



UNIVERSIDADE D  
COIMBRA

Rúben Filipe Dias Tomás

## SOLUTION FOR TEMPORARY PHONE NUMBERS

Dissertation in the context of the Master in Informatics Engineering, Specialization in Software Engineering advised by Professor João Barata and presented to Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2020

Faculty of Sciences and Technology  
Department of Informatics Engineering

# Solution for Temporary Phone Numbers

Rúben Filipe Dias Tomás

Dissertation in the context of the Master in Informatics Engineering, Specialization in Software  
Engineering advised by Prof. João Barata and presented to the  
Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2020



UNIVERSIDADE D  
COIMBRA

This page is intentionally left blank.

---

## Acknowledgements

Foremost, I thank WIT Software S.A for this internship opportunity, a great chance to learn and for professional development. Therefore, I consider myself as a fortunate individual, as I was provided with a chance to be a part of it.

Bearing this in mind, I am using this opportunity to express great gratitude and special thanks to Jorge Sousa and João Costa my supervisors at WIT Software who, in spite of being busy with their duties, took time out to hear, guide and keep me on the correct path and allowing me to carry out the project.

Next, I would like to express profound gratitude to Professor João Barata, my tutor, for his guidance, encouragement and useful advice all throughout this internship.

A big thanks to all my friends but especially to those who allowed me to experience the true academic spirit during this stage of my life, André Clemêncio, André Gouveia, António Filipe, Cristiano Campos, Christophe Diogo, Luís Amaro, Joel Pires, José Canelha, Pedro Andrade, Rafael Henriques and Rui Reis.

I also thank my girlfriend, Carolina, for all the encouragement, words of motivation, and of course, for her guidance during the writing of this document that without her, would not be the same.

To Jessica, the best sister in the world, that was always there for me, to help and encourage me to finish this dissertation.

Finally, I would like to thank my parents, who always fought to provide me with the means and necessary conditions during my academic life. They are the two people who made all of this possible. It is to you that I dedicate my work. Thank You!



This page is intentionally left blank.

---

## Abstract

Since the beginning, communication has been the key for humankind to evolve. Throughout the years, the way people communicate with each other has changed drastically, leading to an exponential growth in the communications industry.

Nowadays, people use their phone numbers for almost everything, since bank transactions, social networks, selling goods, manage businesses or even in online dating sites. As a result, a new type of phone number emerged on the market to assure the privacy and commodity of the users. Temporary phone numbers, as they are called, are numbers that act as an intermediary between the user's personal phone and other entities.

WIT Software S.A, aware of the importance of phone numbers in the daily life of consumers, as well as security and privacy concerns related with communications, intends to know if there is a place in the market for a value proposition for a system capable of providing temporary phone numbers.

Regarding this, the purpose of this project is to develop a Minimum Viable Product of a platform able to provide temporary phone numbers to the users. In this platform, the users will be able to acquire temporary numbers for communication purposes (calls and messages), adding a layer of protection and privacy.

The present report intends to describe the whole engineering process that led to the implementation of this particular platform. This includes a representation of the planning stages of the project, a review of the state of the art, a system and architecture description, an overview of the development and testing phases, and the conclusions drawn.

## Keywords

"Temporary Phone Numbers", "REST API", "Dashboard", "Mobile Application", "CPaaS", "Twilio", "Virtual Number", "Calls", "Messages", "Platform", "Communications".

This page is intentionally left blank.

---

## Resumo

Desde do início, a comunicação tem sido essencial para a evolução da humanidade. Ao longo dos anos a forma como as pessoas comunicam entre si mudou drasticamente, levando a um crescimento exponencial na indústria das comunicações.

Nos tempos que correm, as pessoas utilizam os seus números de telefone para quase tudo desde transações bancárias, redes sociais, compra e venda de itens, gestão de negócios ou até mesmo em sites/aplicações de encontros. Como resultado, surgiu no mercado um novo número de telefone para garantir a privacidade e comodidade dos utilizadores. Designados por Números Temporários, estes números agem como intermediário entre o numero de telefone pessoal do utilizador e outras entidades.

A WIT Software S.A, estando ciente da importância dos números de telefone para a vida diária dos consumidores, bem como preocupações de segurança e privacidade relacionadas com comunicações, pretende saber se há mercado para uma proposta de valor de um sistema capaz de providenciar números de telefone temporários.

Neste sentido, o propósito deste projeto é desenvolver um Produto Viável Mínimo de uma plataforma capaz de providenciar números de telefone temporários aos utilizadores. Nesta plataforma, os utilizadores vão ser capazes de adquirir números temporários para comunicações (chamadas e mensagens), adicionando uma camada de proteção e privacidade.

O presente relatório tem a intenção de descrever todo o processo de engenharia que levou à implementação desta plataforma em particular. Isto inclui uma representação das fases de planeamento do projeto, uma revisão do estado da arte, uma descrição do sistema e da arquitetura, uma análise das fases de desenvolvimento e teste, e a apresentação das conclusões.

## Palavras-Chave

"Números de telefone Temporários", "REST API", "Dashboard", "Aplicação móvel", "CPaaS", "Twilio", "Número Virtual", "Chamadas", "Mensagens", "Plataforma", "Comunicações".

This page is intentionally left blank.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	The Institution . . . . .	1
1.2	Motivation . . . . .	1
1.3	Goals . . . . .	3
1.3.1	Internship Goals . . . . .	3
1.3.2	Project Goals . . . . .	3
1.4	Structure . . . . .	4
<b>2</b>	<b>Background Concepts</b>	<b>7</b>
2.1	Phone Numbers . . . . .	7
2.1.1	Classification of Phone Numbers: . . . . .	7
2.2	Temporary Phone Numbers . . . . .	8
2.2.1	Verification and Validation . . . . .	9
2.2.2	Communication Process . . . . .	9
2.2.3	Recycling . . . . .	10
2.2.4	Emergency Calls . . . . .	10
2.2.5	Privacy and Traceability . . . . .	11
2.2.6	Binding and Billing . . . . .	11
2.3	Service Providers . . . . .	12
2.3.1	PBX Providers . . . . .	12
2.3.2	CPaaS Providers . . . . .	12
2.3.3	Carrier Providers . . . . .	13
<b>3</b>	<b>State of the Art</b>	<b>15</b>
3.1	Competitors . . . . .	15
3.1.1	Examples of Competitors . . . . .	16
3.1.2	Competitors Analysis . . . . .	19
3.2	Market Vison . . . . .	20
3.2.1	Market Key Elements . . . . .	20
<b>4</b>	<b>Methodology and Work Plan</b>	<b>21</b>
4.1	Methodology . . . . .	21
4.1.1	Scrum Framework . . . . .	21
4.1.2	Scrum Project Adaptation . . . . .	23
4.2	Work Plan . . . . .	24
4.2.1	First Semester . . . . .	24
4.2.2	Second Semester . . . . .	25
4.3	Risk Management . . . . .	28
4.3.1	Concept of Risk Management . . . . .	28
4.3.2	Risks Description . . . . .	29
4.3.3	Risks Verified . . . . .	30

---

<b>5</b>	<b>Requirements Elicitation</b>	<b>33</b>
5.1	User Stories . . . . .	33
5.2	Functional Requirements . . . . .	34
5.3	Non Functional Requirements . . . . .	36
5.4	Constraints . . . . .	38
5.4.1	Technical Constraints . . . . .	38
5.4.2	Business Constraints . . . . .	39
5.4.3	Legal Constraints . . . . .	39
<b>6</b>	<b>Solution’s Design</b>	<b>41</b>
6.1	Model C4 . . . . .	41
6.2	Architecture . . . . .	43
6.2.1	Context View . . . . .	43
6.2.2	Containers View . . . . .	44
6.2.3	Components View . . . . .	46
6.3	Data Model . . . . .	49
6.4	Security . . . . .	50
6.4.1	Authentication . . . . .	50
6.4.2	Authenticity and Non Repudiation . . . . .	50
<b>7</b>	<b>The WIT Temporary Numbers Platform</b>	<b>53</b>
7.1	Accounts . . . . .	53
7.1.1	Account Creation . . . . .	54
7.1.2	Account Cancellation . . . . .	55
7.2	Temporary Phone Numbers . . . . .	56
7.2.1	Purchasing . . . . .	57
7.2.2	Canceling . . . . .	58
7.2.3	Calling . . . . .	58
7.2.4	Messaging . . . . .	59
7.3	Statistics . . . . .	60
7.4	Management . . . . .	61
<b>8</b>	<b>Implementation</b>	<b>63</b>
8.1	API Backend . . . . .	63
8.1.1	Structure . . . . .	63
8.1.2	Architectural Constraints . . . . .	66
8.1.3	Web Security . . . . .	67
8.1.4	Exception Handling . . . . .	67
8.1.5	Data Management System . . . . .	67
8.1.6	Integration with Twilio . . . . .	68
8.2	Mobile App . . . . .	70
8.2.1	Structure . . . . .	70
8.2.2	Navigation . . . . .	71
8.2.3	Data Management and Permissions . . . . .	72
8.2.4	Interface . . . . .	72
<b>9</b>	<b>Verification and Validation</b>	<b>83</b>
9.1	Verification . . . . .	84
9.1.1	Unit Testing . . . . .	84
9.1.2	Integration Testing . . . . .	85
9.1.3	Functional Testing . . . . .	85
9.2	Validation . . . . .	86

<b>10 Conclusions and Future Work</b>	<b>87</b>
10.1 Overview . . . . .	87
10.2 Success Evaluation . . . . .	88
10.3 Future Work . . . . .	88
10.4 Final Considerations . . . . .	88
 <b>Appendices</b>	 <b>97</b>
 <b>A Requirements</b>	 <b>99</b>
A.1 Functional Requirements . . . . .	99
A.2 Non Functional Requirements . . . . .	106
 <b>B Project Risks</b>	 <b>108</b>
B.1 Risks . . . . .	108
 <b>C Mockups</b>	 <b>112</b>
C.1 Mobile Application Mockups . . . . .	112
C.2 Dashboard Mockups . . . . .	115
 <b>D Tests</b>	 <b>118</b>
D.1 Functional Tests . . . . .	118



This page is intentionally left blank.

# Acronyms

**CPaaS** Communications Platform as a Service.

**DAO** Data Access Object.

**DID** Direct Inward Dial.

**IMS** IP Multimedia Subsystem.

**MVP** Minimum Viable Product.

**PBX** Private Branch Exchange.

**RCS** Long-Term Evolution.

**RCS** Rich Communication Services.

**SIP** Session Initiation Protocol.

**VOIP** Voice Over Internet Protocol.

**VoLTE** Voice over Long-Term Evolution.

This page is intentionally left blank.

