant are offered for free up to a certain point. This is useful during development because the free tier is almost never surpassed and no extra costs are spent on the technical architecture. Additionally, AWS ships their products with load-balancers ensuring application fault tolerance and availability[25]. For example, there is no need to worry about possible DDoS[24] attacks.

Monolithic vs Microservices Architecture

A server-side application usually consists of various modules or components in charge of handling the whole backend infrastructure. The most common components according to [42] are:

1. Presentation, which handle the HTTP requests and responses to the frontend
2. Business logic, it performs all the required steps set by the business side to fulfill its purpose
3. Database access, reading and writing to the database
4. Application integration, integration of the infrastructure with other services via a REST[30] API.

These different components can be built using a monolithic or microservices architecture as seen in Figure 4.1.

Even though, a server-side application is clearly divided by a logically modular architecture, the infrastructure is usually packaged and deployed as a monolith [42]. A monolith architecture refers to an architecture where all the components are tightly coupled within

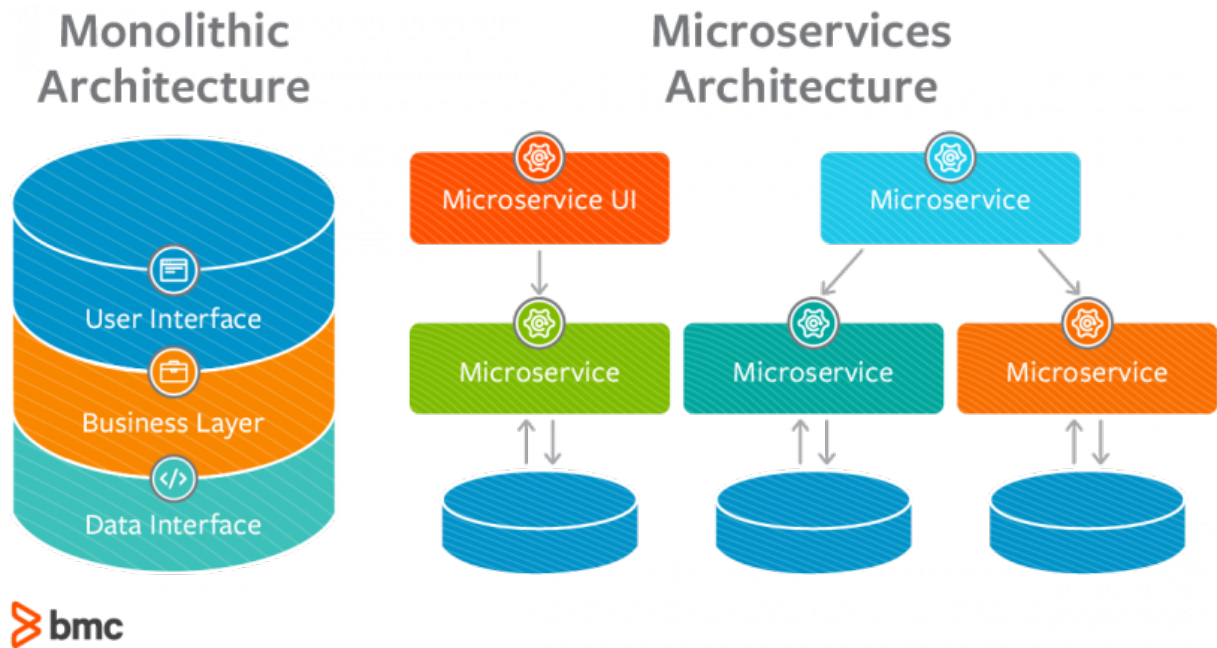


Figure 4.1: Monolithic vs Microservices architecture, taken from BMC.com

one API. This means it is simpler to develop given the logic of the whole server-side application resides in one place and it can be tested from end-to-end. It is not difficult to deploy to different instances given the API just needs to be replicated to different containers where it can be hosted and scaled. Furthermore, it is simple to scale horizontally by running multiple copies of the server-side application at the same time behind a load balancer [42].

However, this type of architecture also has some drawbacks. This approach grows in size and complexity rapidly as more components are added and becomes difficult to maintain. Developers who are just added to projects with this type of architecture require a long time to learn the ropes of the project and is too complex to be fully understood. Additionally, by having an application which grew in size rapidly, the start-up time is slowed. Given its properties of being a single API, if an update has to be made, the whole entire application has to be redeployed. Testing becomes difficult and most of the test cases are usually done manually instead of integrating an automated testing tool because that would also increase the complexity of the entire application. This also means continuous integration and deployment becomes too difficult and is mostly left out. Reliability starts to become a problem given bugs are replicated between the copied instances when scaling and if a critical runtime bug is present within any module (such as a memory leak), it can potentially bring down the entire application [42].

To tackle these drawbacks, a different server-side architecture is available which is based on microservices.

The microservices architecture split the application into smaller, independent services. These are loosely coupled and are exposed as a REST based API. Usually these services have their own database schema and they handle their own data, which is not written or read by any other microservice. This opens up new possibilities where different databases

might be used by the server-side application, and these databases do not have to follow the same schema. One might be a NOSQL database, while the others can be MySQL databases based on relationships within their tables. The microservices cannot be directly accessed, they are managed by the API Gateway 4.2.1. The API Gateway is responsible for handling infrastructure tasks such as load balancing, caching, access control and monitoring [42]. By having separate microservices, they do not tend to grow in complexity as quickly as the monolithic. The microservices are manageable services which are easier and faster to develop, as well as easier to understand and maintain. The different microservices do not have to follow one technology as they can be developed in different programming languages. These services are also deployed independently, with no danger of affecting other microservices if one fails or has a any type of bug. Additionally, this opens up the possibility of scaling independently. If one microservice is requested often and by various user at the same time, it can be scaled to meet the requirements with no need of cloning or copying code. The other microservices can remain with the previous configuration with no side effects.

However, microservices also have their own set of drawbacks. Microservices cannot be instantiated by themselves. They need a complex architecture like AWS Cloud Services to be used. These are usually paid services with subscriptions per month. Additionally, most of the time replicated code will be found within the microservices, given they cannot share a codebase. If a file is needed in various microservices and this file changes at a certain point, the file needs to be copied and pasted within every microservice. This can lead to errors when the file was not copied to every microservice using it or there was a mistake on the copied file.

In conclusion, monolithic architecture are easier to start off in a project. It performs quite good in the early stages of a project and most of the big and successful server-side projects which can be seen today started as a monolith [42]. The microservices architecture is a better choice for really complex projects with many dependencies which evolve through time. And according to [41], the use of services specifically designed to deploy and scale microservices reduces infrastructure costs by 70% or more.

Microservices and DynamoDB

As seen in Figure 4.1, microservices are usually connected to their own instance of a database. However, for this project the architecture was changed to force the microservices to connect just to one database. Indeed, this forfeits the idea of loosely-coupled services which are completely independent of each other. At this point, they are dependant on the same database. Nevertheless, this is the best approach for this project given that the digital on-boarding process happens in a linear manner where there is only one way the data is being changed. This means the entry of a given user will not be changed simultaneously by various microservices accessing to the database at the same time. On the contrary, the database will always write to a user's entry one at a time, after every step of the digital on-boarding process is fulfilled by the user. There is no chance of data mismatch or data being overwritten by a different microservice. There can never be inconsistencies on the database unless a person is logged in simultaneously in different devices and goes through the digital on-boarding process.

Even though the risk exists by using this approach, the benefit of having only one database pays off on the state management of the users by delivering the best user experience while navigating through the digital on-boarding process of the mobile application.

Environments

By using the serverless architecture, it is possible to create stacks. A stack is an ecosystem of AWS Cloud services designed in such a way that changes in one stack do not affect changes in any other stack. It is possible to leverage the idea of stacks to replicate the content of all the services together and separate them by environment. In any software development project, it is advised to have more than one environment [26]. For this project, the best approach is to have three different environments: dev, stage and prod. All three environments are exactly the same, have the same capabilities. The only difference is the sensibility of the data saved in each one of the environments.

The dev environment will be used for development purposes, where errors and mistakes can be made on a daily basis. It contains no crucial information and it is meant to handle breaking changes without consequences. The stage environment is used as a previous step before production. On stage, extensive testing is done and the real case scenarios are simulated to prepare for the production environment. Finally, the production environment is the one to be used once the application is open to the public. This one is sensitive and should not be taken lightly when creating changes or pushing new features as it will have a direct impact on users relying on it for the service.

As a measure to protect the production environment, most privileges are disabled. This means a developer cannot delete a record from the database or accidentally remove or stop one of the services offered by AWS on the stack.

In the next subsections, the cloud products used by the backend will be described and their role within the three environments previously mentioned.

Route 53

Route 53 [27] is a Domain Name System (DNS) which is used to route requests to the different AWS services within the stack. Its main purpose is to translate names to IP addresses. For example, `www.example.com` could be translated to `192.0.0.2`. However, as this project is divided between three different stacks, the goal is to use Route 53 to direct the requests to the correct stack. It is configured to have the environment as the prefix of the domain name. This means the environment variable (dev, stage, prod) will be placed beforehand the url to be contacted. The end url will look something like: `https://env.domain.ch`. This configuration is created to simplify the request url to be used in the mobile application. It is much simpler to notice the environment which is being called when reading a url starting with 'dev.domain.ch' as to one automatically generated by the different AWS services: 'https://t5tigER2.execute-api.eu-central-1.amazonaws.com'.

AWS API Gateway

Amazon API Gateway is a service which creates, publishes, maintains, monitors and secures the different endpoints used by the mobile application. It provides the access data, business logic, or functionality from the backend services [29]. The core concept is to create RESTful[30] APIs which will create a two-way communication with the mobile application. The requests will arrive to the API Gateway and a response will be always be delivered. It is equipped with a robust architecture supporting up to hundreds of thousands of concurrent API calls [29] including traffic management and CORS[31] support. It is meant to automatically scale as more requests are needed to be serve and the payment is also consumption oriented, charged only for the amount of requests received.

AWS Virtual Private Cloud

Amazon VPC allows to create a virtual network to launch AWS resources in[28]. This service is used to assign an elastic IP to the API Gateway service. An elastic IP is a static IPv4 address designed for dynamic cloud computing [32]. It is a public address which is reachable from the internet. This service is crucial given one of the requirements of the MVNO described in section 4.3. The MVNO is protected by a firewall which can only be accessed by predefined IP addresses saved within the configuration files of the MVNO. This means one IP address was saved within the configuration files of the MVNO, and every request which lands into the firewall must come from said saved IP address or the request will be blocked and dropped. For example, if a request is originated from the API Gateway, directed to the MVNO to add a subscription to a number, it needs to be originated from the elastic IP address and sent through the network via the VPC.

AWS Elastic Load Balancing

Elastic Load Balancing automatically distributes the traffic across the different AWS services [33]. It offers three different types of load balancers which scale automatically depending on how much traffic will be generated from the users and it provides a robust security to protect the services and make them fault tolerant. The three load balancers are: application, network and classic. The application load balancer manages the HTTPS traffic and routes to the microservices and containers. The network load balancer controls the different network layers such as Transmission Control Protocol (TCP) or Transport Layer Security (TLS) traffic. This one redirects the traffic incoming to the VPC. The classic load balancer is not used by this project.

AWS CloudWatch

Amazon CloudWatch is the a service which monitors the applications and provides reports for the developers[34]. After every request is processed by the different Lambda functions, a log file will be saved within CloudWatch with crucial information about the request just

served. This is useful when developing new features which are interacting with different services and the log is needed to assert the functionality of the method being called. Additionally, it is a useful tool when finding bugs or finding out why a certain action happened. For example, if a Sim Card could not be activated by the MVNO, it is quite useful to open the log within CloudWatch to see what exactly happened with the request to be fixed for the future interactions.

AWS Lambda

Each AWS lambda function is created to run code without a server [35]. It creates different microservices which can be instantiated by requests coming from the mobile application and live in stateless containers. Stateless containers refer to a container with an independent lifecycle, not attached to other microservices or to a parent service. Each lambda function runs independently of the others and even if there is a fault or an error within one of the microservices, the others keep running without interruption. They are also robust and automatically scaled depending on the traffic each microservice requires[35]. To access one of the microservices, a path has to be defined. For example, to access a user information the path could be `/user/id`. This path will be appended to the API Gateway domain (masked by the domain provided by Route 53) and a response will be generated. At the end, a request to a microservice will simply look like: `https://dev.domain.ch/users/id`.