# List of Figures

1.1	Mobile Users by % of Population . . . . .	2
1.2	"When is a Mobile used?" . . . . .	2
4.1	Scrum life cycle . . . . .	22
4.2	Gantt chart for the first semester . . . . .	24
4.3	Planned Gantt chart for the second semester . . . . .	25
4.4	Gantt chart for the work done in the second semester . . . . .	27
4.5	Risks Matrix . . . . .	29
4.6	Risks Matrix of the project . . . . .	30
6.1	Notation used with C4 model . . . . .	42
6.2	Contextual Diagram of WIT Temporary Numbers Platform . . . . .	44
6.3	Containers Diagram of WIT Temporary Numbers Platform . . . . .	46
6.4	Components Diagram of API Backend container . . . . .	48
6.5	Entity Relationship Diagram . . . . .	49
6.6	Diagram of JWT creation and usage . . . . .	51
7.1	Diagram of account managing process . . . . .	54
7.2	Diagram of account creation process . . . . .	55
7.3	Diagram of account cancellation process . . . . .	56
7.4	Sequence diagram of purchasing a number . . . . .	57
7.5	Sequence diagram of canceling a number . . . . .	58
7.6	Sequence diagram of calls process . . . . .	59
7.7	Sequence diagram of sending and receiving a message . . . . .	60
7.8	Sequence diagram of the statistics features . . . . .	61
7.9	Sequence diagram of management features . . . . .	62
8.1	Spring Boot structure [1] . . . . .	64
8.2	Spring Boot flow architecture [1] . . . . .	64
8.3	Twilio credentials generated after the Twilio account creation . . . . .	68
8.4	Example of a HTTP request . . . . .	69
8.5	Twilio credentials generated after the Twilio account creation . . . . .	69
8.6	Acknowledgement message showed in every Regulatory Bundle . . . . .	69
8.7	Types of navigators chosen . . . . .	71
8.8	Navigation diagram . . . . .	72
8.9	Phone number insertion screen . . . . .	73
8.10	Code verification screen . . . . .	73
8.11	Message received containing the OTP code . . . . .	73
8.12	Account without purchased temporary phone numbers screen . . . . .	74
8.13	Account with purchased temporary phone numbers screen . . . . .	74
8.14	Phone number settings screen . . . . .	74
8.15	Edit phone number description screen . . . . .	75

8.16	Change phone number plan screen . . . . .	75
8.17	Selecting phone number characteristics screen . . . . .	75
8.18	Example of temporary phone numbers screen . . . . .	75
8.19	Naming temporary phone number screen . . . . .	76
8.20	Choosing a plan screen . . . . .	76
8.21	User profile screen . . . . .	76
8.22	Dialpad screen . . . . .	77
8.23	User profile screen . . . . .	77
8.24	Call in progress screen . . . . .	77
8.25	Allow access to contacts message screen . . . . .	78
8.26	Imported contacts screen . . . . .	78
8.27	Message history with no conversations screen . . . . .	79
8.28	Message history with conversations screen . . . . .	79
8.29	Create new message: select select contact screen . . . . .	79
8.30	Create new message: write message screen . . . . .	79
8.31	Conversation screen . . . . .	80
8.32	Calls history screen . . . . .	80
8.33	Empty call history screen . . . . .	80
8.34	Settings screen . . . . .	81

# List of Tables

3.1	Competitors Analysis . . . . .	19
4.1	Risk 01 - Project Complexity . . . . .	29
5.1	Users Stories. . . . .	34
5.2	Functional Requirement 01 - Registration . . . . .	35
5.3	Quality Attribute scenario 01 - Availability . . . . .	38
5.4	Project technical constraints . . . . .	38
5.5	Project business constraints . . . . .	39
6.1	Platform Containers . . . . .	44
6.2	Platform Containers Technologies . . . . .	45
6.3	Communication Protocols . . . . .	45
6.4	Data model entities . . . . .	49
8.1	Dependencies used on the API Backend implementation . . . . .	65
8.2	Platform Components . . . . .	71
9.1	Functional test case example . . . . .	86
A.1	Functional Requirement 01 - Registration . . . . .	99
A.2	Functional Requirement 02 - Sign In . . . . .	100
A.3	Functional Requirement 03 - Show Main Menu . . . . .	100
A.4	Functional Requirement 04 - Buy Number . . . . .	101
A.5	Functional Requirement 05 - Make a Call . . . . .	101
A.6	Functional Requirement 06 - Show calls history . . . . .	101
A.7	Functional Requirement 07 - Send a Message . . . . .	102
A.8	Functional Requirement 08 - Messages History . . . . .	102
A.9	Functional Requirement 09 - Show Contacts . . . . .	102
A.10	Functional Requirement 10 - Show Settings . . . . .	103
A.11	Functional Requirement 11 - Add Funds . . . . .	103
A.12	Functional Requirement 12 - Cancel a Number . . . . .	103
A.13	Functional Requirement 13 - Cancel an Account . . . . .	104
A.14	Functional Requirement 14 - System Overview Menu . . . . .	104
A.15	Functional Requirement 15 - Managing Menu . . . . .	104
A.16	Functional Requirement 13 - Add/Remove Funds to/from a User Account . . . . .	105
A.17	Functional Requirement 15 - Send a Notification . . . . .	105
A.18	Functional Requirement 19 - Delete User Account . . . . .	105
A.19	Quality Attribute scenario 01 - Availability . . . . .	106
A.20	Quality Attribute scenario 02 - Security . . . . .	106
A.21	Quality Attribute scenario 03 - Interoperability . . . . .	106
A.22	Quality Attribute scenario 04 - Modifiability . . . . .	106

B.1	Risk 02 - Schedules	108
B.2	Risk 03 - Integration with External Entities	108
B.3	Risk 04 - Technologies Learning	109
B.4	Risk 05 - Requirements Change	109
B.5	Risk 06 - Tests	109
B.6	Risk 07 - Pandemic Situation	110
B.7	Risk 08 - Project Validation	110

# Chapter 1

## Introduction

This document intends to explain the work done by the author during the one-year internship that took place at WIT Software, S.A headquartered in Taveiro, Coimbra.

The internship is included in the Master's degree in Informatics Engineering of the University of Coimbra and is supervised by João Barata, professor at the University of Coimbra, and by Jorge Sousa, project manager at WIT Software.

This chapter is divided into four sections and intends to demonstrate the overall perspective of the report. The first section identifies the company where the internship took place. The second section describes the motivation for the project. Next, section 1.3 exposes the internship and project goals. Lastly, the fourth section presents the structure of the remainder document.

### 1.1. The Institution

WIT [2] is a software company that creates advanced solutions and white-label products for mobile telecommunications. Founded in March of 2001, WIT was born as a spin-off company of the University of Coimbra with the goal of creating innovative solutions for the mobile world. Since 2001, WIT Software specialized in advanced solutions and whitelabel products for mobile telecommunications companies. Some of the expertise areas are I.P. Communications and carrier messaging, Rich Communication Services (RCS) technology, Voice Over Internet Protocol (VOIP), Mobile voip and Voice over Long-Term Evolution (RCS), IP Multimedia Subsystem (IMS) and Voice over Long-Term Evolution (VoLTE), and others. The company also expanded across the world with offices in Portugal (Porto, Leiria and Lisboa), Germany (Dusseldorf), England (Reading) and USA (Menlo Park) [3].

### 1.2. Motivation

Communication, as we know it today, is the result of years of technological evolution. It all began in 1876 when Graham Bell invented the telephone. Since then, technological



breakthroughs have allowed new forms of communication and new types of devices to emerge.

Nowadays, cell phones have a meaningful influence on our lives, no matter where we live. The infographic presented in figure 1.1 shows staggering statistics about the number of mobile users by percentage of population. As we can observe, areas like the countries that once made up the Soviet Union (shown as CIS in the figure) have a percentage of 143%, which means around one mobile phone and a half per person, and in China and India together, we can find 30% of the world's mobile users. This graphic leads us to conclude that the communications industry is now widespread around the globe.

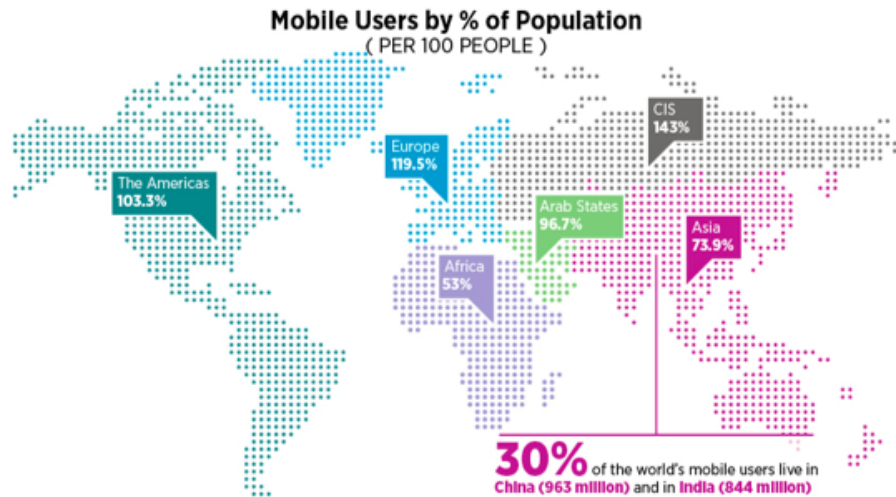


Figure 1.1: Mobile Users by % of Population [4]

With the expansion and ease of communication, people are starting to get concerned about their privacy. We use our phone number for almost everything since bank transactions, social networks, managing businesses, selling goods and even in dating apps/sites. As we can see in figure 1.2, mobile devices have a strong influence in our everyday lives because, although mostly used at home (98% of mobile users admit to using them at home), we use them any and everywhere.



Figure 1.2: "When is a Mobile used?" [5]

New apps and services are emerging on the market [6] to ensure privacy and commodity, with the promise of providing more than one number in the same device to any part of the globe.

Acknowledging the expansion of the market, WIT Software SA. is interested in developing its own solution. This should allow users to buy temporary numbers from any country/region of the world and use them to communicate with other users with assured

privacy. WIT Software SA is interested in understanding what can be done differently from other competitors to offer an Minimum Viable Product (MVP) [7] to the customers. With that purpose in mind, a prototype must be developed to verify if it suits the market. Therefore, this internship will comprise in the implementation of a platform capable of providing temporary numbers and communication services.

In essence, the WIT Temporary Number Platform prototype should offer the users the possibility of purchasing temporary numbers and use them to communicate without having to buy another device.

### 1.3. Goals

This section describes the goals that are expected of the internship and the project. For this purpose, it was divided into two subsections. The first subsection explains the internship's goals and the second describes the goals of the project itself.

#### 1.3.1. Internship Goals

As this internship is the last step of the intern's academic career and the first real-world project that he will work on, the author of this thesis is aware that he must consider himself first as a learner and only then as a contributor. With that in mind, the intern defined two main goals that are interconnected:

- **Personal:** from a personal point of view, the intern desires to perfect interpersonal skills (soft skills) and start to develop a solid work ethic and professional demeanour.
- **Professional:** from a professional point of view, the author intends to obtain working knowledge, master technical skills and learn the processes and methodologies used in the enterprise environment.

During the internship, the intern knows that he must be a proactive person in terms of using knowledge already acquired during the Master's degree in Informatics Engineering or search and use new methods/techniques to solve the issues that may appear throughout the internship.

#### 1.3.2. Project Goals

As mentioned previously, WIT Software, S.A is conscious of how the voice communications market is saturated and also of the new ways and ideas that operators can resort to in order to encourage users to use mobile communications more safely and privately.

Taking the previous points into account, the goal of the intern's proposal consists of designing and implementing a solution capable of providing temporary phone numbers for specific operations (e.g. for a user to make an online sale or to use in "dating websites") and then dispose of the number. Although it is disposable, the numbers provided need to be associated with a valid user's phone number until they decide to cancel it. After a

number is disposed of by a user, a recycling process needs to take place before the number can become available to be purchased by other users once again.

As no software related to this proposal has been developed by WIT, the intern is in charge of developing an MVP of a platform that allows:

- Buying temporary numbers;
- Receiving and making phone calls using a temporary number;
- Sending and receiving messages using temporary numbers;
- Managing the temporary numbers;

Keeping in mind the situation stated, the solution decided by the intern together with Wit Software, SA should be composed by three major components:

- **Server** - The creation of a server is crucial to provide the services (e.g. numbers, calls and messages). The server will be responsible for providing the services, managing data structures and handling logic operations.
- **Mobile App** - In order to use the services provided by the server, an app would be created with a user-friendly interface. As the target client's OS of the internship was not restricted, the intern opted for Android.
- **Admin Dashboard** - The output expected from this component should be a website where system administrators can manage the platform itself (temporary phone numbers and users).

Building these three new systems, the Server, the App, and the Dashboard, will allow creating a solution that can be used in the future to understand what can be done differently and better in order to achieve a better solution.

At the end of this internship, a final product is not expected to be ready to deliver to a real customer. However, it is deemed that the output should be able to help the company define the next steps to be taken in the future.

## 1.4. Structure

The document structure is explained in this section through a brief description of the chapters and the appendices.

The chapters are as follows:

- **Chapter 1 - Introduction:** introduces the project and its goals.
- **Chapter 2 - Background Concepts:** introduces knowledge needed beforehand.
- **Chapter 3 - State of Art:** contains a brief description of project concepts, a competitors analysis and the technologies chosen to implement the solution.

- **Chapter 4 - Methodology and work plan:** describes the used methodology during the internship, the development plan and the risk analysis
- **Chapter 5 - Requirements Elicitation:** presents the requirements elicitation of the project.
- **Chapter 6 - Solution's Design:** entails the architecture of the solution to be developed by the intern.
- **Chapter 7 - The WIT Temporary Numbers Platform:** presents the overall decisions made by the author to implement the solution.
- **Chapter 8 - Implementation and Work Done:** describes the work done throughout the year, the final result and some implementation details
- **Chapter 9 - Verification and Validation:** explains how the the process of verification and validation were handled and presents the tests that were performed.
- **Chapter 10 - Conclusions and Future Work:** exposes the overall conclusions of the project and presents future work.

In terms of appendices this document is made up by:

- **Appendix A - Requirements:** details the requirements (functional and non-functional) established during the elicitation requirements phase.
- **Appendix B - Risks:** describes the risks encountered during the internship.
- **Appendix C - Mockups:** illustrates the UI's that were made based on the defined use cases.
- **Appendix D - Tests:** exposes the tests that were performed in the verification/validation process.

This page is intentionally left blank.

## Chapter 2

# Background Concepts

The purpose of this chapter is to introduce and contextualize some concepts of the internship's topic. Thus, all the information presented helps the reader understand better the problem being investigated and promotes confidence in the overall quality of analysis and findings presented further ahead.

### 2.1. Phone Numbers

A phone number at its most basic level is just a series of digits that can be compared to a computer's IP address or even to a street address. The evolution of technology has brought the possibility of being in touch instantly with other people, hence the combination of numbers allowed the systems to know who the caller seeks to connect to. This led the cell phone to become a necessity for many people either for personal or business matters.

In fact, in the last years, we have seen much activity related to the use of our phone number not only to make communications but also for authentication mechanisms, payment authorizations, bank transactions authorizations, among others. Since then, companies have started to acknowledge the usefulness of the phone number beyond communications [8].

#### 2.1.1. Classification of Phone Numbers:

Modern technology made it possible to create many types of phone numbers. Nowadays, the variety of phone numbers available on the market is so big that it is possible to classify them by format (fixed or mobile), per application (for personal or business purposes) [9] or by location (geographical and non-geographical) [10, 11, 12] . This type of classifications allows to generalize and group the phone numbers.

Depending on the size, type of device or capabilities, phone number types can be:

- **Landline Number:** is a regular phone number that relies on physical wires to enable voice calls. [13] [14]
- **Mobile Number:** is a telephone number like any other. The difference is that it

is registered to a mobile phone/network, instead of a fixed or VoIP phone/network [9, 15].

- **Internal Number:** is a private number that can be dialled by any connected extension in your phone system. It is possible to route an internal number to any destination from the user's account, such as an address contact, or a voicemail box [16].
- **Virtual Number:** is used for rapid international communication without being tied to the SIM card and cable. Virtual phone numbers can be paired with local or toll-free exchanges, and route calls that originated from the public switched telephone network (PSTN), voice over Internet protocol (VoIP) networks or cellular phones [9].
- **Short Code:** is a special 5 or 6 digit phone number that's shorter than a full phone number. Short codes are used to send and receive SMS and MMS messages to and from mobile phones. Short codes are pre-approved by carriers to have a high throughput. This makes them perfect for sending those high-volume or time-sensitive messages. [17].
- **Toll Free Number:** is available in many countries such as the USA, United Kingdom and many others. It is generally free for the caller to call a Toll free number – as long as they call within their own country. The company that answers the calls pays the call on behalf of the caller, so it is more expensive to receive calls on toll free numbers than on other types of numbers [9, 15].
- **SIP Number:** A Session Initiation Protocol (SIP) number is a virtual Direct Inward Dial (DID) number that handles local or international calls online. SIP DID numbers can be used for inbound call routing and outbound SIP trunking services. SIP phone numbers include features like IVR, call recording, custom caller ID, advanced routing rules among others [9].
- **Local:** is a phone number that represents a specific region, city or state, identified by their 3 digit area code [9].
- **National:** is a phone number designated to be reachable from the entire country at the same price anywhere in the country as a local phone call [9, 15]. Unlike local phone numbers, which are tied to a particular city or region, national phone numbers are not tied to a particular local area.

## 2.2. Temporary Phone Numbers

In recent years, consumers have been overwhelmed by telemarketing and spam calls [18], robocalls [19] and by a binding personal phone number [20]. Hence the need to resort to a temporary number arises, for the most varied reasons. Temporary numbers have many benefits to the target audience, such as: [21, 22]

- Security
- Anonymity
- Ease
- Privacy
- Non-Binding

In technological terms, a temporary number is a virtual phone number used temporarily. Purchased from a service provider (see section 2.3), temporary numbers allow the same communication services as a regular phone number.

When a call is made or a message is sent to a temporary number, a forwarding process has to occur. This process consists of rerouting all the communications (calls and SMS) to the user's existing phone.

Nowadays, these numbers are provided through a mobile application or an online call forwarding service. The numbers are usually functional until a specific date or if the customer decides to cancel it.

### 2.2.1. Verification and Validation

Currently, the temporary numbers available on the market must comply with many rules before they can be made available to be purchased by the users. Many countries around the globe have been increasing their scrutiny of how phone numbers are used and by whom. The increase of incidents of misuse and abuse of phone numbers is the reason that new regulations to ensure better control of communications have been implemented.

In order to comply with the new regulations, providers (Carriers, CPaaS's, PBX's and others) are adopting a strategy that involves requesting extra documentation before a user purchases a temporary number. Through this strategy, the providers can ensure the phone number is linked to a real person. Thus, in the event of any misconduct, the user linked to that phone number will be held responsible.

The documentation requested of the users depends mostly on the provider and country. For personal phone numbers is usually requested: a Government-issued ID, Residence permit or Passport [23, 24]. For business phone numbers is commonly requested: an excerpt from the commercial register or a government-issued ID showing tax number.

The temporary phone numbers supplied by the providers such as Carriers, CPaaS's and PBX's must meet the legal requirements before being made public for purchase. On the other hand, apps must comply with conditions established by providers in order to use temporary phone numbers from them that will then be made available to consumers.

### 2.2.2. Communication Process

There are three ways of processing communications using temporary numbers: [25]

- **VOIP:** Instead of using a phone line, communications made through VOIP run over an encrypted internet connection. The user's main phone line is not forwarded, since only an internet connection is used. A permanent internet connection is required, and in case of poor wifi connection, the call/text will fail. In those cases, the apps have the possibility to change between user carrier connection and in-app calling.
- **Bridge Dial:** With this process, when the user wants to make a call, the app/provider calls their real number first. When the user answers, the app/provider calls the recipient with the user's temporary number as caller ID. This means that the user will use their real number for the connection, but the recipient only will see the temporary number as caller ID.



- **Forwarding:** Communications (calls and texts) are redirected through a temporary number. In this case, when the user wants to make an outgoing call to a person, a call from the user's real phone number is initiated/made to the temporary number and then is redirected from the temporary number to the person they want to contact. On the other hand, incoming calls received by the temporary phone number are redirected to the user's real phone. In this case, the user will see their temporary phone number as caller ID.

In all of the modes described, the temporary number can be considered as a middle agent since it adds an extra layer between the caller and the recipient allowing security and privacy of communications.

### 2.2.3. Recycling

Phone numbers are limited, so providers have to recycle unused numbers. When a phone number is not used during a period of time (typically 90 days) it is deactivated in order to be recycled and tested before being made available once more to be purchased by other users. With temporary phone numbers, the process is the same. [26, 27, 28].

The process can occur in case of cancellation by the user (the user does not want to use that number anymore) or cancellation by the provider (the user did not respect the terms of use or the service requested by the user expired).

### 2.2.4. Emergency Calls

When it comes to dialling an emergency service, it is required by law that every phone must allow to call the emergency number independently if it is bound or not to a carrier.

For convenience purposes, instead of repeatedly writing 'emergency number', this term will be replaced by '911'.

In case of temporary numbers, the providers have two options: allow the dialling to an emergency number using proper mechanisms and ensuring that it meets the requirements imposed by legislation or simply not allow it. Since it is a sensitive subject, most of the providers opt to not allow emergency calls to avoid legal problems related to fake calls or other types of illegalities.

The suppliers that allow emergency calls through temporary numbers use the following mechanisms: [29, 30, 31, 32]

- **Enhanced 911 (E911):** The temporary phone number and specified 911 address are sent to the local emergency centre and emergency operators have access to this information in order to send help or call back to the user if they need.
- **Basic 911:** With Basic 911, the local emergency operator that answers the user call may not see the temporary phone number and the specified 911 address. In this case, the user must be prepared to provide emergency services with this information. Until this information is received by the emergency responders, the dispatch of help or a call back is not possible or if the call is disconnected. However, once the local emergency centres are capable of receiving the information, the user is automatically upgraded to the Enhanced 911 service. Even so, the entities that allow emergency

calls force the users to specify a 911 address to ensure the availability of emergencies services at the location of where that phone number is being used. Then, how the 911 services are provided to the user can differ, depending on user location and the device (mobile or landline).

- **National Emergency Call Center:** Service provider owns national emergency center that in first instance will ask user for name, telephone number and location and then transfer the call to the local emergency center that is closest to them.