Every Lambda function is declared to be protected by IAM Authentication, which means an identified user with a valid session needs to generate the requests or they will be dropped and ignored by the microservice.

AWS Cognito

AWS Cognito is designed to add user capabilities to an application[21]. It keeps track of the authenticated users and provides mechanisms to sign up, sign in and verification of users. AWS Cognito offers User Pools and Identity Pools. The User Pools are the user directories where users can sign in with the normal combination of email and password. The identity pools provide the AWS Credentials which are needed to interact with different services of AWS. For example, the credentials needed to inject into the headers of a request to be sent to the Lambda functions. Every user created within the User Pools will be mirrored into the Identity Pools. However, users created via a social login (Google or Facebook) are created directly into Identity Pools. There is an exception which is Apple Sign In. Apple Sign In is not yet supported by the Identity Pools so it is only available as a federated identity within the User Pools. This means a user which logs in using Apple Sign In will be added into the user directories of a User Pool and mirrored into the Identity Pools to get access to the AWS Credentials.

Once an a user is authenticated with the Identity Pools, they will be provided with an access key, a session token and a secret key. These are the key pieces to perform a successful authentication to the different AWS services matching AWS Signature Version 4.

Every authenticated user created will be granted a unique identifier which will be directly used by the mobile application and the database to keep track of the user.

AWS DynamoDB

Amazon DynamoDB is a NOSQL [37] database service delivering single-digit millisecond performance at any scale [36]. It is a great service when no relations are needed between tables. The database is accessed to query what is the current step of the user within the on-boarding process. However, reading and writing privileges are only granted to the AWS Lambda functions. This means that the mobile application will send requests to the backend and the latter will be in charge of modifying the state and adjusting it when necessary depending on the user actions.

State Machine

The different states of the digital on-boarding process were enumerated beforehand within the Redux subsection. However, they are handled by the backend by a combination of the Lambda functions and DynamoDB. As soon as a user is registered within the mobile application, an entry will be created within the DynamoDB with the initial state as seen in Figure 4.2.

```
▼ Item {8}
+   __ACTIVATE_SIM_CARD_STATE String : INITIAL
+   __AUTHENTICATION_STATE String : ACCEPTED
+   __CREDIT_CARD_STATE String : INITIAL
+   __KYC_STATE String : INITIAL
+   __ON_BOARDING_STEP String : OFFER
+   id String : 15ab4660-e512-4947-bde1-bef47a392e68
+   killbillId String : da00d508-c51a-4321-8c89-d69600db2c05
+   onfidoId String : 9e56b547-57c1-4cf0-bb20-d0ceblef5e64
```

Figure 4.2: Initial state of a registered user

At the start of the process, the user has a unique id which is provided by AWS Cognito, as well as a unique Killbill and Onfido identifier. The last two will be discussed in subsections 4.2.2 and 4.2.4. What is important to note from Figure 4.2 are the variables preceded by the double underscore ('_'). These variables are the ones controlling where a user currently is within the digital on-boarding process. The mobile application is requesting user information from the Lambda functions every time an action is completed. The new

information contains information regarding the identity verification, the payment method and the activation of the sim card.

As seen in Figure 4.2, a user which is just registered has a state INITIAL for the identity verification, the credit card and the activation of the sim card variable. Nevertheless, as a user keeps advancing within the on-boarding process, the values of the variables will change until the value of `__ON_BOARDING_STEP` is `ON_BOARDED`. This value means the digital on-boarding process was a success and the user will be kept at the last step which is within the carousel of numbers.

The different values the variables can contain are INITIAL, SUBMITTED, ACCEPTED and DECLINED or ERROR. If the identity verification performed by Onfido returns values which might indicate a fraudulent or suspicious person, the value is set to ERROR and the user is blocked within the on-boarding process until the verification is successful. A user can retry the identity verification process as many times as they desire. The same behaviour is applied to the activation of the Sim Card. If the activation is not successful for any reason, the user is also blocked within the on-boarding process given they cannot go forward unless they have an activated Sim Card to add a subscription to. The blocked state can be seen in Figure ??.

AWS Secrets Manager

AWS Secrets Manager is an application which securely stores database credentials, API keys and OAuth tokens in a centralized manner [38]. It is used by the stacks within the AWS Cloud services whenever a request needs to be made to a service which requires authentication. Instead of hard-coding the API keys or tokens from the different values, they are always requested from the Secrets Manager to be used dynamically. This ensures an extra layer of security if the code written on the Lambda functions is ever exposed or seen by malicious parties.

AWS CodeCommit

AWS CodeCommit is a source control service meant to secure Git-based repositories [39]. It is where the code is kept for both the mobile application, Lambda functions and the code configuration for the different AWS services such as Cognito. The repositories can be accessed using any existing Git tools and to merge new features or changes into the code, pull requests are created for an extra layer of quality assurance of the code.

AWS Code Pipeline

AWS Code Pipeline is a continuous delivery service which helps automate release pipelines by automating the build, test and deploy phases of a code change [40]. It is used to track when a commit is made into AWS CodeCommit. As soon as a new commit is pushed to the master branch, AWS Code Pipeline will automatically build and test the

code. Once the test pass, it will also perform a deploy of the service with the new changes. However, it is important to notice that the automatic pipeline runs only for the development environment. Once the development environment has passed all the tests and is deployed, the stage pipeline is unlocked. To trigger the stage pipeline, a manual approval of the pipeline has to be approved. Once the stage environment is build, tested and deployed, the production pipeline is unlocked. The same manual approval has to be done at this point to trigger the production pipeline. These measures are taken to properly test the code before advancing between the different environments and to prevent accidental triggers of the pipeline which could have big consequences in practice.

4.2.2 Killbill Subscription Management

Killbill is an open-source platform in charge of handling billing and payment, it is the backend subscription manager of e-commerce websites [43]. This service manages accounts, adds subscriptions previously defined to the accounts and is in charge of billing them for the start of an entitlement and also for the recurring billing which happens when a subscription is renovated.

The catalog

The definition of the subscriptions is the first step when setting up Killbill as the subscription manager. Killbill uses a file called the Catalog to define which subscriptions it supports and their properties. The catalog is able to define various options for handling trials, add-ons, upgrades / downgrades and cancellations [43].

The catalog defining the subscriptions offered by the mobile application consists of a base product which is the main selling point of the application. The base product can be of two types: a daily subscription or a monthly subscription. The daily subscription is charged 2 Swiss francs at the activation of the subscription and then every 24 hours it is charged again until cancelled. The monthly subscription has no initial charge given the first month is for free. However, after the free month, 42 Swiss francs are charged to the account. The definition of the free trial can be seen in Listing 4.5, where the phase DISCOUNT has a duration of 30 days. This subscription has a price of 0 Swiss francs which must be defined.

```
1 <initialPhases>
2   <phase type="DISCOUNT">
3     <duration>
4       <unit>DAYS</unit>
5       <number>30</number>
6     </duration>
7     <recurring>
8       <billingPeriod>THIRTY_DAYS</billingPeriod>
9       <recurringPrice>
10        <price>
11          <currency>CHF</currency>
```

```

12         <value>0</value>
13     </price>
14 </recurringPrice>
15 </recurring>
16 <usages />
17 </phase>
18 </initialPhases>

```

Listing 4.4: Free trial for 30 days

However, in Listing ??, the normal subscription without a free trial can be seen. The phase, in this case, is EVERGREEN which means it does not have an expiration date. As long as the customer does not cancel the subscription, it will keep running forever. In this type of phase, it is possible to declare a duration UNLIMITED and below the billing period defines how often will this subscription be charged to the customers. For a daily subscription, the billing period is DAILY and has a value of 2 Swiss francs.

```

1 <finalPhase type="EVERGREEN">
2   <duration>
3     <unit>UNLIMITED</unit>
4     <number>-1</number>
5   </duration>
6   <recurring>
7     <billingPeriod>DAILY</billingPeriod>
8     <recurringPrice>
9       <price>
10        <currency>CHF</currency>
11        <value>2.00</value>
12      </price>
13    </recurringPrice>
14  </recurring>
15  <usages />
16 </finalPhase>

```

Listing 4.5: Normal subscription with recurring billing

Additionally, there is also another type of subscription which is an add-on to the base subscription. The add-on refers to an international subscription which enhances the daily or monthly base subscription to have service outside of Switzerland. This subscription is also of the phase type EVERGREEN and is charged 3 Swiss francs on a daily basis.

Child Accounts

The business requirement for this project is to have a registered user with up to 5 different numbers at the same time. Each number can have an active Base subscription and attached to that subscription a possible international subscription active as well. To solve this problem, the best approach is to create child accounts. A child account is an account linked to a parent account by a unique identifier. Additionally, the child account may use the parent account payment method to pay for the active subscriptions.

At the beginning, it was simple enough to think that a registered user could have all the information stored within the parent account and each number attached to the account could be a child account. In theory, this approach works for most of the cases given each child account could have its own Base subscription and when needed an international subscription as well as an add-on. However, it turns out that in practice, it is not so straightforward to maintain subscriptions where the users can decide to cancel at any time and activate again when needed. The best solution is to create two child accounts per number, one to hold the base subscription and a second one to hold the international add on when there is an existing base subscription. On the next subsection, it will be clarified why this approach was the best one when dealing with subscriptions which can be cancelled and activated again at any time by the users.

Recurring billing periods

After extensive testing and usage of the subscriptions managed by Killbill, it was noticeable that the recurring payments were happening always at midnight and grouped together. For example, a daily subscription was activated at a certain day at 17:00 o'clock. At that point, one payment was triggered and a user has 24 hours of service on their activated number. However, the next payment would not happen at 17:00 o'clock on the next day. Instead, it would happen at 23:59 o'clock. Just a bit before midnight. Killbill uses this approach to align all the subscriptions to the same billing period. In other words, all the subscriptions that a parent account might have and the subscriptions from all the child accounts will be aligned to midnight and charged simultaneously. At this point, one big problem was already clear: the user would have mobile service until 17:00 o'clock the next day and no service until the next payment comes which is at midnight.

This problem was, at first, tackled by giving users the extra time for free. Instead of activating the user until 17:00 o'clock, it was activated until 23:59 o'clock for no extra charge. Nevertheless, this approach was soon discarded given another problem appeared with the international subscription. As both, the Base and the international subscription were being aligned together on the billing periods, the payments were grouped together. What was even a bigger problem was with multiple numbers with different subscriptions, the payment was always grouped together and it was impossible to differentiate for which subscription exactly was the user charged for. This is not an acceptable case given the Swiss law requires a subscription to be paid separately or correctly described within a subscription tree which was not possible to deliver.

The solution to this problem was add to the child accounts their own payment method, separated from the parent account. Then each child account would pay for its subscriptions and would not be aligned with the parent account billing period. Nonetheless, the subscriptions would still be renewed at strange periods of time. For example, if a user activated daily at 13:00 o'clock and an international subscription at 16:00 o'clock, the user would get charged 5 Swiss francs (2 Swiss francs for the daily subscription and 3 for the international one) at 10:00 o'clock. After extensive testing, it was noticed that the subscriptions were being renewed at the same time when the account was created. The default billing alignment set by Killbill for this type of accounts is the same as the account creation time. To solve this problem, it was necessary to close the child account

and open a clone of it at the time the base subscription was activated. For example, if a user decided to activate a daily subscription at 13:00 o'clock, then at that moment the child account would be closed and a clone of it would be created to have the recurring billing period aligned with the subscription activation time. By following this approach, the daily subscription would now be charged the next day at 13:00 o'clock as expected.

At this point, it seemed that the problem was finally being solved. However, after extensive testing, it was noted that the international subscription would still be aligned to the creation time of the account. For example, if a base subscription was activated at 13:00 o'clock and an international subscription activated at 16:00 o'clock, both subscriptions would be renewed the next day at 13:00 o'clock. This means the user could not use the last 3 hours of their international subscription and also the payment for both subscription was also grouped as only one payment. Therefore, another solution was needed to solve the last problem.

The final (and the implemented) solution is to have two child accounts per number. One child account holds the subscriptions for the base account and the second child account the subscriptions for the international option. If a base subscription is activated, the child account for the base subscription will be closed and a clone of it will be created with the active subscription. The same procedure is done when activating an international subscription. By having both subscription in different accounts, the payments remain separate and the recurring billing period is never aligned. Each subscription has it's own lifecycle with the correct time renewal after exactly 24 hours. The pairing of the child accounts is done by having the mobile number as the name of the child account holding the Base subscription, and the name of the child account holding the international subscription is the unique id from the first child account.

Subscriptions activation and cancellation

4.2.3 Swiss Post Free Address Verification

Swiss Post has a free address verification online web service which can be accessed by any registered Swiss Post business customer [44]. The web service is quite old and it uses SOAP [9] to communicate between applications. This means it is necessary to send a XML document with the information the user has written down for the address and wait for a response to know if it is a valid address in Switzerland.

The XML document usually looks like this:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
   envelope/" xmlns:v4="http://post.ch/AddressCheckerExtern/V4-02-00">
3   <soapenv:Header />
4   <soapenv:Body>
5     <v4:AdressCheckerRequest>
6       <Params>
7         <MaxRows>100</MaxRows>

```

```

8      <CallUser>TU_1560433_0010</CallUser>
9      <SearchLanguage>1</SearchLanguage>
10     <SearchType>1</SearchType>
11  </Params>
12  <!--Optional:-->
13  <Names>Felix Muster</Names>
14  <!--Optional:-->
15  <Street>Viktoriastrasse.</Street>
16  <!--Optional:-->
17  <HouseNbr>21</HouseNbr>
18  <!--Optional:-->
19  <Onrp>0</Onrp>
20  <!--Optional:-->
21  <Zip>3013</Zip>
22  <!--Optional:-->
23  <Town>Bern</Town>
24  <HouseKey>0</HouseKey>
25  <!--Optional:-->
26  <PboxAddress>0</PboxAddress>
27  <!--Optional:-->
28  <PboxNbr />
29  </v4:AddressCheckerRequest>
30  </soapenv:Body>
31  </soapenv:Envelope>

```

Many fields are optional. However, a combination of this fields will return a correct response from the Swiss Post API.

The response will contain key information about the provided address. If it is not a valid address in Switzerland, an error is returned to the user. If the name is not registered under the provided address, the backend looks for the last field which may contain extra information. If the last field is blank, an error is returned asking the user to fill in the extra information given the name could not be found under the address. This is the case when an international person is visiting Switzerland for a short time. The person can write down the address of their hotel and their name, but the Swiss Post API will not be able to find the name under the hotel's address. In this case, the name of the hotel would have to be written in the last field, providing extra information for the API.

It is important to note that the Swiss Post address verification API will change by October 2020. A new system is being built which uses REST [10] to communicate with applications. This new system will also support address verification for free, but not name verification. The name verification will become a paid service charged per verification.

4.2.4 Onfido Identity Verification

As stated before, a third-party service is needed to verify the identity of the users. After gathering information and checking out which options are available in the market nowadays, the best match for the application is Onfido [11] given it's verification speed and availability. Onfido is a software company based in the United Kingdom, a company which

offers services like identity verification, user on-boarding, fraud detection or age verification. They use a combination of different pillars to verify an identity and the verification usually takes less than 5 minutes. The cost per verification is 2.50 Swiss Francs. Onfido is able to verify identities from all over the world by using either the passport, identity card or license card of the person.

To use the Onfido API three steps are necessary:

1. First, the backend needs to create an applicant within Onfido by using the API token provided previously by Onfido. When the applicant is created, an SDK token can be requested.
2. Second, the native library in both Android and iOS must use the SDK token to instantiate itself and be presented to the user. Within the native library, the user can take a picture of their passport or identity card and a selfie.
3. Third, the backend awaits a request from Onfido as a webhook [45]. The request sent from Onfido will contain information from the applicant as their id, name and verification status.

The native library screens can be seen in figure 4.3.

Once the request from Onfido is received, the backend is then in charge of parsing the information contained within the body of the request. A handful of parameters are delivered, each of them with crucial information from the identity verification. All of the parameters have a status. If the status is "clear", it means there were no problems with the verification of the parameter. The status "consider" means there was a problem or an issue while verifying the parameter.

The parameters sent by Onfido are:

1. Facial similarity report
2. Document report
3. Compromised document
4. Police record
5. Data consistency
6. Visual authenticity
7. Image integrity
8. Data validation
9. Age validation

If any of these properties return with a status "consider", then the identity verification is set to DECLINED and the user must repeat the process again. They would not be able to keep advancing further within the application until the identity verification is successful.

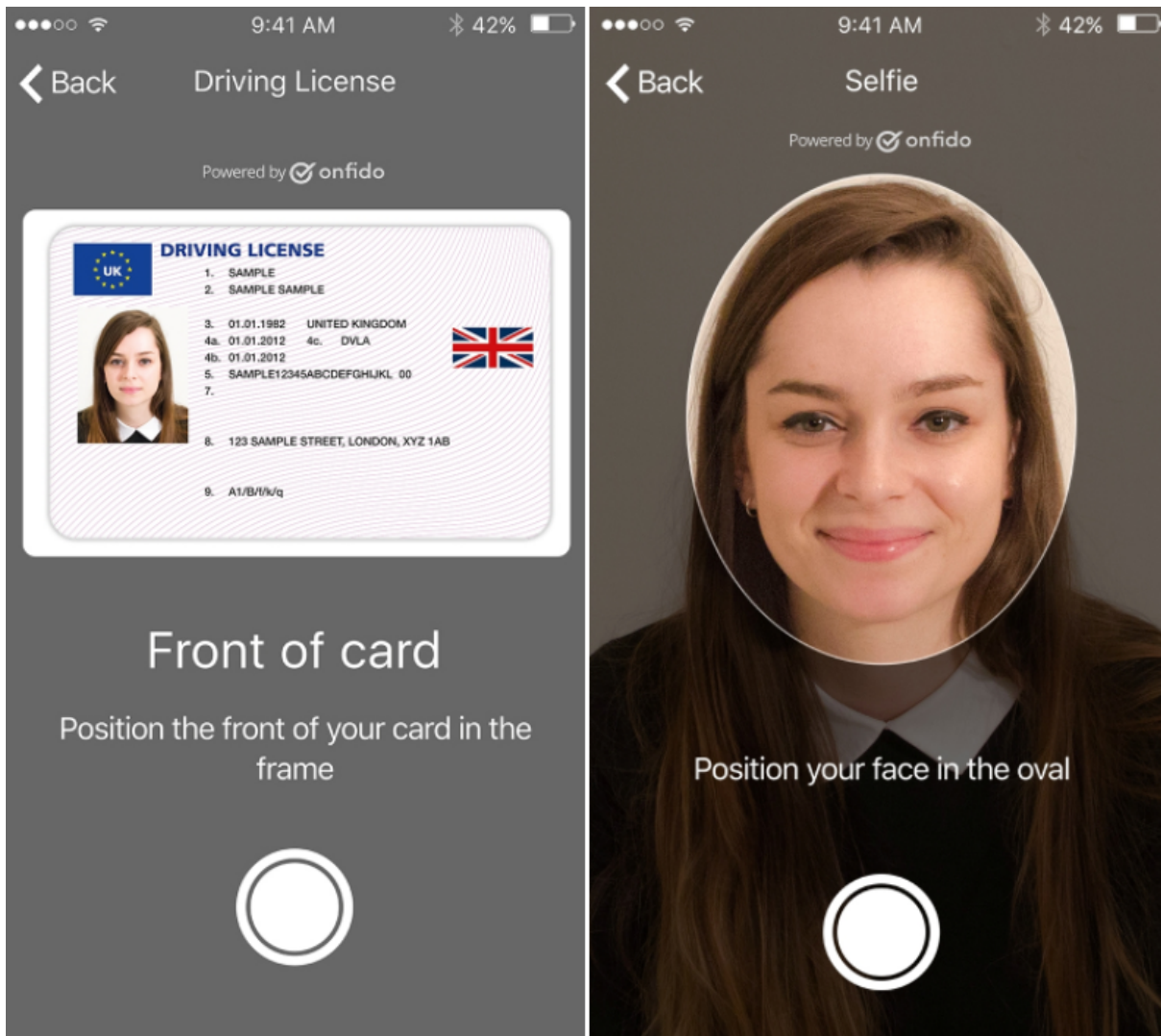


Figure 4.3: Onfido identity verification

4.2.5 Funke Delivery System

Funke Letter Shop [46] is a Swiss company specialized in delivering letters. They are in charge of delivering the Sim Cards to the users when requested, offering a maximum delivering time of 24 hours. Their service is straight-forward. An excel file containing all the addresses where the Sim Cards are to be delivered needs to be uploaded to their ftp server. At the end of the each day, Funke will take this file and ship the Sim Cards to every address the next morning. The backend is in charge of uploading the excel file to ftp server owned by Funke.

4.2.6 Datatrans Payment Service Provider

To fulfill the business requirement of paying for a mobile subscription directly via the mobile application, a third-party service is needed. As explained before, part of the

requirement is to pay using Twint. Exactly for this reason, it is necessary to find a Swiss payment provider with the ability to connect to Twint as a recurring payment. The only service available in the market at the moment which fulfills these requisites is Datatrans [13]. Datatrans is the service which creates a check against the most common credit card brands or Twint to see if it's a valid credit card or account with sufficient funds to create charges. If Datatrans validates the payment method chosen by the users, it is possible to be sure money can be charged to that payment method at any time. To keep users within the application and to keep the mental flow of the on-boarding process, the only option is to use the native libraries provided by Datatrans to integrate into the mobile application.

This is a good idea for two main reasons:

1. If the native library is used, it is possible to create input fields for the credit card number, expiry date and cvv styled according to the mobile application. If a hosted UI is used or a redirect to the phone's browser, Datatrans will render their own content, layout and styles for the customer which breaks the mental model of the on-boarding process.
2. By using this method, the application is never sending sensitive data through the network and it's not prone to sniffing attacks where the credit card data can be obtained by malicious parts. All the credit card verification occurs within the native libraries and Datatrans handle the security of data. What is important for the mobile application side is to never store the credit card data in any way, just pass it through to the native library developed by Datatrans.

The credit card input fields can be seen in figures ?? and ??.

4.2.7 Zendesk Customer Support

To provide users with support during the whole on-boarding process, a customer support feature should be added to the mobile application. Zendesk [47] is a customer service software offered by Zendesk Inc. Zendesk provides a user-friendly interface to create categories, sections and articles. This interface is used by the mobile application support team to fill in with data concerning the brand. For example, articles may answer questions as where can the service be used, how long does it take to activate a Sim Card or what to do when the Sim Card has not yet arrived. This interface is meant to keep the information dynamic and easy to edit. On the mobile application side, the information is fetched from the Zendesk Support API and presented to the users when needed.

The information is divided as the following:

1. Categories: The high level category. For example: account, payments, Sim Card.
2. Sections: 4 sections, each one corresponding to English, French, German and Italian.