In spite of this, providers may supply this service only in some of the countries where they act; therefore, these policies may vary according to country.

### 2.2.5. Privacy and Traceability

The temporary phone number allows privacy, but there are some exceptions. In case of any illegal activity, the law enforcement authority can trace and identify who is using or used a certain temporary phone number.

Another example is if the user is using an app or a provider that provides calls in bridge dialling mode or for forwarding incoming calls to their real number, so if the user misses a phone call, it has the potential to hit the user's voicemail, exposing their identity. One other example could be through accessing the user's phone bill, where the calls made/received through the temporary number appear as calls made from/to the personal to the temporary number. Lastly, if the user uses a service provided by VOIP, only the data usage will show up. Messages sent/received through disposable numbers could stay all in-app, allowing no record elsewhere. [25, 33]

As mentioned before, the temporary phone numbers are not a silver bullet in terms of privacy, but they allow to add an extra layer of protection between the user and the people who would contact them, and vice-versa.

### 2.2.6. Binding and Billing

One of the biggest questions of the users could be "Do I have to sign a binding contract?". A significant advantage is the possibility of buying a temporary phone number without having to worry about the binding process. The only process that could be considered as "binding" is the period during which the user owns a specific phone number. During that time, the user is responsible for the way they use it. Other than that, no further binding contracts are needed.

In terms of billing, the patterns seen on the market are diverse but share the same core concept: the user only pays for the time they desire to use a temporary phone number. The methods of paying for a service are generally through a subscription per month in which each number has a set amount of minutes and messages available to use. After the minutes are spent, users can buy complements or even buy or upgrade to different plans.

## 2.3. Service Providers

A service provider is "a vendor that provides IT solutions and/or services to end-users and organizations." [2]

In the context of this internship, a service provider will be an entity that provides virtual phone numbers and communication services(e.g. calls and messages). Communications Platform as a Service (CPaaS)'s, Carrier's and Private Branch Exchange (PBX)'s are examples of different types of providers. Each one of them provides services and numbers but with different technologies and mechanisms.

Subsequently are presented the types of providers mentioned earlier as well as some examples.

### 2.3.1. PBX Providers

PBX [34] is a system that switches calls between enterprise users on local lines. It is owned and operated by the enterprise rather than the telephone company. This type of system offers multiple inbound and outbound lines, call routing, voicemail, and call management features.

Below are presented some examples of popular PBX providers [35].

#### Ring Central [36]

RingCentral is a publicly-traded provider of cloud-based communications. Founded in 1999, RingCentral is considered the leader in Unified Communications as a Service in terms of revenue and subscriber seats.



#### Vonage [37]

Founded in 2001 and headquartered in Holmdel Township, New Jersey, Vonage is a cloud communications provider based on voice over Internet Protocol.



#### Atlantech [38]

Founded in 1995, Atlantech Online is one of the oldest telecommunications providers. The company delivers Cloud PBX and Unified Communications tools on a Broadsoft platform.



### 2.3.2. CPaaS Providers

CPaaS is a cloud-based delivery model. This model allows companies to add capabilities such as voice, video and messaging, to business applications in real-time [39]. CPaaS capabilities are provided by deployed application program interfaces (APIs).

Below are presented some examples of popular CPaaS providers [40].

**Twilio** [41]

Twilio is a CPaaS founded in 2008 by Jeff Lawson, Evan Cooke, and John Wolthuis, headquartered in San Francisco, California. Twilio is a developer platform for communications that provide voice, video and messaging capabilities through deployed APIs.

**Nexmo** [42]

Founded in 2010, Nexmo is a unit of Vonage created to deliver communication APIs which enable developers and businesses to improve and innovate the way they communicate with customers.

**Plivo** [43]

Founded in 2011 by Michael Ricordeau and Venky Balasubramanian, Plivo started as an open-source telephone project. Plivo is a voice and messaging platform created to deliver better customer experiences to businesses.



### 2.3.3. Carrier Providers

In the context of cellular technology, a Carrier[44] is a wireless service provider that makes it possible to connect the calls from customers. This provider is authorized by regulatory agencies to operate telecommunications systems.

Below are presented some examples of carrier providers.

**AT&T** [45]

Founded on October 5 of 1983, AT&T Inc. is an American multinational conglomerate headquartered at Whitacre Tower in Downtown Dallas, Texas. It is considered one of the largest provider of mobile telephone services.

**Vodafone** [46]

Founded in 1982, Vodafone is an English multinational mobile operator headquartered in London. Vodafone is a successful company of technology communications through mobile, fixed, broadband, TV and voice services.

**Verizon** [47]

Founded in 2000 and headquartered in Basking Ridge, Verizon is a telecommunications company which offers wireless products and services.



This chapter presented some of the key concepts related to the internship, and following will be presented the state of the art.

This page is intentionally left blank.

## Chapter 3

# State of the Art

This chapter, organized into two sections, exposes the current state of the art. The purpose of this chapter is to present the result of the research done by the intern during the first semester.

Section 3.1 presents an analysis of the possible competitors of the product to be developed in WIT. Section 3.2 explains the current state of the market through the identification of key market elements obtained through the analysis of the competitors.

### 3.1. Competitors

This section will cover and report an analysis of market applications that allow any user to purchase and use temporary phone numbers. Through this analysis, the intern identified the strengths of each application in order to understand which features can add value to the solution to be developed by WIT.

The author started by defining the criteria for choosing the applications to be tested because, in a vast market, it is crucial to understand which niche of the market fits the product best.

The criteria defined were:

1. The application provides temporary numbers;
2. The application allows to make/receive calls;
3. The application allows to send/receive messages;

Given the criteria defined above, the author searched online for applications. These are shown below, and their analysis is present in subsection 3.1.2.

### 3.1.1.1. Examples of Competitors

#### Hushed

**Platforms:** Android and iOS

**Play Store Rating:** 3.7/5 stars (22 900 ratings)

**App Store Rating:** 4.4/5 stars (485 ratings)

**Android Compatibility:** Android 5.0 and up

**iOS Compatibility:** iOS 10.0 or later

**Installs:** 7,000,000+

**Link:** [www.hushed.com](http://www.hushed.com)



Hushed is a mobile app created by Affinityclick for Android and iOS that allows people to buy a temporary phone number for short or long term with an encrypted messenger. Affinityclick is a mobile company focused on the telephony and communication sector. Founded in 2010 Hushed offers private labelled VOIP services, SIP and DIDs and other related services [48].

#### Main Features:

- Calls
- Messages
- Voicemail
- Integrations with slack and dropbox
- Toll-free Numbers
- Do not Disturb
- Quiet Mode
- Start up Screen

#### Line2

**Platforms:** Android, iOS, Mac and Windows

**Play Store Rating:** 3.1/5 stars (13 718 ratings)

**App Store Rating:** 3.9/5 stars ( 2100 ratings)

**Android Compatibility:** Android 5.0 and up

**iOS Compatibility:** iOS 10.0 or later

**Installs:** 1,000,00+

**Link:** [www.line2.com](http://www.line2.com)



Line2 is a mobile application developed by Line2 that is a telecommunications company founded in San Francisco in 2008 by Peter Sisson. The company is best known for Line2 apps, which provides Wi-Fi support for mobile phones and multiple devices instead of using the service provider. Line2 is a Virtual PBX phone service that allows companies to develop and maintain multiple internal phone lines to handle large numbers of external calls [49].

#### Main Features:

- Choose the number
- Toll-free number
- Wi-fi calling
- Auto attendant
- Call blocking
- After-hours settings
- Caller ID
- Vanity numbers
- Call screening
- Conference calling
- Call forwarding

## Cloud Sim

**Platforms:** Android and iOS  
**Play Store Rating:** 3.7/5 stars ( 585 ratings)  
**App Store Rating:** 4.0/5 stars ( 48 ratings)  
**Android Compatibility:** Android 5.0 and up  
**iOS Compatibility:** iOS 10.0 or later  
**Installs:** 1,000,00+  
**Link:** [www.cloudsimapp.com](http://www.cloudsimapp.com)



Cloud Sim Telecoms Ltd is a company from UK that developed the Cloud SIM mobile application. Available on Android and iOS Cloud SIM is an application that can make and receive calls across the world, giving the user full control over his communications [50].

### Main Features:

- Extra mobile numbers
- Free chat
- Low cost calls
- Messages
- Extra international numbers
- International numbers
- Personalised voicemail
- Customise Cloud Sim profile
- Decide when you are available
- Free calls

## iPLum

**Platforms:** Android and iOS  
**Play Store Rating:** 3.1/5 stars ( 13 718 ratings)  
**App Store Rating:** 3.9/5 stars ( 2100 ratings)  
**Android Compatibility :** Android 5.0 and up  
**iOS Compatibility :** iOS 10.0 or later  
**Installs:** 1,000,00+  
**Link:** [www.iplum.com](http://www.iplum.com)



Founded in 2015 by people working in Silicon Valley, iPlum is a mobile application that provides a separate second phone line for US, Canada or 800 Toll-free numbers. Using the second line phone users can call or text people around the world.

### Main Features:

- Business hours
- Auto text reply
- IVR/Virtual Digital Assistant
- Text Archiving, Backup
- Block spam numbers
- Ringtones
- Signature text
- Call recording
- Call transfer
- Do not disturb
- Could voicemail with email alert
- New number or port the existing number



## Sideline

**Platforms:** Android and iOS

**Play Store Rating:** 3.2/5 stars (52 289 ratings)

**App Store Rating:** 4.1/5 stars ( 4400 ratings)

**Android Compatibility :** Android 5.0 and up

**iOS Compatibility :** iOS 10.0 or later

**Installs:** 1,000,000,0+

**Link:** [www.sideline.com](http://www.sideline.com)



Sideline is an iOS and Android app that joins a temporary number to the user's smartphone. Developed by Pinger, Inc., a US Telecom provider, Sideline is an app which allows messages, pics, calls and voicemails. The company was founded in 2006 in San Jose, California [51, 52].

### Main Features:

- Number porting
- Group messaging
- Separate caller ID
- Voicemail to text
- SMS+MMS messaging
- Custom voicemail
- VoIP optional
- Auto-reply
- Branded messages
- Use of carrier minutes

## Swapp

**Platforms:** Android and iOS

**Play Store Rating:** 3.3/5 stars ( 71 ratings)

**App Store Rating:** 2.8/5 stars ( 43 ratings)

**Android Compatibility :** Android 5.0 and up

**iOS Compatibility :** iOS 10.0 or later

**Installs:** 5000+

**Link:** [www.swapp.pt](http://www.swapp.pt)



Swapp is a mobile application, developed by Meo that provides a temporary phone number to any person independently of their carrier or subscribed plans. MEO is a mobile and fixed telecommunications service created in 2007 after the separation of PT Comunicações and PT Multimédia. [53, 54].

### Main Features:

- SMS
- Caller ID
- Calls
- Block numbers
- Voicemail
- Choose the number
- Edit contacts information
- Use of carrier minutes

### 3.1.2. Competitors Analysis

During the first months, the intern studied and tested the solutions available on the market in order to understand their functioning. For this analysis two devices were used, a smartphone Xiaomi Mi A1 (with the android operating system) and an iPad (with the iOS operating system). Table 3.1 represents the result of the analysis made to the major competitors.