3. Articles: Here is the written information meant to be helpful to a user when going through the on-boarding process.

The backend is in charge of requesting the information from Zendesk API and parsing it in a nicer form to be sent back to the mobile application. The category overview can be seen in figure ??.

4.3 MVNO

As explained by [48], a mobile virtual network operator (MVNO) is an extension from other telecommunication supplier. An MVNO owns no infrastructure. Instead, it resells the service in a virtual manner without a underlaying network. This is essential to minimize costs and allows smaller companies to sell these services as a cheaper alternative than the companies with the infrastructure.

To offer competitive prices on the daily and monthly subscriptions seen on the mobile application, it is needed to partner up with an MVNO. Sunrise offers their infrastructure as a service to smaller companies in the form of a software called Provisioning Portal (PP) [49]. PP will then be in charge of the following steps:

1. Produce the Sim Cards and deliver them to Funke.
2. Activate the Sim Card and give the user a phone number.
3. Activate daily, monthly and international subscriptions.
4. Block unblock a Sim Card
5. Activate deactivate a the voicemail of a phone number.
6. Provide information of a phone number as their pin, puk, expiration date, voicemail status, subscriptions status.

As PP is a software developed by Sunrise, it has a few security measures which have to be taken into account while trying to connect to the backend system. Requests sent to PP have to come from a whitelisted IP, which has to be previously saved in the firewall protecting the endpoints. The IP has to redirect to a predefined host name which also needs to be saved within the firewall. As PP is not a new software, all requests have to be sent as SOAP requests. An example of a request to activate a Sim Card can be seen here:

There are a few variables in this request such as the msisdn which is the phone number provided by PP. The iccid which is the 20 digits code found on the Sim Card provided by PP. Finally, the language and the date of birth of the user trying to activate the Sim Card.

Every request to PP must be created by the backend given the firewall restrictions imposed by Sunrise.

Given most of the PP endpoints are asynchronous, no concrete response is received after sending a request. By using a second request, it is possible to find out the status of the

first request. For example, while trying to activate a Sim Card, PP will return only an order id. By using this order id in a second request, it is possible to know if the Sim Card has already been activated or not yet. For this asynchronous restriction, it is needed to implement waiting screens on the mobile application.

Extra screens have to be created to inform the user that their action was triggered successfully; however, it is still being processed. At this point, the mobile application will send a request to the backend server every 7 seconds asking the status of the order. Once the backend responds with a status different than "NOT PROVISIONED YET", the mobile application will redirect the user to the corresponding screen. An example of this behaviour can be seen in figure ??.

It is important to note that given this requirement of asynchronous requests to PP, it is no longer viable to use the local environment to develop the backend system. Given another Lambda function has to be called with the information from PP, this is not possible on the local environment and, therefore, the only available environments are dev, stage and prod.

4.4 End-to-end architecture and API calls

Figure ?? shows an overview of the whole architecture used by the mobile application and backend server. The main points of the figure are:

1. The mobile application is only directly connected to Onfido and Datatrans given the necessity of the native libraries. Besides those, the main and only connection of the mobile application is to the backend server.
2. The backend server is connected to the Provisioning Portal (PP), Funke delivery system, Zendesk API, Swiss Post Address Verification API, Killbill API and Onfido API.

Chapter 5

Evaluation

This chapter presents the evaluation of the designed mobile application. The evaluation is focused around the prototype built to show the digital transformation of the whole process of getting a new mobile subscription in Switzerland. Firstly, statistics from the prototype are presented as well as the results from a questionnaire concerning its user acceptance. Secondly, the backend usage is described and AWS is evaluated by analysing the costs per service from December 2019 to March 2020 to find out where exactly the resources are allocated and if they can be improved. Thirdly, findings from the mobile application, which were not anticipated before deploying the prototype are explained and mentioned. Lastly, a case study is presented where the prototype of the mobile application was designed, implemented, built and launched.

5.1 Mobile application launched for Android and iOS

The mobile application was developed targeting both Android and iOS operating systems. The first version was released on December 10, 2019. It is available for free to every person with a mobile phone running an Android or iOS operating system.

The release included a connection to Firebase [66], which is a service offered by Google capable of recording data from the mobile application in both Android and iOS operating systems. Firebase offers real-time data of active users, installs and audiences for each platform. It also provides information regarding application crashes and version statistics.

5.1.1 Application statistics

The mobile application was downloaded, as of the moment of writing this thesis, a total of 21037 times from December 2019 to the end of February 2020. From this group, 32.7k were sessions (one session equals one time the mobile application is opened) recorded within Switzerland. This is understandable given the entirety of the marketing and publicity created for the mobile application was released within Switzerland.

From this group of users who have already downloaded the mobile application, 88% are Android users and only 12% are iOS users, as seen in figure 5.1. The large amount of Android users can be explained by three reasons:

- The goal of the mobile application is to offer a new and cheaper mobile subscription in comparison with other mobile providers. Android devices are cheaper than iOS devices. Thus, Android users are prone to find and investigate new alternatives to their current service. iOS users do not typically worry about the price [75]. Instead, they put value on service and commodity as explained by PCMagazine [76].
- The marketing for the mobile application was targeted to people who are not in Switzerland for long periods of time (e.g., more than one month). For instance, business people who might need a mobile subscription for a few days and not for an entire month.
- Android devices offer the capability of having two Sim Cards active at the same time in one device [74]. Users with Android devices who are aware of these capabilities might be interested in purchasing a second mobile subscription which can be used internationally.

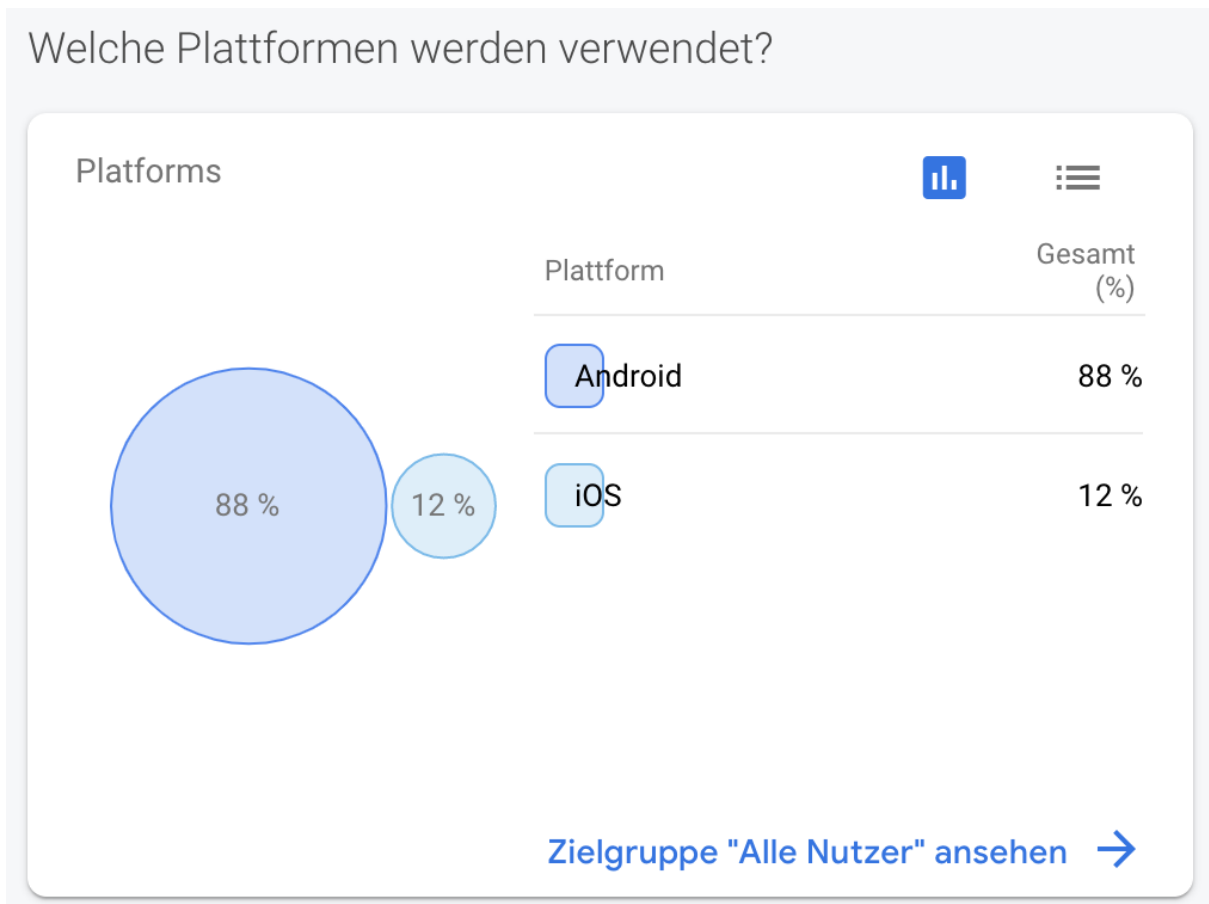


Figure 5.1: Platforms being used

Even though the numbers of installs (21k) is impressive, it does not mean those are active users of the mobile application. These are users who downloaded and installed the mobile application. If only the Android part is considered, 18.4k users have installed the mobile application. However, from those users, 12k have already been lost. A lost user means a user who has already uninstalled the mobile application or has not opened it for more than 30 days in a row. This means users registered and saw the possible subscriptions the mobile application offer and their prices. After looking into this information, they uninstalled the application or never used it again.

5.1.2 System Usability Score

To provide an overview of how usable the prototype published in December is, the System Usability Scale (SUS) [68] metric was used in the form of an online questionnaire sent to users on February 2020. SUS is a reliable metric developed in 1986 which provides insights for measuring usability [68]. Even though it was published in 1986, it still attracts attention given it evaluates a wide variety of systems in a very accurate way. It has references in over 1300 articles and publications [68]. Additionally, a template is provided which users can choose options varying from disagree to agree in a scale of 1 to 5 [69].

As explained by [69], usability is context dependant. Thus, it is not possible to qualify usability in an absolute sense. Rather, it is possible to combine the system within the context in which it is going to be used and analyze how users perform with the mobile application. The goal is to measure three items as specified by [69]:

- effectiveness (the ability of users to complete tasks using the system, and the quality of the output of those tasks),
- efficiency (the level of resource consumed in performing tasks)
- satisfaction (users' subjective reactions to using the system).

Also it is important to choose a variety of users who are going to evaluate the mobile application. According to Pelloni [70], it is possible to divide the users groups into categories, depending on the stakeholders of the project. For the mobile application, three categories of users have been chosen divided in the following way:

Developers

- Developers are selected as people who understand technology, have a computer science background and understand user interfaces quite easily. It has been selected developers working in the industry, as well as, computer science students in the last year of their studies.

Quality Assurance / Testing

- Quality assurance / Testers are considered people who do not have a computer science background, but understand the functional requirements of the mobile application and how the user experience should look like.

Stakeholder / Brand Manager

- Brand managers are people imposing the functional requirements for the mobile application with no technical background. Instead, the majority of them have a business / financial background.

The SUS questionnaire was responded by a total of 14 users. The instructions for the questionnaire were quite straightforward and use to understand:

1. To complete this form, you must have downloaded the Swype application and used it at least once
2. All items should be checked
3. If you feel that you cannot respond to a particular item, then you should mark the centre point of the scale

Given the questionnaire was sent at the end of February 2020 and the mobile application was released in December 2019, the users had at least some experience with the mobile application and could respond to the questions with no major difficulties.

As explained by [69], each question gets a raw SUS Score. After every question has been scored, a final score is obtained by multiplying the result by 22.5. The average of all the scores will be a number between 0 and 100 which yields insights of usable the mobile application really is.

The raw and final scores can be seen in table 5.1.

The average of the final scores yields a result of **86.78**. By having such a score, it is possible to assert that the mobile application is very usable within its context. In comparison with 500 other software systems studies, the average SUS score for those systems was 68 [73]. Anything above 68 is considered to be above average [68].

The number of participants for each survey can be seen in table 5.2. The largest group pertains to Developers given the survey was forwarded to a group of developers working professionally and Computer Science students.

From this group of people, the age range was from 18 to 44 years old. All of them had used the mobile application at least once and 9 people out of 10 had successfully completed the digital on-boarding process.

User	Raw Score	Final Score
U1	34	85
U2	38	95
U3	39	97.5
U4	38	95
U5	32	80
U6	31	77.5
U7	32	80
U8	33	82.5
U9	38	95
U10	33	82.5
U11	33	82.5
U12	30	75
U13	37	92.5
U14	38	95

Table 5.1: Raw and Final Scores for SUS

Group	#
Developer	10
Quality Assurance / Testing	2
Stakeholder / Brand Manager	2

Table 5.2: Participants SUS Score Survey

5.1.3 User experience questionnaire

A questionnaire was sent out by the mobile operator to people using the mobile application. The questionnaire was responded by 92 users, all of them being active users of the mobile application and had at least one active subscription.

It is important to note that the users responding the questionnaire are end-users and not developers. End-users do not know, for example, how to answer questions related directly to user interfaces or software design. Instead, the questions were designed in a broad/open-ended manner. Additionally, end-users have only one goal, which is to use the mobile application. If their goal was reached, it is typically considered a good application. Especially, if the product offered by the mobile application is not up to their standards, they will deem the mobile application as "not good enough". In addition to the information previously mentioned, every question within the questionnaire was optional.

Within figure 5.2, it is possible to observe the variability and dispersion of the data gathered from every question. It is possible to identify how the data from the scores is spread. It is also possible to verify for each question the outliers, the minimum and maximum score and where the median is. It is important to notice, for each of the responses, in which percentile the data is shown.

Question 1 contains no outliers as the data is equally spread. However, for question 2,

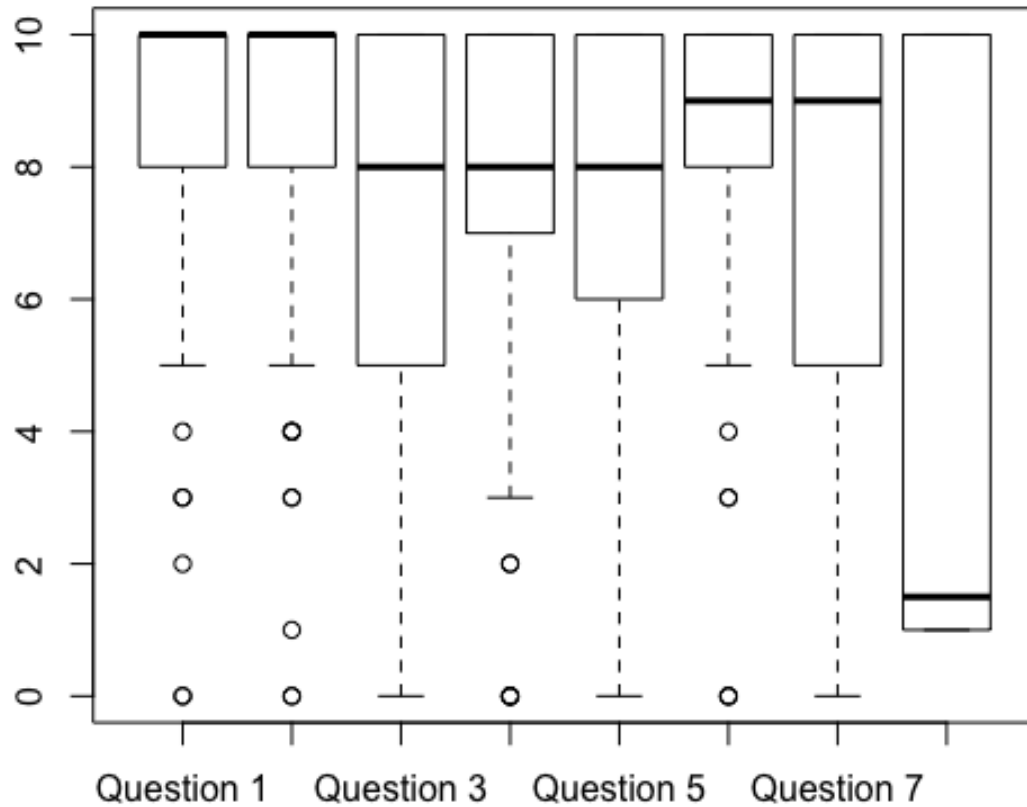


Figure 5.2: Boxplot Questions 1-10

3, 5 and 7, it is possible to see many outliers. Additionally, the majority of answers for those questions is approaching a score of 10. For questions 4 and 8, the median is quite high as well, reaching from 8 to 10. In overall, it is possible to see that every question had a good score report except for question 8 which will be discussed in detail further down.

Question #1: Effort needed to activate a Sim Card?

The first question asked was how much effort was needed to use the mobile application and activate a Sim Card which can be seen in figure 5.3.

The majority of people answered with a 10, meaning the mobile application was easy to use and to move forward within the digital on-boarding process. A 10 also means they could activate the Sim Card and use it without major troubles. A small group of people responded with a 4 or less, only 8 people out of 92. This hints the mobile application

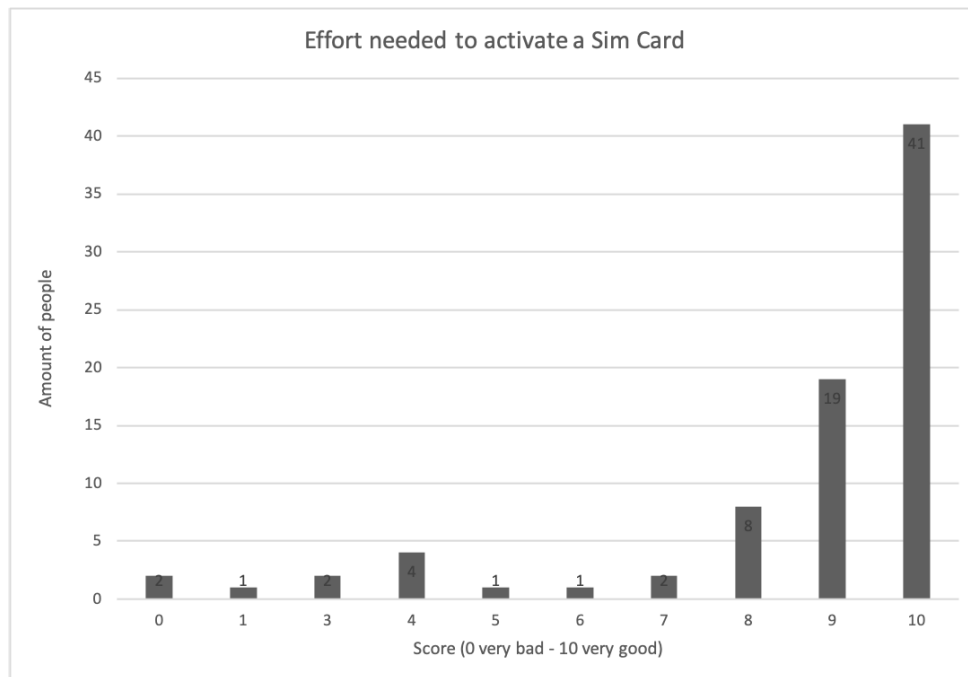


Figure 5.3: Effort needed to activate a Sim Card

will not be liked by every person using it but the majority will find it easy to use and to understand.

The users argued on their responses that the mobile application runs smoothly on their mobile devices, the interface is quite intuitive and they were not lost during the digital on-boarding process. They also said that new features were discovered in every screen, which added value to their experience and overall satisfaction with the mobile application. The users who responded with the maximum score said they were happy no errors were encountered during app-usage and it never crashed.

Question #2: Duration until the Sim Card was delivered?

The next question was designed to find out whether the delivery service of the Sim Cards were fulfilling their promise of one day delivery. The question asked if the users found the duration until the Sim Card was delivered acceptable or not as seen in figure 5.4.

The majority of users stated the delivery times were good enough and they did not have to wait more than one day to receive the Sim Card. This information is valuable given it provides feedback on one of the services being used by the mobile application and their delivery rates. The users who has a bad experience with the delivery times said they ordered a Sim Card before the weekend or the Sim Card did not arrive and they had to order a new one. It is important to note that the delivery system for the Sim Cards is a new system, not used before for a task like the one needed in the application. Issues with the Sim Card delivery was expected at the beginning of the mobile application lifetime.

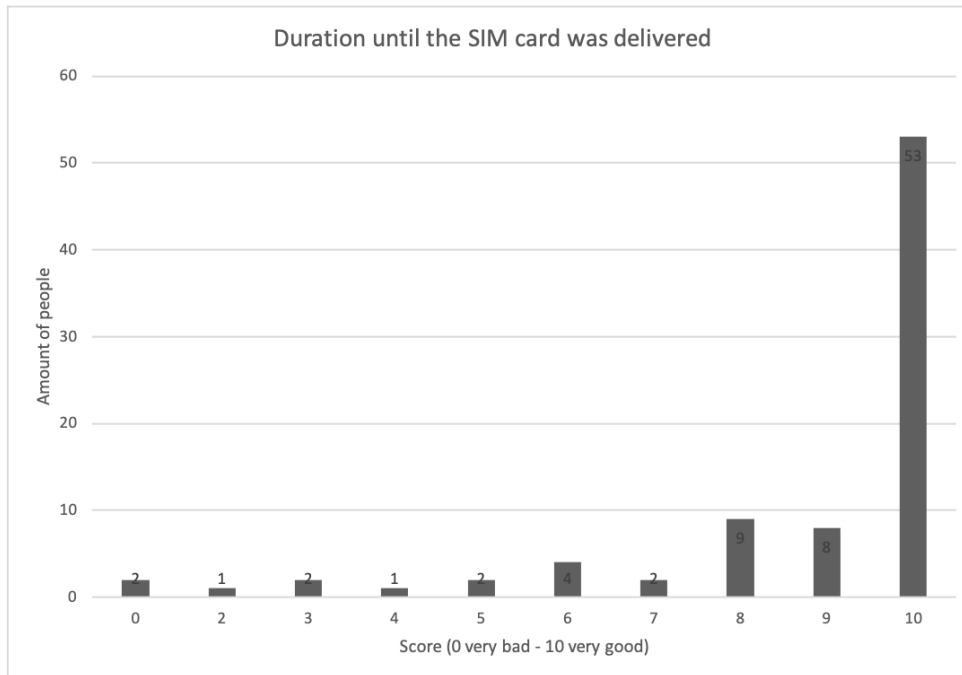


Figure 5.4: Duration until the Sim Card was delivered

Question #3: How satisfied are you with the Price / Value ratio?

The third question was intended to discover whether customers are satisfied with the current price of the subscriptions given what is included in each subscription. The answers for this question were quite positive. More than 62% of the answers are above 5 as can be seen in figure 5.5. The majority of users think the prices are okay for the subscriptions they are receiving. However, they think as well that the price could be improved even more to be more competitive in the market.

Question #4: How satisfied are you with the Mobile network coverage?

This question was meant to find out if the network coverage is delivery a service of quality to the customers or it could be improved even further. The majority of users (scores from 8 - 10) agreed that the network service is quite good and fast as displayed in figure 5.6. However, the rest of the users who responded with a 5 or less argued that the service is not so good outside of the main cities of Switzerland, as soon as the user is travelling to another city the network coverage is lost.

Question #5: How satisfied are you with the speed when surfing or downloading data?

Question #5 is an extension of question #4. The majority of the users were satisfied with the speed while surfing or downloading data within Switzerland as can be seen in figure 5.7. They mentioned that the surfing speed while being outside of Switzerland

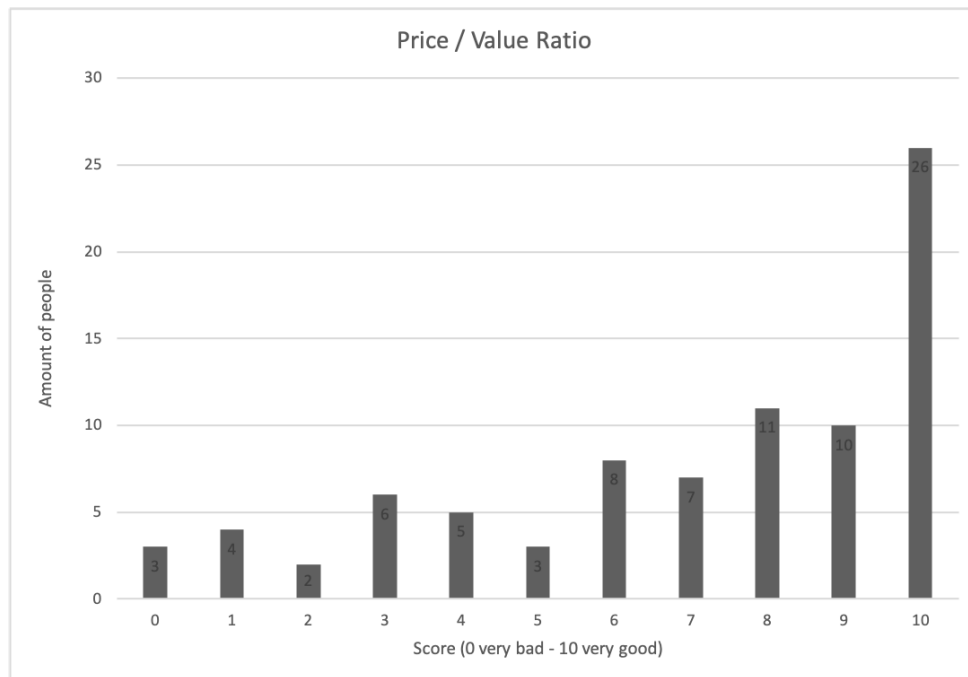


Figure 5.5: Price / Value Ratio

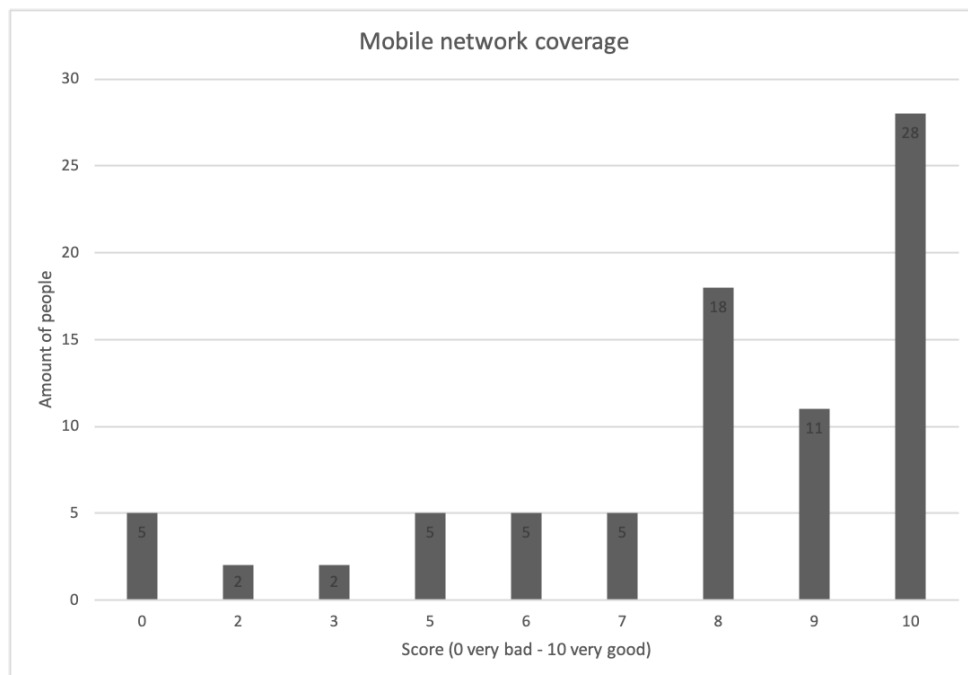


Figure 5.6: Mobile network coverage

is quite slow and could be improved to have a better experience. It was argued that better contracts with mobile operators outside of Switzerland could be achieved to offer customers regular download speeds and not capped ones.

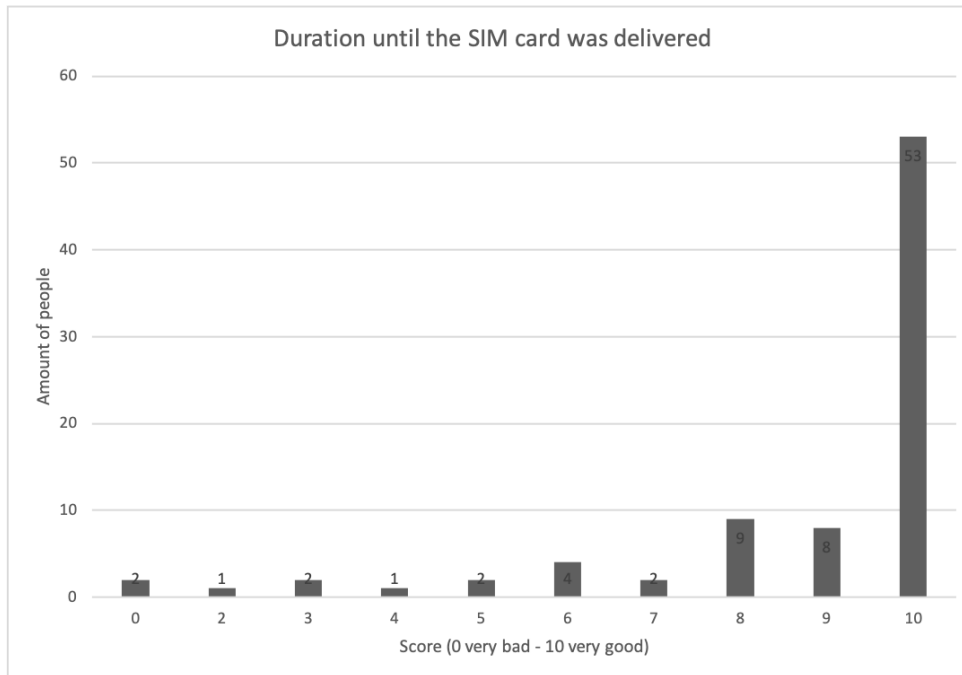


Figure 5.7: Duration until the Sim Card was delivered

Question #6: How satisfied are you with the voice quality?

Given a notorious offer from the subscriptions is unlimited voice calls to other phone numbers, the quality of the calls is of quite interest to the customers. Additionally, voice calls can also be performed through social applications such as Whatsapp [67]. 56 users responded with a score over 8 and just 12 responded with a score from 7 to 1 as seen in figure 5.8. This was a great achievement for the mobile operator given quite some effort was spend in improving the land lines for telephone services. Users said that even while traveling by train from one city to another, they could always call and receive calls with no problems at all.

Question #7: How satisfied are you with the customer support during the digital on-boarding process?