Features	Hushed	Line2	Cloud Sim	iPlum	Sideline	Swapp
Choose a Number	Yes	Yes	Yes	Yes	Yes	Yes
Cancel a Number	Yes	Yes	Yes	Yes	Yes	Yes
Extend a Number	Yes	Yes	Yes	Yes	Yes	Yes
Send Messages	Yes	Yes	Yes	Yes	Yes	Yes
Message History	Yes	Yes	Yes	Yes	Yes	Yes
Auto-reply Message	Yes	Yes	No	Yes	Yes	No
Group Messages	Yes	Yes	No	Yes	Yes	No
Make a phone call	Yes	Yes	Yes	Yes	Yes	Yes
Calls history	Yes	Yes	Yes	Yes	Yes	Yes
Call Routing	Yes	No	No	No	No	No
Call Forwarding	Yes	Yes	Yes	Yes	Yes	No
Call Recording	No	No	No	Yes	No	No
Caller ID	Yes	Yes	Yes	Yes	Yes	Yes
Send Calls to Voicemail	Yes	Yes	Yes	Yes	Yes	Yes
Voicemail History	Yes	Yes	Yes	Yes	Yes	Yes
Change Voicemail greeting	Yes	Yes	Yes	Yes	Yes	Yes
Record a Voicemail Greeting	Yes	Yes	Yes	Yes	Yes	Yes
Voicemail Transcription	No	Yes	No	No	Yes	No
Import Contacts	Yes	Yes	Yes	Yes	Yes	Yes
Create Contact	Yes	Yes	Yes	Yes	Yes	Yes
Edit Contacts	Yes	Yes	Yes	Yes	Yes	Yes
Block Phone Numbers	Yes	Yes	Yes	Yes	Yes	Yes
Dot not Disturb	Yes	Yes	Yes	Yes	Yes	Yes
Notifications	Yes	Yes	Yes	Yes	Yes	No
Business Hours	Yes	Yes	No	No	No	No
Support	Yes	Yes	Yes	Yes	Yes	Yes

Table 3.1: Competitors Analysis

There are many options available on the market, and they are directed towards providing communications via Wi-Fi calling or VoIP. The services provided to the user do not vary much from application to application, as table 3.1 shows. Through this analysis, the intern was able to understand the basic/standard features to be inserted in the MVP but also to recognise which ones are less common and can be used to provide a better value proposition.

In chapter 5 are detailed the requirements that resulted from this analysis.

## 3.2. Market Vision

Before starting to develop any product, it is essential to understand the current state of the market and what to expect will happen in near the future. Defining a market vision for new products can "allow companies to begin to focus on developing and emphasising key non-technical resources and competences that are linked to achieving success when developing radically new products in highly uncertain technological and market environments". [55]

Although the internship's goal is to develop only a functional MVP of a possible solution for the stated problem, gathering data about strategies and practices used in the current market by the competitors can be useful to define and implement a better MVP or even help the company establish a formal market vision. Bearing this in mind, after analysing the competitors, the author identified not only the main features (mentioned in the section before) but also some key elements present in them. These fundamental elements can be useful because they represent the actual strategies used on the market by the providers. In the subsection below, the key elements identified by the author are explained.

### 3.2.1. Market Key Elements

- **Phone numbers per device:** Most providers don't limit the number of temporary phone numbers that a user can have. The user is free to have one, two or more phone numbers as long as they have the funds to maintain them. The temporary numbers can be cancelled at any moment.
- **Types of temporary phone numbers provided:** The most common temporary numbers provided are local, national and mobile. Each phone number can have restrictions in terms of capabilities (SMS and calls) or location.
- **User's personal information provided:** From the apps' analysis, only a valid Sim card was found to be necessary to complete the registration, no further documentation was required. However, due to the new regulations imposed by the governments more documentation could be requested in the future.
- **Geographical availability:** Most of the apps offer phone numbers from many countries around the globe (e.g. Portugal, USA, Spain, Venezuela, Australia and many more). The communications made between phone numbers of the same region have no additional cost. Calling or messaging phone numbers from other countries/regions can have an additional cost.
- **Billing mechanisms:** From the providers analysed, the billing mechanism is found to consist in subscriptions of time (week, month or year) with a specified number of minutes and SMS available to spend.

In this chapter was made the state of the art, were a competitors analysis was done, and the next chapter will present the methodology used during the internship, as well as the work plan.

## Chapter 4

# Methodology and Work Plan

In this chapter are documented and explained the overall decisions made by the author related to project management. The section 4.1 presents the methodology used during the internship. Subsequently, section 4.2 explains the work plan and work done in two semesters. Lastly, section 4.3 presents the project risks's analysis.

### 4.1. Methodology

Project management is crucial because it establishes the principles and processes to adopt during project development. Choosing the right methodology is usually a hard task because the project team should analyse and discuss which methodology is more suited for a project.

Since the methodology preferred by the software development teams at WIT is Scrum, the intern was warned *a priori* that this would also be the methodology to use during his internship.

With that in mind, the subsection 4.1.1 contextualizes the methodology used and the subsection 4.1.2 explains how the chosen framework was applied to the project.

#### 4.1.1. Scrum Framework

According to the Scrum creators Ken Schwaber and Jeff Sutherland, Scrum "is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value"[56]. Much like any other methodology, SCRUM is composed by phases where each one has its specific purpose:

#### Scrum Phases [57]

- **Pregame:** is made up by the planning and architecture phases. In planning is defined the Product Backlog that consists in the functionalities of the project to be implemented, along with an estimate of its schedule and costs. In the architecture phase, it is designed how the backlog items will be implemented. Therefore, this phase includes the designing of the system's architecture and the high level design.

- **Game:** occurs the implementation of the functionalities set in the Product Backlog, always respecting schedules, quality costs and the competition. The implementation is done in an interactive way trough sprints that are used to evolve the system in development.
- **Postgame:** the preparation to lauch the finished product is done, including all the final documentation.

As aforementioned, these three phases make up how a project that uses this methodology should be developed. Out of these phases, the Game has the most relevance because it encompasses the implementation process for the functionalities of a certain product. For this reason, it comprehends the execution of several cycles. Each life cycle is composed by roles, artifacts and events. In Figure 4.1 the Scrum process is illustrated.

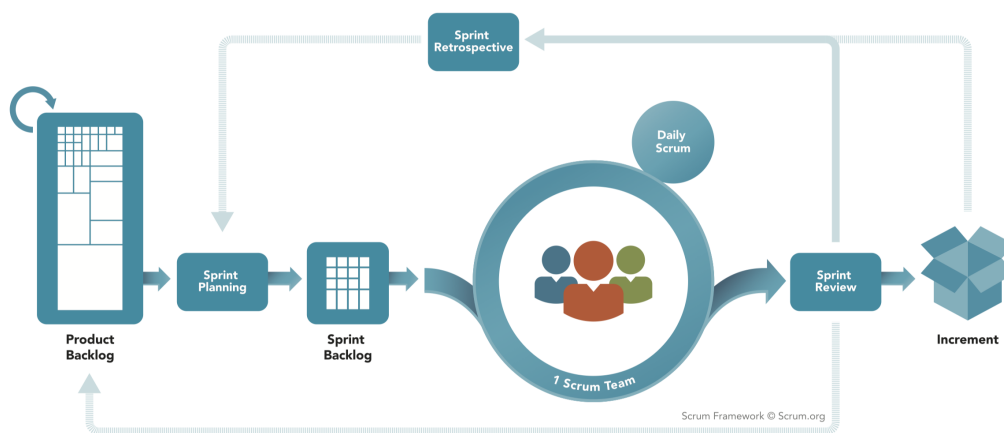


Figure 4.1: Scrum life cycle [58]

### Scrum Roles [56]

- **Product Owner:** the person responsible for managing the Product Backlog and responsible for maximizing the value of the product in development.
- **Scrum Master:** the person responsible for helping everyone involved in the project understand Scrum theory, practices, rules, and values. The Scrum Master tries to maximize the value created by the Development Team.
- **Development Team:** individuals who do the work of implementing the functionalities of the Product Backlog with the purpose of delivering a potentially releasable Increment of "Done" product at the end of each Sprint. The regular size of a standard Development Team is usually seven individuals plus or minus two.

### Scrum Artifacts [56]

- **Product Backlog:** ordered list of features that are known to be needed in the product, which is never complete. Content additions, removals or updates could be made to the project, so it is essential to know that the Product Backlog is not static. Each item from the list could be ranked, meaning that some items must be

implemented first than others. The Product Owner is responsible for the Product Backlog.

- **Sprint Backlog:** the set of Product Backlog items selected for being developed by the Development Team in the Sprint.
- **Increment:** list of Product Backlog items developed by the Development Team during a Sprint. When an item from Product Backlog is developed, it is considered "Done" and everyone must understand what "Done" means. The definition of "Done" can vary per Scrum Team.

### Scrum Events [56]

- **Sprint:** period of time of one month or less during which, a Development Team works to create an Increment. For this internship, sprints of 2 weeks will be used.
- **Sprint Planning:** period of time in which the Scrum Team defines the work to be performed during a sprint.
- **Daily Scrum:** daily meeting of 15 minutes to inspect and plan the next 24 hours of work. The objective is to ensure that progress is trending toward completing the work in the Sprint Backlog. Since the team is comprised only by the intern, and the supervisors are in charge of other projects, these meetings will occur weekly.
- **Sprint Review:** At the end of each sprint, a Sprint Review is carried out. An inspection of the Increment is done, and Product Backlog could be adapted if needed.
- **Sprint Retrospective:** Sprint Retrospective is done for the Development Team to understand what went well, what could be improved, and what they will commit to improving in the next sprint.

Understanding all these characteristics, Scrum can be considered a model designed to "optimize flexibility, creativity and productivity" [59] during the execution of a project.

Although it was mentioned that the intern would follow Scrum methodology, an adaptation took place because project characteristics (e.g. team size and pandemic situation) made it clear that following this methodology fully would be impossible. In order to clarify the adaptation made, the subsection below explains all the stages and processes performed based on the Scrum methodology.

#### 4.1.2. Scrum Project Adaptation

The Scrum Team for this project was composed by Paulo Sousa, Head of New Product Development at WIT Software as a Product Owner, Jorge Sousa and João Costa as Scrum Masters and the author as a team member of the Development Team.

The Product Backlog was defined as being composed of all the requirements established during the first semester. Subsequently, a Sprint Planning took place so that the requirements to implement could be defined and then developed during the sprint. Each sprint had a duration of two weeks, and a Daily Meeting took place between the intern and the Scrum Master, João Costa, to keep track of the progress done. At the end of each sprint, a Sprint Review was not supposed to happen; instead, a monthly meeting with the Development Team was planned. Due to the pandemic situation and the work overload of the

Product Owner and the Scrum Masters, this was not possible every month. This sequence of events repeated itself until all the requirements that make up the Product Backlog were implemented.

Concerning the management of the sprints, the used software was Redmine, in accordance with the Scrum framework.

## 4.2. Work Plan

In this section, an overview of the work plan that was followed during the two semesters of the internship is presented.