Question #7 was interesting given customer support is important to a user, especially when they ran into a problem. Customer support needs to be able to find solutions, provide feedback and answer any questions related to the service, subscriptions or mobile application. In this regard, the users were not satisfied with the customer service and they argued it could be greatly improved. The majority of respondents have a good score (8-10). However, these scores come from users who had no issues at all and did not have to contact Customer Support. The interesting data is from the users who qualified the service from 0 to 4 as seen in figure 5.9. These users argued customer support within the mobile application was not helpful and only the direct communication via Whatsapp was the solution to their problems. They said they did not bother anymore to read what is

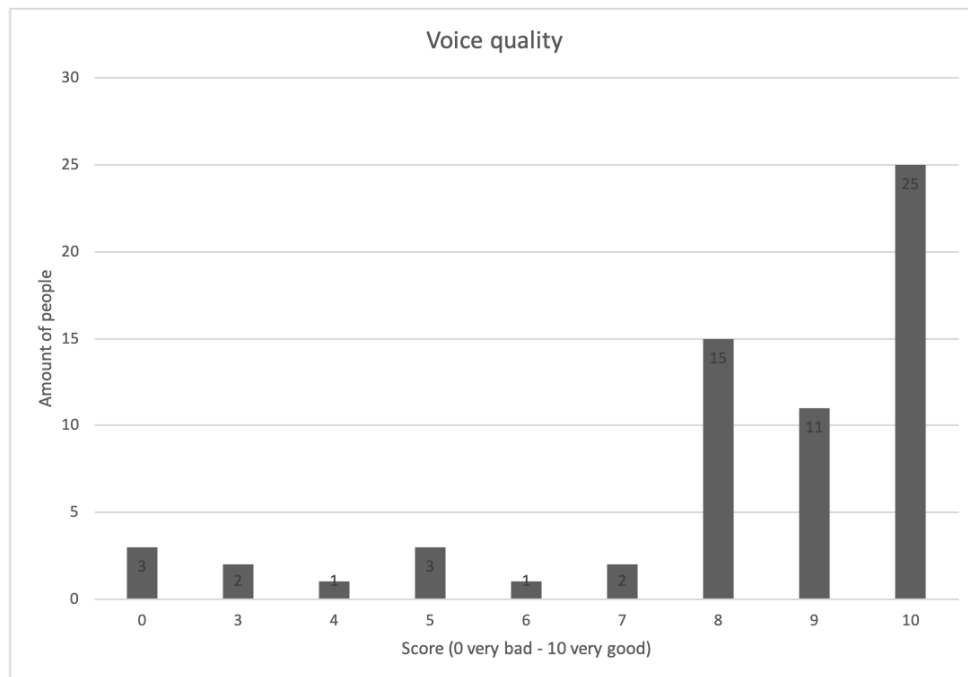


Figure 5.8: Voice quality

written within the app. Instead, they go directly to the Whatsapp chat to ask questions or find out more information from the services.

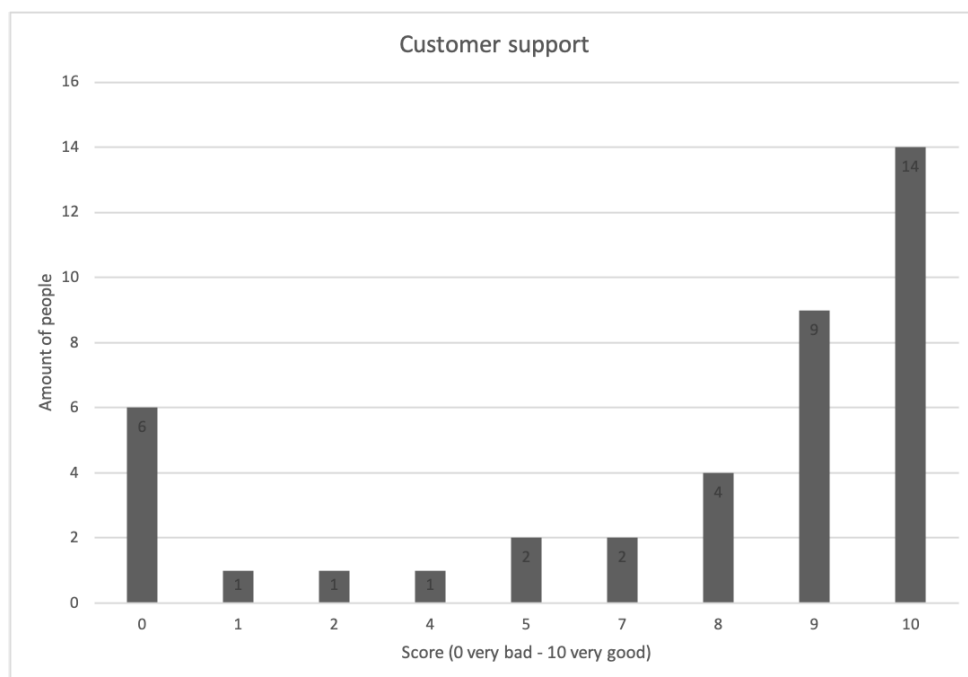


Figure 5.9: Customer support

Question #8: Would you recommend the mobile application to your friends or colleagues?

The data gathered from the last question was a surprise. The majority of users (46%) responded with the lowest possible score: 1. The rest of the users are spread until score 10 which has 32 users as seen in figure 5.10. The outcome of this question was answered in an open-ended question within the questionnaire. The users said it is a innovative idea but many features are still missing. They cannot recommend a mobile application to their friends or colleagues until it is functional in every aspect. Further, they argued critical functions such as switch an old number to the mobile application is still missing. As well as taking a number from the mobile application to another mobile operator. Additionally, services such as Electronic Sim Card is not yet available but they read on the publicity and advertisements that it was. The users discussed that for a first release, the mobile application has quite the potential to grow and disrupt the market. However, it still needs time and more work.

As with another direction, the users who said they would definitely recommend the mobile application to their colleagues were users fascinated by the user experience offered by the application. They enjoyed not receiving any papers to sign and how fast it was to activate a subscription. These users all agreed on the fact that the mobile subscriptions are a interesting offer while travelling abroad or for to recommend to their friends from other countries when they come to visit Switzerland.

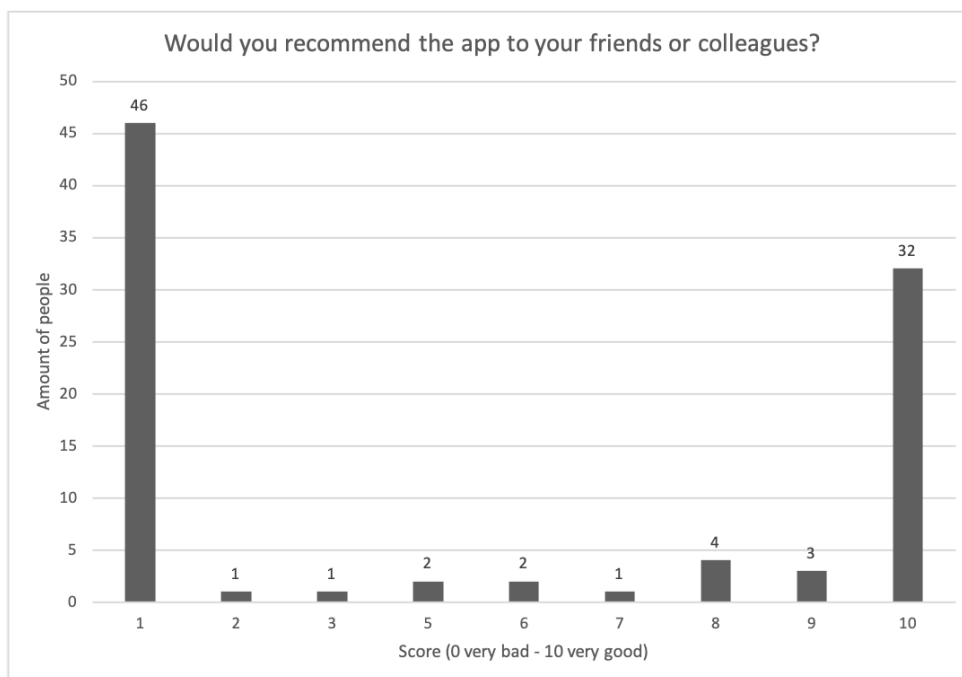


Figure 5.10: Would you recommend the mobile application to your friends or colleagues?

5.2 AWS Usage

AWS is characterized by its price calculations which are based on *Usage plans*. AWS intends to provide the least amount of resources to run a project, only growing or scaling whenever needed. In other words, the more AWS services are used, the higher the monthly cost needed to pay for the infrastructure. However, AWS provides one year of free services called the free tier [72]. This period includes a wide variety of services completely free, others with a free trial and a small, optimized servers at the normal price. This approach is attractive for companies given there is no possibility of paying too much at the beginning due to inexperience of developers or mistakes in deploying the infrastructure [70].

5.2.1 Description of costs for the first three months using AWS

The mobile application was released in December 2019. In the following table, the costs of AWS are detailed from December 2019 until February 2020.

Whenever the cost is incremented by using AWS, it is intuitive to think it costs more because more and more users are downloading the mobile application and a bigger quantity of requests are being sent to the infrastructure. However, this was not the case given the amount of requests sent to AWS such as API Gateway and Lambda were almost the same every month and they are still within the free tier offered by Amazon. The total amount of requests received can be seen in table 5.3.

Period	Requests received (Lambda)	Cost (\$)
December 2019	164.586	\$0.00
January 2020	340.873	\$0.00
February 2020	394.317	\$0.00

Table 5.3: Requested received and processed by AWS Lambda from December 2019 to February 2020

Indeed, the requests received during not so far apart. Only a difference of 44000 requests were received and they both had a cost of \$0 US dollars per month. As a result of this analysis, the cost of the infrastructure was not growing because of the requests received and processed by AWS. Instead, it was growing because of the saved data of the users.

On figure , the services which are consuming resources per month can be seen. It is clear that the services EC2 Container Service and Relational Database service have the higher costs from all the other services. By far, the highest costs. The service using this two services is the subscriptions manager Killbill. This means that the container running the Killbill instance was receiving so many requests that AWS decided to give it full power and let it grow. Additionally, Killbill uses intensively it's database for any type of query. Especially, costly queries such as writing. Therefore, the cost of the Relational Database service spiked and reached over US Dollars in January. Something was clearly needed to be done with these two services.

The instance for Killbill was re-scaled and optimized. Not so many CPU cores were given to the virtual machine holding the instance and the memory was also reduced. It was reduced to the point where it was almost always crashing and from that point, it was upscaled again to fit the needs of customer growth. With this optimization, almost \$800 US Dollars were reduced and saved from January to February 2020.

After the first year of service, the free tier ends for the service AWS API Gateway. The free tier offers 1 million requests for free. At the moment, the maximum requests were almost 400.000. Therefore, it is expected that in 3 or 4 months of constant growth, the threshold of 1 million requests will be surpassed. At this point, only the cap over the mentioned threshold will be billable. However, after 1 year of service, when the free tier expires, the real costs will be discovered. In this regard, it is left as future work the prediction of users growth and the estimated cost after the free tier of AWS services use.

5.3 Discussion and Findings

In this section, it is described events which were not anticipated before publishing the first version of the mobile application. As time passed by and feedback was received from users, bugs and errors appeared which had to be fixed as quickly as possible.

5.3.1 Issues with the payment method

The consideration with the highest impact was on users who had problems with their payment methods. For instance, the scenario where a customer who inserted a credit card which was approved by the payment service provider. However, the credit card reached its spending limit in the next few days. After a few days, the user could not call or use the Sim Card anymore given there was a problem with the payments and the service was not being activated by the backend. However, the subscription manager (Killbill) counted every day that passed as if the user would still be using the service and created invoices for each day. The payments for each invoice still failed and Killbill has a default feature which retries to charge the credit card after some time if the payment is not successful. After some time, the credit card could be used again and the user noticed he was being "randomly" charged by the mobile application. Afterwards, the user decided to open the mobile application and was asked to provide a new credit card. The user changed the payment method and, all of a sudden, the user was charged one day over 45 Swiss francs and the next day over 60 Swiss francs. Naturally, the user was not happy and contacted customer support to try to recover what was charged to his credit card and to stop the random payments of 2 or 3 Swiss francs which appeared on his credit card statement everything once in a while. The problem here relied on the subscription not being cancelled within Killbill after a problem with the payment method. Additionally, Killbill retried to charge money to the credit card after some time if the payment was not successful. This issue had to be fixed quickly to respond positively to users with problems within their payment method.

5.3.2 Sim Card being activated after 10+ minutes

Another hotfix had to be deployed for a considerable amount of users regarding the Sim Card activation. As it was explained in the previous chapter, the API calls to the Provisioning Portal (PP) had to be split into two calls given they are asynchronous requests. In other words, it is not feasible to wait until a response comes from the PP. Instead, every 30 seconds, a request asking the status is sent.

However, a problem appeared when many users were found to be on-boarded but with no phone number. This is a state which should never happen. Every on-boarded user should have an active number, a subscription attached to it and payments should be charged to that subscription. What was happening was that the users tried to activate their Sim Card and PP took more than 10 minutes to respond with an activated status. After the 10 minutes, the function returned an error to the users and asked them to rescan their Sim Card to attempt to activate it again. Nevertheless, in this second attempt to activate the Sim Card, the PP responded immediately with an error given the Sim Card was already activated with another number. There was a bug in the backend which returned the error to the user but at the same time made changes to the database allowing the user to move forward within the digital on-boarding process.