### 4.2.1. First Semester

During the first semester, the so called pregame was made. In this phase of the Scrum methodology, the intern realized the goal of the project and found the list of functionalities needed to correspond to the needs set by WIT. For that purpose, a meeting was made with the Project Owner.

After deciding the scope of the project and requirements to develop, the intern developed a standalone application to prove that the provider suggested by the supervisors during the meetings was viable for this purpose. Then, the next step was for the intern to start trying to figure out which components would integrate the system and its relations. In this sense, he proceeded to the specification of the project's architecture.

During the first semester there were no major problems and the deadline for every task was met. The Gantt chart below (figure 4.2) shows the task plan for the first semester.

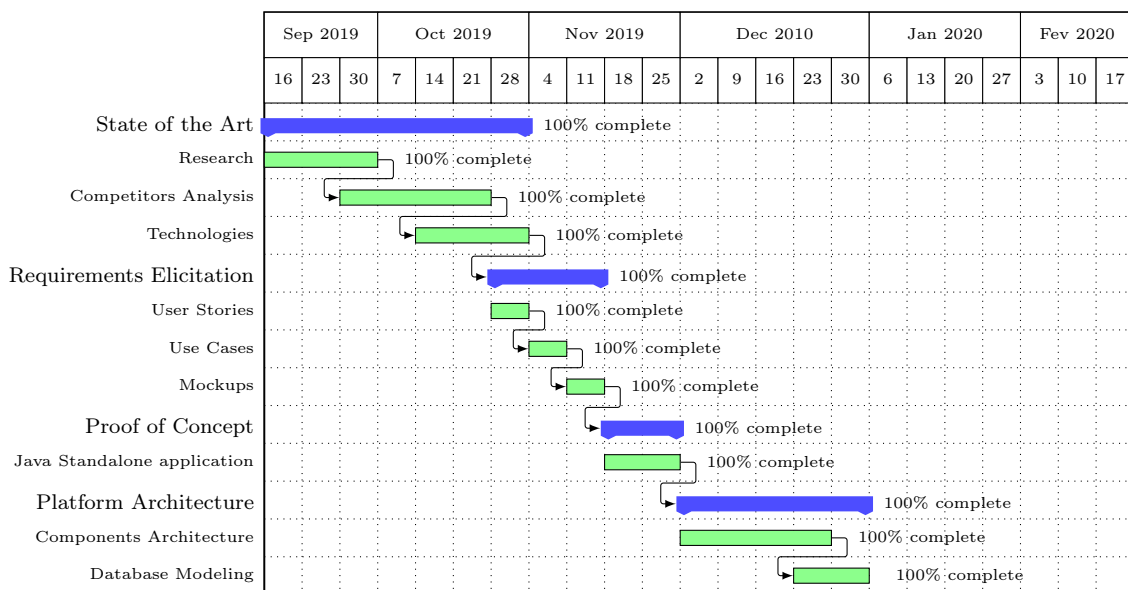


Figure 4.2: Gantt chart for the first semester

### 4.2.2. Second Semester

During the second semester proceeded the Scrum Game phase. In this phase, the author began to implement the functionalities of the Product Backlog, always trying to uphold the schedules.

Although the Scrum methodology generally does not follow a plan generated by a Gantt diagram, two were created in order to identify the work plan for the semester and the work done during the semester.

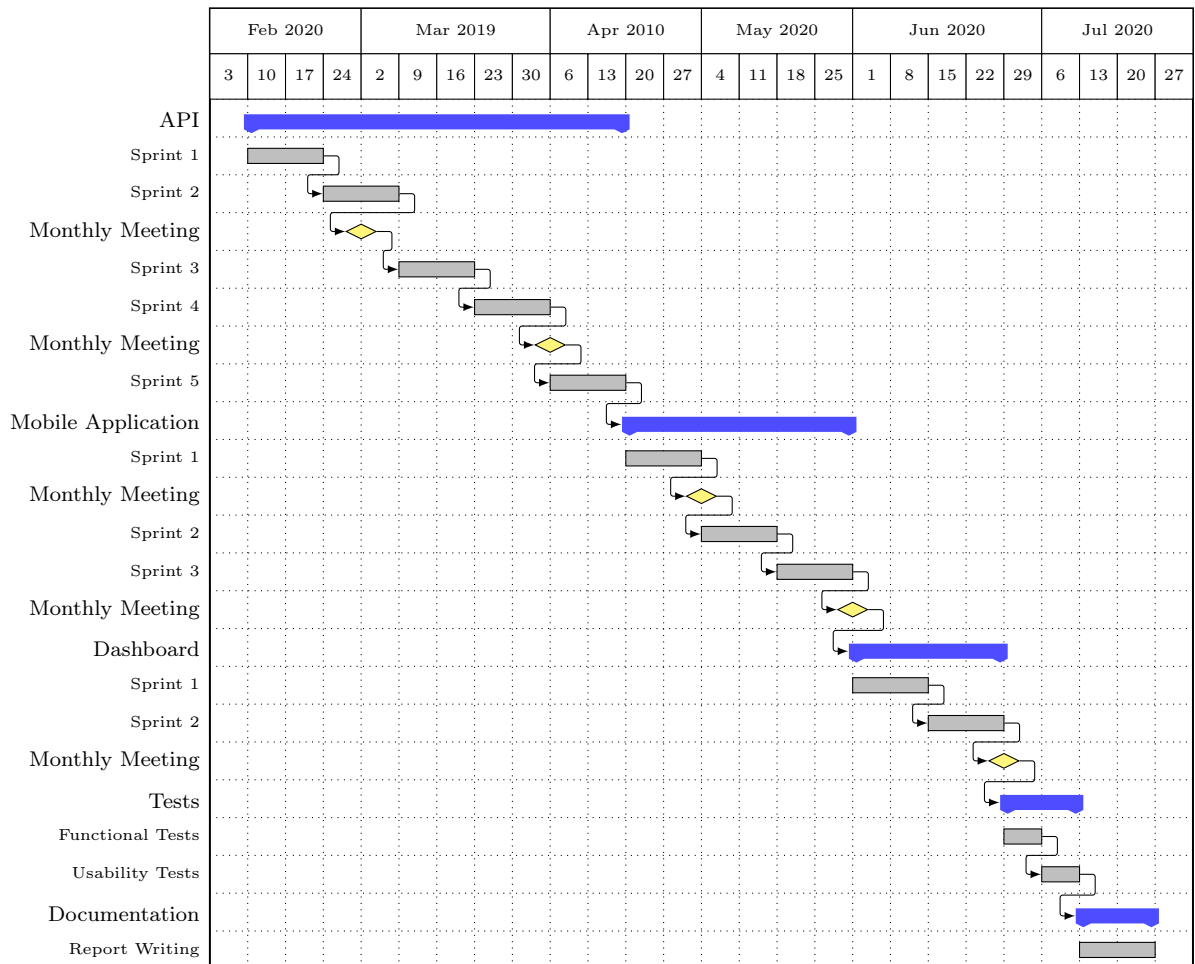


Figure 4.3: Planned Gantt chart for the second semester

The Gantt diagram presented above shows how the intern planned the work for the second semester. However, as expected in any agile software development project, the plan is merely indicative and was adjusted according to the project needs.

One of the first changes occurred during the first week of API development, when the company decided that the Mobile App should be implemented first. Therefore, the sprints were adjusted accordingly. The next setback was the pandemic situation (COVID-19), leading the intern to work remotely from the beginning of March onwards, bringing along some difficulties related to communication and productivity. Along with this, the lack of experience of the intern regarding React Native brought out successive delays affecting the sprints, due to the need to search and learn new concepts and methods to implement the needed features. This particular issue brought consequences to the whole project as, by the end of May, it did not uphold all the requirements necessary to be a functional



prototype. At that point, the intern, the supervisors from WIT, and the tutor from DEI, agreed that the best option was to postpone the final delivery to September so that the expected functional prototype could be delivered. The agile approach revealed important benefits in this particular situation, allowing constant adjustments and adaptations to the work plan.

The postponement of the final delivery resulted in the Admin Dashboard component (Web Portal) not being implemented since nothing had been developed to date, and there were no mockups re-created by WIT 's designers. As the curricular internship ended on June 30, it was decided that the trainee would until then implement the missing features related to the Mobile Application.

The project's complexity and the intern's lack of experience in the technologies also had an impact in the planning process and in the impossibility to deliver a prototype containing all the components (API Backend, Mobile Application and Admin Dashboard).

Despite these problems, the trainee always kept high levels of motivation and made a constant effort to adjust the project over time. Postponing the final delivery to present a better prototype was a natural step to attain the expectations of WIT and learn as much as possible during the internship. Figure 4.4 illustrates the main events and processes that took place during the second semester.

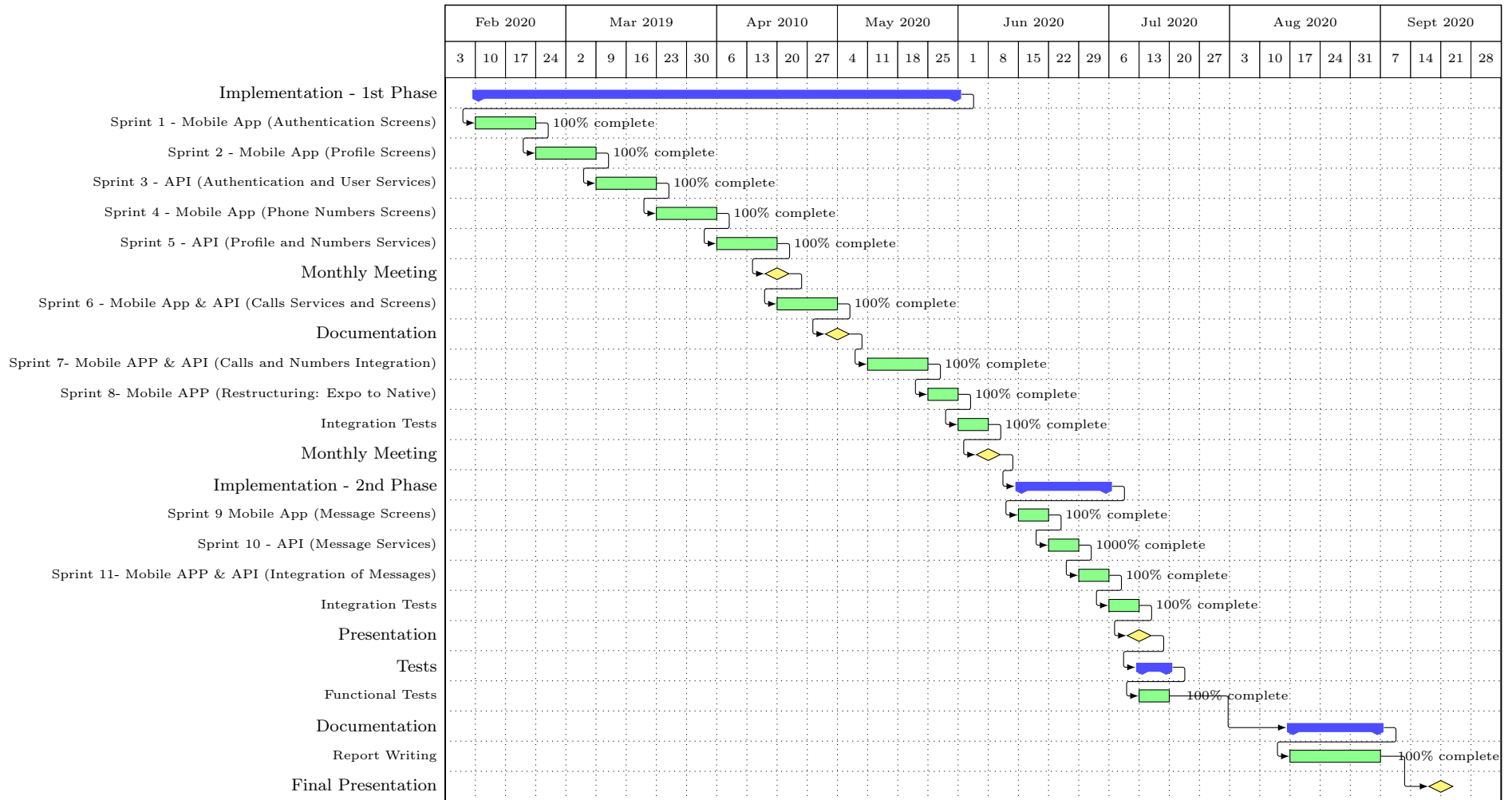


Figure 4.4: Gantt chart for the work done in the second semester

## 4.3. Risk Management

This section explains the process of analyzing and defining risks for this internship. First, the concept and the importance of risk management is explained. Secondly are described the risks found throughout the internship. Lastly, the verified risks during the internship are explained.