Once a user had the failed activated Sim Card and moved forward in the on-boarding process, they could choose a subscription to activate. Either a daily, which is charged every day, or a monthly one. If the user chose a daily one, the subscription would be added to a ghost account with no number attached to it. Therefore, the user would see no number within their carousel but would be charged every day for the active subscription in the ghost account. Additionally, they could not deactivate the subscription given there was no number within the carousel to interact with. Moreover, when a user tried again to activate the Sim Card within the carousel, the same error was returned to the user with no new information. That meant the users were stuck in a ghost state, being charged every day for a subscription which they are not using and with no possibilities of deactivating it.

These users had to be manually hot-fixed, which meant refund the money they had already paid and attach the ghost account to the correct phone number. The Sim Cards were, indeed, activated with the first phone number but after 90-120 minutes. Not the initial 10 minutes which was expected from the promises of the Provisioning Portal. After this experience, a new logic had to be implemented to the activate number endpoint. After the 10 minutes, an error would be returned to the user and no data would be written to the database so the users are stuck in the on-boarding process and cannot move forward. Then a different endpoint had to be implemented which was to retry to find the status of the Sim Card, instead of retrying to activate the Sim Card again. Therefore, after the 10 minutes wait, the users could now manually request information about the Sim Card. If after the manual retry, the Sim Card is still not active, a message is returned to the user asking them to recheck the status in 30 minutes. Until eventually the Sim Card will have the activated status and the digital on-boarding process can move forward.

5.3.3 Misunderstood publicity and advertisement

A large amount of publicity and advertisement was launched within Switzerland to promote the mobile application. To attract new users, it was offered the first month for free. This offer was valid if a user would choose the monthly subscription. However, users understood as if they choose any subscription they would not have to pay for the first month. For example, a user contacted customer support asking why was he being charged every day for his Daily subscription if the first month should be for free. Afterwards, the publicity and advertisement had to be adjusted to clearly state that the offer is only valid on the Monthly type of subscriptions.

5.3.4 Identity Verification instructions

At first, it was expected that the instructions to perform an identity verification were easy to understand and nobody would have trouble with the service. After noticing the big amount of failed attempts, an extra screen had to be included explaining the steps needed to the users. Especially, users had trouble taking a selfie. Some one them would take a selfie to the picture printed on the identity card or on the passport. Others would take pictures with glasses or objects blocking the selfie which caused the verification to fail.

5.4 Future Deployments

Chapter 6

Conclusions and Future Work

In this master thesis, a novel method to achieve a complete digital transformation has been presented. The method was designed and implemented to be evaluated from a usable and technical point of view.

The main goal of the thesis was to remove the unnecessary paperwork needed to purchase a new mobile subscription in Switzerland. The usual process takes around 2 to 3 weeks to be completed and costs 50 francs to order a new Sim Card. By developing a mobile application and transforming the whole process into a digital one, the time was reduced to approximately 5 minutes. The Sim Card is offered for free and two subscription plans can be bought, a daily or a monthly one. The functional requirements were to have a mobile application that is straightforward, intuitive and containing seamless animations, which can keep a customer engaged during the digital on-boarding process, and the process performed within the mobile application was needed to be as legal as the regular process done with signed physical contracts.

The objective was accomplished and described within this thesis. By choosing the right technologies, it was possible to create a working prototype which was evaluated by the System Usability Scale. For the frontend, React Native was used being it is a powerful tool and easy to learn . The choice of AWS, for the backend, was a challenging one given AWS is not an easy infrastructure to learn and deploy for any project. Only by leveraging the knowledge already acquired from the supervisor of this project, it was possible to set up the necessary infrastructure to communicate the backend server with the mobile application, the services attached to it and to provide excellent response times.

Many difficulties were encountered during the implementation of this project given the functional requirements and their reach. The infrastructure was created keeping in mind its capabilities of self-scaling, capable of handling thousands of users at the same time with an SLA level of 99.9 % uptime/availability. For every service introduced within this thesis, it was necessary to learn it in a quickly manner to overcome the challenges imposed by them. Some services communicate via SOAP, others with REST. Some services are firewall protected and need extra layers of security to communicate with external processes. Other services require native libraries within the mobile application to function and be legally valid.

The mobile application was published on December 2019 within both stores: Google Play Store and Apple App Store. Since then, the application has been closely monitored to gather insights, statistics and usage data.

6.1 Future Work

Although, the mobile application is already deployed and it is being used by real users at the moment, there is still improvements to be done in the future. For example, there is the need to transport existing numbers into the MVNO's network to reach a bigger quantity of users. This means if a user is already a customer from another provider, the mobile application should present them with the opportunity of switching their number to the application's network. Additionally, in the future it is expected to have mobile devices which support Electronic Sim Cards. At the moment, only a few iPhones are capable of this feature. Once most devices support ESim, the mobile application should not deliver Sim Cards to the users. Instead, it should automatically write the Electronic Sim Card profile to the device. To create a better user experience, push notifications should be introduced informing the user of every step which has happened. For example, if a recurring payment is charged or the credit card is found to be expired, then a notification should be sent to the user informing them of what just happened. This method can also be leveraged to send mass promotions or news about the project to the users.

Bibliography

- [1] Li, L., & Yang, X. (2016). U.S. Patent No. 9,439,062. Washington, DC: U.S. Patent and Trademark Office.
- [2] Saxena, S., Low, D., & Vaish, G. (2017). U.S. Patent No. 9,806,942. Washington, DC: U.S. Patent and Trademark Office.
- [3] Chin, E., & Wagner, D. (2013, August). Bifocals: Analyzing webview vulnerabilities in android applications. In International Workshop on Information Security Applications (pp. 138-159). Springer, Cham.
- [4] CVE-2019-5765 Detail, <https://nvd.nist.gov/vuln/detail/CVE-2019-5765>, last accessed January 2020.
- [5] CVE-2012-3746 detail, <https://www.cvedetails.com/cve/CVE-2012-3746/>, last accessed January 2020.
- [6] Pilorz, L & Zmyslowski M. Exploring and Exploiting iOS Web Browsers, <https://www.cvedetails.com/cve/CVE-2012-3746>, last accessed January 2020.
- [7] Davis, N. (2011). Information overload, reloaded. Bulletin of the American Society for Information Science and Technology, 37(5), 45-49.
- [8] Pham, X. L., Chen, G. D., Nguyen, T. H., & Hwang, W. Y. (2016). Card-based design combined with spaced repetition: A new interface for displaying learning elements and improving active recall. Computers & Education, 98, 142-156.
- [9] XML Soap, https://www.w3schools.com/xml/xml_soap.asp, last accessed January 2020.
- [10] What is REST?, <https://www.codecademy.com/articles/what-is-rest>, last accessed January 2020.
- [11] A guide to digital identity verification: the technology and trends, <https://hub.onfido.com/web-embeds-document-verification/a-guide-to-digital-identity-verification-the-technology-and-trends>, last accessed January 2020.
- [12] Payments, <https://www.twint.ch/en/private-customers/functions>, last accessed January 2020.

- [13] Datatrans features, <https://www.datatrans.ch/en/features/>, last accessed January 2020.
- [14] Yifrah, K (2019). Are you sure you want to do this? Microcopy for confirmation dialogues shorturl.at/jCEJR, last accessed January 2020.
- [15] React Native, <https://facebook.github.io/react-native/>, last accessed January 2020.
- [16] Performance, <https://facebook.github.io/react-native/docs/performance>, last accessed January 2020.
- [17] Expo, <https://expo.io/>, last accessed January 2020.
- [18] React Native Reanimated, <https://software-mansion.github.io/react-native-reanimated/index.html>, last accessed January 2020.
- [19] Control Access to an API with IAM Permissions, <https://docs.aws.amazon.com/apigateway/latest/developerguide/permissions.html>, last accessed January 2020.
- [20] Amplify Framework, <https://aws-amplify.github.io/docs/>, last accessed January 2020.
- [21] Amazon Cognito, <https://aws.amazon.com/cognito/>, last accessed January 2020.
- [22] Signature Version 4 Signing Process, <https://docs.aws.amazon.com/general/latest/gr/signature-version-4.html>, last accessed January 2020.
- [23] Getting Started with Redux, <https://redux.js.org/introduction/getting-started>, last accessed January 2020.
- [24] Mirkovic, J., & Reiher, P. (2004). A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2), 39-53.
- [25] Serverless Computing. <https://aws.amazon.com/serverless/>, last accessed January 2020.
- [26] Habermann, A. N., & Notkin, D. (1986). Gandalf: Software development environments. *IEEE transactions on software engineering*, (12), 1117-1127.
- [27] Amazon Route 53. <https://aws.amazon.com/route53/>, last accessed January 2020.
- [28] Amazon Virtual Private Cloud. <https://aws.amazon.com/vpc/>, last accessed January 2020.
- [29] Amazon API Gateway. <https://aws.amazon.com/api-gateway/>, last accessed January 2020.
- [30] Masse, M. (2011). *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. " O'Reilly Media, Inc."

- [31] Cross-Origin Resource Sharing (CORS). MDN developers. <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>, last accessed January 2020.
- [32] Elastic IP Addresses, <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html>, last accessed January 2020.
- [33] Elastic Load Balancing. <https://aws.amazon.com/elasticloadbalancing/>, last accessed January 2020.
- [34] Amazon CloudWatch. <https://aws.amazon.com/cloudwatch/>, last accessed January 2020.
- [35] AWS Lambda. <https://aws.amazon.com/lambda/>, last accessed January 2020.
- [36] Amazon DynamoDB. <https://aws.amazon.com/dynamodb/>, last accessed January 2020.
- [37] Stonebraker, M. (2010). SQL databases v. NoSQL databases. *Communications of the ACM*, 53(4), 10-11.
- [38] AWS Secrets Manager. <https://aws.amazon.com/secrets-manager/>, last accessed January 2020.
- [39] AWS CodeCommit. <https://aws.amazon.com/codecommit/>, last accessed January 2020.
- [40] AWS CodePipeline. <https://aws.amazon.com/codepipeline/>, last accessed January 2020.
- [41] Villamizar, M., Garces, O., Ochoa, L., Castro, H., Salamanca, L., Verano, M., ... & Lang, M. (2016, May). Infrastructure cost comparison of running web applications in the cloud using AWS lambda and monolithic and microservice architectures. In 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid) (pp. 179-182). IEEE.
- [42] Kharenko, A. (2015, Oct). Monolithic vs. Microservices Architecture. In *Microservices Praxitioner Articles*. <https://articles.microservices.com/monolithic-vs-microservices-architecture-5c4848858f59>, last accessed January 2020.
- [43] What is Kill Bill? https://docs.killbill.io/latest/what_is_kill_bill.html, last accessed January 2020.
- [44] Address verification. <https://www.post.ch/en/business-solutions/address-maintenance/address-verification>, last accessed February 2020.
- [45] What is a webhook? <https://support.onfido.com/hc/en-us/articles/115002001969-What-is-a-webhook->, last accessed February 2020.
- [46] Funke Lettershop. <https://www.funkelettershop.ch/>, last accessed February 2020.

- [47] Support API. https://developer.zendesk.com/rest_api/docs/support/introduction, last accessed February 2020.
- [48] Shin, D. H. (2010). MVNO services: Policy implications for promoting MVNO diffusion. *Telecommunications Policy*, 34(10), 616-632.
- [49] Sunrise und Finecom unterzeichnen MVNO-Vertrag. Aug, 2010. https://www.itreseller.ch/Artikel/65107/Sunrise_und_Finecom_unterzeichnen_MVNO-Vertrag.html, last accessed February 2020.
- [50] Martanteile der Mobilfunkanbieter (2019). Schweizerische Eidgenossenschaft. <https://www.comcom.admin.ch/comcom/de/home/dokumentation/zahlen-und-fakten/mobilfunkmarkt/martanteile-der-mobilfunkanbieter.html>
- [51] Sunrise mobile rate plans (2020). Sunrise. <https://www.sunrise.ch/en/residential/mobile/freedom.html>
- [52] Swisscom inOne mobile go (2020). Swisscom. https://www.swisscom.ch/en/residential/mobile/subscription-tariffs_new.html
- [53] The mobile subscriptions (2020). Salt. <https://www.salt.ch/en/plus>
- [54] Carine Lallemand and Guillaume Gronier. 2012. Enhancing User eXperience during waiting time in HCI: contributions of cognitive psychology. In *Proceedings of the Designing Interactive Systems Conference (DIS '12)*. Association for Computing Machinery, New York, NY, USA, 751-760. DOI:<https://doi.org/10.1145/2317956.2318069>
- [55] Doherty, R. A., & Sorenson, P. (2015). Keeping users in the flow: mapping system responsiveness with user experience. *Procedia Manufacturing*, 3, 4384-4391.
- [56] Westerman, G., Bonnet, D., & McAfee, A. (2014). The nine elements of digital transformation. *MIT Sloan Management Review*, 55(3), 1-6.
- [57] Ivanecky, J. Evolution of retail banking in the digital era (2016), <https://www.ubs.com/microsites/innovation/en/innovation-now/2016/retail-banking-in-the-digitalization-era.html>
- [58] Lebara Switzerland App. <https://play.google.com/store/apps/details?id=ch.lebara.app&hl=en>
- [59] My Swisscom. <https://play.google.com/store/apps/details?id=com.swisscom.myswisscom&hl=en>
- [60] Evolution of the mobile phone. <https://www.tigermobiles.com/evolution/#start>
- [61] Nakhimovsky, Y., Eckles, D., & Riegelsberger, J. (2009). Mobile user experience research: challenges, methods & tools. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems* (pp. 4795-4798).
- [62] Mendoza, A. (2013). Mobile user experience: patterns to make sense of it all. Newnes.

- [63] UBS Mobile Banking. <https://apps.apple.com/ch/app/ubs-mobile-banking/id441068021?l=en>, last accessed February 2020.
- [64] Revolut. <https://www.revolut.com/>
- [65] How to sign up to Revolut. <https://youtu.be/laDDN-yQfFo>
- [66] Firebase helps mobile and web app teams succeed. <https://firebase.google.com/>
- [67] Whatsapp, simple, secure, reliable messaging. <https://www.whatsapp.com/>
- [68] System Usability Scale (SUS), <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>, last accessed February 2020.
- [69] System Usability Scale Template, <http://www.uxforthemasses.com/wp-content/uploads/2011/04/SUS-System-Usability-Scale.doc>, last accessed February 2020.
- [70] Pelloni, L. (2019). Analytics with Passive Wi-Fi Signals. Publications, Department of Informatics - Communication Systems Group, University of Zurich.
- [71] Verordnung über die Überwachung des Post- und Fernmeldeverkehrs. Schweizerische Bundesrat. (2017) <https://www.admin.ch/opc/de/classified-compilation/20172173/index.html#a20>, last accessed March 2020.
- [72] AWS Free Tier. <https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc>, last accessed March 2020.
- [73] Sauro, J. (2013). 10 THINGS TO KNOW ABOUT THE SYSTEM USABILITY SCALE (SUS). MeasuringU. <https://measuringu.com/10-things-sus/>, last accessed March 2020.
- [74] Codrut, N. (2019). Dual SIM - What is it? What does Dual SIM mean? How does Dual SIM work? <https://www.digitalcitizen.life/simple-questions-what-dual-sim-and-how-does-it-work>, last accessed March 2020.
- [75] Android vs. iOS Users: How Do They Behave Differently?. Buildfire. <https://buildfire.com/ios-android-users/>, last accessed March 2020.
- [76] Albanesius, C. (2011). Infographic: What Does the Average Android, iOS User Look Like? <https://uk.pcmag.com/news/111939/infographic-what-does-the-average-android-ios-user-look-like>, last accessed March 2020.

List of Figures

3.1	Workflow of the method	13
4.1	Monolithic vs Microservices architecture, taken from BMC.com	25
4.2	Initial state of a registered user	30
4.3	Onfido identity verification	38
5.1	Platforms being used	44
5.2	Boxplot Questions 1-10	48
5.3	Effort needed to activate a Sim Card	49
5.4	Duration until the Sim Card was delivered	50
5.5	Price / Value Ratio	51
5.6	Mobile network coverage	51
5.7	Duration until the Sim Card was delivered	52
5.8	Voice quality	53
5.9	Customer support	53
5.10	Would you recommend the mobile application to your friends or colleagues?	54

List of Tables

4.1	Comparison of different open-source cross-platform frameworks	18
5.1	Raw and Final Scores for SUS	47
5.2	Participants SUS Score Survey	47
5.3	Requested received and processed by AWS Lambda from December 2019 to February 2020	55

Listings

4.1	Function returning an authenticated user or undefined	19
4.2	Function authenticating requests	21
4.3	Headers being injected into the requests	22
4.4	Free trial for 30 days	32
4.5	Normal subscription with recurring billing	33

Appendix A

Installation Guidelines

The mobile application can be downloaded for both Android and iOS platforms.

To download the iOS version: <https://apps.apple.com/ch/app/swype-mobile-just-better/id1479174713>

To download the Android version: <https://play.google.com/store/apps/details?id=ch.swype&hl=en>

Appendix B

Contents of the CD

Within the cd, 4 Android APK can be found pointing to the DEV environment. Each APK is designed for a different architecture CPU for Android devices.

The test data to complete the on-boarding process within the DEV environment is:

- The address can be any address, as long as it is a valid address in Switzerland.
- For the identity verification, a photo of anything can be taken to proceed with the process. This works for both passport and identity card, as well as for the selfie required.
- Test credit card number: 4242 4242 4242 4242
- Test credit card expiry date: 12/21
- Test credit card cvv: 123
- Sim Card manual code: 11111111111111111111