### 4.3.1. Concept of Risk Management

Any project is subjected to failure. Since the beginning, a project has a set of risks attached that if not identified and monitored can have a big impact on project development. Risk is defined as "an activity or event that may compromise the success of a software development project"[60]. In order to avoid and minimize these problems, it is necessary to do risk management. Risk Management is "a methodology or a mechanism, carried out throughout the development process to identify, manage and control risks evolved before and during the development process"[61].

Risk management involves the following activities [62]:

- **Identification:** consists in identifying the factors that can compromise the success of the project. Risk identification is made throughout the project's life cycle since the project is not immutable, and changes can occur.
- **Analyzing:** involves analyzing how project outcomes and objectives might change due to the impact of the founded risks. In order to analyse the risks, some attributes are used. The most common are the probability of its occurrence and the impact of the risk on the project. To each one of the attributes, metrics need to be defined.

The probability of the occurrence of a risk can have the following values:

- **High** - probability of happening is higher than 80%.
- **Medium** - probability of happening is between 30% and 80%.
- **Low** - probability of happening is below 30%.

The impact of the risk on the project can be represented through the following values:

- **High** - has a big impact on project schedule or performance. Success goals may not be achieved.
- **Medium** - has a significant impact on project schedule or performance. Success goals may be achieved but with extra effort.
- **Low** - has no impact on development time or performance. Success goals are not affected and can be achieved easily.

It is also in this activity that is concluded if a risk can be mitigated or eliminated.

- **Planing:** consists of creating risk responses to mitigate or eliminate the risks previously identified. To accomplish that, a plan must be created.
- **Monitoring:** iterative process that consists of keeping track of existing risks, identifying new ones, risks reclassification and reporting throughout the project development.

After the identification and evaluation of the risks, it is possible to display the results in the form of a risk matrix. A risk matrix is "a practical and easy to use tool, which can help most organizations"[63] in many ways (e.g. prioritizing risks and promoting discussion between team elements).

A project matrix example is represented in Figure 4.5. It is formed by nine categories, each one defining a level of danger of the risk. The green colour represents the risks that are negligible or marginal. Yellow identifies the risks that are neither catastrophic nor negligible and red represents the risks that are critical or catastrophic.

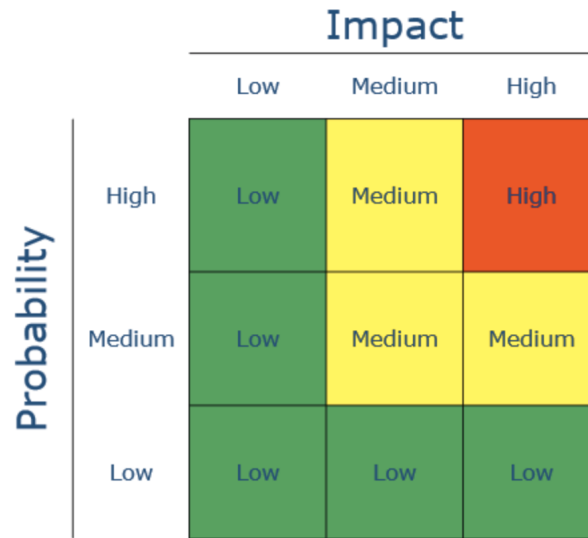


Figure 4.5: Risks Matrix [64]

### 4.3.2. Risks Description

In this section, the project’s risks are described. As risk management must be done periodically, this section contains all the risk identified during the internship. Each risk was rank according to its probability of occurrence, the impact on the project and has a mitigation plan.

Below is presented an example of a risk description, the remainder are in Apendix B.

Project Complexity	
<b>ID</b>	R01
<b>Date of Identification</b>	November
<b>Description</b>	There is nothing implemented, so the intern has to understand how the platform should be designed, structured, and built. This is, therefore, a time-consuming task because the author does not have any knowledge about the internship topic.
<b>Impact</b>	High
<b>Probability</b>	High
<b>Mitigation Plan</b>	Keep in contact with the supervisors by scheduling meetings to receive helpful information and feedback on how the platform must be implemented.

Table 4.1: Risk 01 - Project Complexity

After the risks description, each risk was inserted into the projet's risks matrix. The result is displayed in figure 4.6.

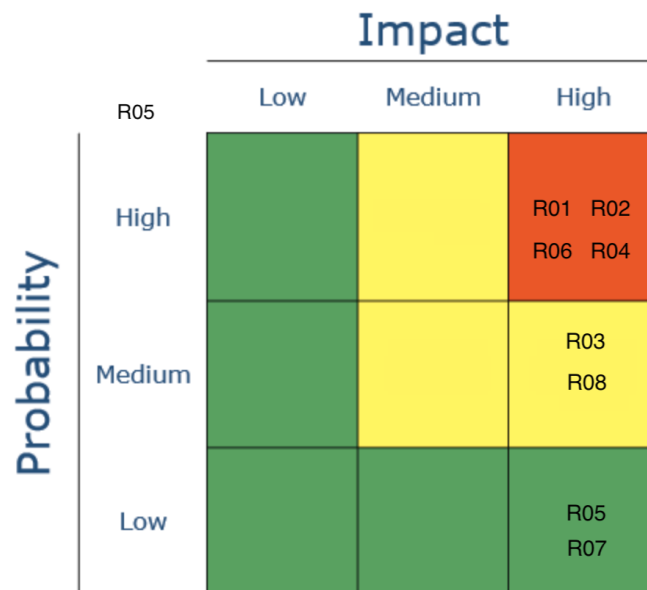


Figure 4.6: Risks Matrix of the project

As can be seen from the figure 4.6, the risks fall in the yellow and red zone. This led the author to conclude that the project has hazardous risks that should be avoided through good mitigation plans. Constant monitoring should take place in order to eliminate/control the known risks and identify new ones.

In the subsection below are exposed the risks that have been verified as well the mitigation actions that took place during the development of the project to reduce their impact.

### 4.3.3. Risks Verified

During the internship, risk control was carried out in order to ensure that the risks were identified and mitigated. Notwithstanding this control, some risks occurred, meaning that the mitigation actions were not enough to prevent delays on the project. Of the risks identified, the R02, R03, R04 and R06 were the ones that had an impact on the project development.

Following a timeline, during the first semester, of the risks identified, there was no impact since the semester was more focused on research and know-how. However, during the second semester, the situation was reversed. It all began in February with the start of the solution implementation, when the lack of experience on React Native lead to successive delays. From the moment the delays began to be noticed by the intern, mitigation actions established for the risk R04 were put into practice. As a consequence of that, the delays started to get more noticeable because the intern had to spend time searching and reading information to solve the issues found during the development. It was at the end of May, one month before the delivery date that the risk R02 come to be verified. After introspection and verification of the work done, it was decided to postpone the final delivery to September due to the fact of the prototype's current state was not functional.

In addition to these two, risk R03 was also come about in mid-February after Twilio (the CPaaS provider) announced that due to legislations, the way of how phone numbers were provided would change. Since then, it became mandatory to provide the user's documentation to acquire phone numbers. In order to discuss how this change would affect the solution's actual design, a meeting took place to define the mitigation action to be used. As a result, it was decided that the intern would provide his personal information in order to be able to acquire phone numbers for the platform. This decision was made because a redesign of the current solution was not viable in terms of time and MVP purposes.

This chapter pours over the chosen methodology, summing up the phase of identifying and treating risks found throughout the project. In the next chapter will be introduced the requirements elicitation process.

This page is intentionally left blank.

## Chapter 5

# Requirements Elicitation

The requirements elicitation is one of "the most difficult, most error-prone and most communication intensive stages in software development"[65]. According to Raul Sidnei Wazlawick [66] "The process of elicit requirements consists of searching for information about the tasks that the system must perform, and for the constraints under which the system must operate". Many techniques can be used such as interviews, brainstorming, prototyping, and surveys/questionnaires. "The success of an elicitation technique used depends on the maturity of the analyst, developers, users and the customer involved"[65].

In this section, it is detailed the process of requirements elicitation for the Temporary Numbers Platform, along with the user stories. A detailed description of the requirements is available in Appendix A.

### 5.1. User Stories

One of the techniques used for defining the requirements of a project is by creating user stories. "User stories are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system"[67]. The user story describes the type of user, what they want and why. This type of tool helps to create a simplified description of the requirement and it can be inserted in the agile frameworks. The user stories do not replace use cases or other technical requirements. They typically abide by the following template:

*"As a <type of user>, I want to <goal> so that <reason>"* [68]

Several meetings were scheduled with the product owner in order to identify the project's user stories. During those meetings, the Temporary Numbers Platform major services were discussed. Table 5.1 presents the user stories defined for the project that resulted from the meetings with the product owner.



User Stories			
ID	"As a"	"I Want to"	"So that I"
US1	User	Purchase a number	Can communicate with other people
US2	User	Make calls	Can communicate with other people
US3	User	Import my contacts	Can communicate with other people
US4	User	Send Messages	Can communicate with other people
US5	User	Subscribe to a Plan with minutes and SMS	Am able to use the service
US6	User	Check the minutes and SMS used	Manage my minutes and SMS
US7	User	Add funds to my account	Can subscribe to plans
US8	User	See my messages history	Be updated
US9	User	See my calls history	Be updated
US10	User	Cancel my account	Can uninstall the app
US11	User	See my numbers	Can know which numbers i own
US12	Admin	Cancel a temporary phone number	Can ensure that number is not used by the user
US13	Admin	Add/remove money to/from a user balance	Can ensure user balance is up to date
US14	Admin	Get statics of the system	Can analyze the system behaviour
US15	Admin	Send an Alert/Notification	Can ensure the user is informed and updated
US16	Admin	Re-activate an Account	Can ensure the user will be able to use the service again
US17	Admin	Suspend an Account	Can ensure the user temporarily can't use the account to access the services
US18	Admin	Close an Account Permanently	Can ensure the user will not use the account to access the services

Table 5.1: Users Stories.

## 5.2. Functional Requirements

As aforementioned, expressing requirements in the form of user stories helps us to understand the "raw" user needs. After that, it is necessary to define how the software will answer those needs. There is no standard template when it comes to describing requirements found through user stories, but most of the times, use cases are used. A use case is "a written description of how users will perform tasks" [69] on the software. Through use cases, it is possible to describe the behaviour of the system in order to answer those user "needs".

In order to achieve that, the intern followed WIT's template for documenting the system's functional behaviour. The template used follows the MoSCoW [70] nomenclature to prioritize the functionalities to be implemented, which goes as follows:

- **Must have:** requirements that are critical for the success of the project.
- **Should have:** requirements that are important, but not mandatory for the success of the project.
- **Could have:** requirements that could be implemented to improve the system, but not mandatory for the success of the project.
- **Won't Have :** requirements that will not be included in this version of the project

The description of the requirements is based on Alistair Cockburn's template [71] with slight modifications. The fields chosen are:

- **Use Case Id:** unique identifier of the requirement.
- **Primary Actor:** who will trigger the requirement
- **Description:** a brief description of the requirement
- **Pre-conditions:** conditions that the system needs verified in order to start the use case
- **Main Success Scenario:** the steps that will occur if everything goes as expected (the ideal scenario)
- **Extensions:** the alternative steps that the system performs when one or more steps from the main scenario do not go as expected.

Table 5.2 represents an example of a functional requirement described as mentioned earlier. The remainder of the requirements are detailed the Appendix A.

### Example

FR01 - Registration	
<b>Primary Actor</b>	Non Registered User
<b>Description</b>	User enters their phone number in order to make the registry
<b>Priority</b>	Must have
<b>Pre-conditions</b>	User has the application installed User has access to an internet connection User has a valid phone number
<b>Post-condition success</b>	User registers and can start using the application
<b>Main Success Scenario</b>	<ol style="list-style-type: none"> <li>1. User inserts his phone number</li> <li>2. System verifies the inserted number</li> <li>3. The System sends a verification code (via SMS) to the user</li> <li>4. The user is redirected to verification screen</li> <li>5. User inserts the code received</li> <li>6. The system validates the code</li> <li>7. User is redirect to the main page</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>2a Number inserted by the user already exists               <ol style="list-style-type: none"> <li>2a1. User is informed that number is already registered</li> </ol> </li> <li>2b Number inserted is not valid               <ol style="list-style-type: none"> <li>2b1. Error message is shown</li> </ol> </li> <li>5a. Code inserted does not match               <ol style="list-style-type: none"> <li>5a1. Error message is shown</li> <li>5a2. The user tries again</li> <li>5a3. User presses a button to receive another code</li> </ol> </li> </ol>

Table 5.2: Functional Requirement 01 - Registration

### 5.3. Non Functional Requirements

In any software application, non-functional requirements (often called by quality attributes) are as critical as functional requirements. Nowadays, the current competitive market "demands software product which is not functionally fit but also implements the non-functional aspects such as: cost, readability, security, maintainability, portability, accuracy" [72].

Although the main internship goal consists in developing an MVP, it is important to notice that dealing exclusively with functional requirements could not be enough to fulfill the client demands. Due to that, "there is a need of formally incorporating NFRs into the software development life cycle and guaranteeing their satisfaction". [72].

Acknowledging the value of NFRs in software development, the intern, along with his supervisors identified the NFRs that better suit the project. To choose the most critical quality attributes, the intern assumed a critic point of view and analyzed possible tradeoffs between them.

The quality attributes defined are: [73, 74]

- **Availability:** is the ability of a specified system to mask or repair faults such that the cumulative service outage period does not exceed a required overvalue specified time interval. In the context of this project, this quality attribute is essential; however, due to the degree of external dependence entities, it is difficult to establish the degree of availability of the system.

For MVP purposes there is no need to define and detail that value, but it is worth to mention that for a real market product, an in-depth analysis of each external entity must be done (e.g. verifying the availability for each service used) to define a more precise value of system availability.

- **Security:** consists of resisting to unauthorized attempts to use/destroy data or services while still providing access to people and systems that are authorized. This quality attribute can be characterized by the following characteristics: confidentiality, integrity, and availability (CIA). In addition to these, there are others such as Authentication, Non Repudiation and Authorization that help to support the CIA.

For this project, the main characteristics that have been taken into account were the Authentication (responsible for verifying if the identities are truly who they claim to be), the Authorization (in charge for ensuring if the entity has the privileges to perform a task) and Confidentiality (responsible for ensuring that the data are protected from unauthorized access).

This quality attribute was chosen because one of the main goals is to build a system that is safe and does not put users communication's privacy at risk.

- **Interoperability:** is the degree to which two or more systems can exchange information via interfaces in a particular context. Not only includes the ability to transfer data (syntactic Interoperability) but also requires the ability to interpret the data exchanged (semantic Interoperability). Therefore, to handle Interoperability, we have two ways: If we already know the interfaces with which our system will operate, then we can apply this knowledge directly on our system design; If the external entities our system will communicate with are not yet ready then we design our system with

a more generic approach so that when they are available in the future, they can be bound in the life cycle, at build time or runtime.

Since the WIT Temporary Numbers Platform will use external systems with services already deployed and well documented then the design of the system will take care of all of this. The main reason that led the author to choose this quality attribute scenario was the fact that the platform services require capabilities from already existing systems and if there is no proper exchange of information, then our system could be compromised.

- **Modifiability:** is about changes that can occur to the system and how we can control the cost and risk of making those changes. In order to plan Modifiability the architect has to consider three questions: "What can change?", "When is the change made and who makes it?" and at last, "What is the cost of the change?".

This quality attribute was considered critical because, at any time, new features (e.g. add new temporary phone numbers providers, add video conference calls) could emerge to aggregate into the solution already developed. Therefore, it is crucial that the designed solution be prepared to support changes. Through this, it will be possible to avoid changes' high costs and better control the impact of these changes in the system.

A quality attribute requirement "should be unambiguous and testable"[75]. To ensure this property it is usually used a form to specify it. The form used in this report is based on the template presented in the book by Paul Clements, Len Bass and Rick Kazman [75].

The form is composed by the following fields:

- **Source of Stimulus:** is an entity (a human being a computer system, or any other actuator) that generated the stimulus. The stimulus can be an event to the performance community, a user operation to the usability community, or an attack to the security community.
- **Stimulus:** is a condition that requires an answer when one arrives at a system.
- **System Response:** consists of the responsibilities that the system (for runtime qualities) or the developers (for development-time qualities) should perform in response to the stimulus.
- **System Measures:** is the measurement taken when the response occurs, so that the requirement can be tested.
- **Environment Condition(s):** is the set of circumstances in which the scenario takes place. The environment acts as a qualifier on the stimulus.

Through the fields previously mentioned is created a collection of quality attribute scenarios that characterize the quality attributes defined. These scenarios must be system-specific to allow the translation of the generic attribute characterizations into requirements. Table 5.3 represents an example of a quality attribute scenario described. The remainder of the quality attributes scenarios are detailed in Appendix A.

**Example:**

NFR01 - Availability	
<b>Source of Stimulus:</b>	Internal to the system
<b>Stimulus</b>	A component responds but the response is late.
<b>Environment Condition(s):</b>	Normal operation
<b>System Response:</b>	The fault is detected and logged.
<b>System Measures:</b>	Time to detect the fault.

Table 5.3: Quality Attribute scenario 01 - Availability

## 5.4. Constraints

In terms of project constraints, people involved in the process (e.g. clients, stakeholders, developers and others) can impose restrictions on the decisions that are made, as the project progresses. Constraints play a prominent role in every project development because they are design decisions with 0 (zero) degrees of freedom. Therefore, three constraint groups were defined (technical, business and legal). Each group of constraints is explained in the following subsections bellow.

### 5.4.1. Technical Constraints

Technical constraints are technical design decisions which have an impact in the design of the solution. They usually consist of technologies, programming languages, protocols, standards, among others. The project's identified technical constraints are described in the table below.

<b>ID:</b>	<b>TC01</b>
<b>Title:</b>	REST architectural style
<b>Description:</b>	System architecture must follow REST architectural style.
<b>Dependencies:</b>	None
<b>ID:</b>	<b>TC02</b>
<b>Title:</b>	Temporary Numbers Provider
<b>Description:</b>	Twilio, a CPaaS provider will be used to supply temporary phone numbers as well as the communication services associated with them (calls and SMS).
<b>Dependencies:</b>	None

Table 5.4: Project technical constraints

### 5.4.2. Business Constraints

Business constraints are decisions imposed by business considerations that can affect the achievement of the business goal. Thus, these must be satisfied with the architecture of the system. As this internship does not expect a real product, the business constraints presented below were stipulated by Paulo Sousa (Product Owner), Jorge Sousa (Scrum Master) and João Costa (Scrum Master).

<b>ID:</b>	<b>BC01</b>
<b>Title:</b>	Schedule
<b>Description:</b>	The MVP development and validation must be finished by June. This period might be extended until the 1st of September.
<b>Dependencies:</b>	None
<b>ID:</b>	<b>BC02</b>
<b>Title:</b>	Temporary Numbers per Device
<b>Description:</b>	It should be possible for users of the mobile application to buy/own more than one temporary phone number in the same device.
<b>Dependencies:</b>	None
<b>ID:</b>	<b>BC03</b>
<b>Title:</b>	Mobile Application Calls
<b>Description:</b>	Calls made via the Mobile App must be made natively.
<b>Dependencies:</b>	None

Table 5.5: Project business constraints

### 5.4.3. Legal Constraints

Legal constraints are events or circumstances under which either party's access to, provision of, or use of, the system results in a violation of any Law. Therefore, as a response to the legal compliance of IT systems, system developers must always consider the laws and legal issues in their system development.

Despite the solution defined in this document serving only as an MVP, it is necessary to notice that even for this purpose it was required to comply with legislation, namely with the provision and use of the Twilio's phone numbers. Therefore, for a real product scenario one must bear in mind the following aspects:

- **User's Information:** This is related to how the system will have to deal with all users' information. For legal purposes, if the system is designed only for Europe, it will have to comply with the Regulation (EU) 2016/679 named as "The General Data Protection Regulation" (GDPR).
- **Temporary Phone Numbers:** This is related to how the system will deal with the supply and use of temporary phone numbers by the users. In case the system uses an external entity as a supplier (as was used in this internship) then, as a rule, it will have to fulfill the requirements imposed by that entity. On the other hand, if the system designed has the mechanisms for creating and supplying temporary numbers without external dependencies, then a higher set of laws and obligations must be complied with.

After presenting the process of elicitation of requirements, the next step is to detail the system architecture developed throughout the internship.

## Chapter 6

# Solution's Design

Establishing a software architecture is one of the most critical tasks of any project because "there is no universal standard on good or bad" [76]. The quote "all models are wrong; some of them are useful" written by George Box, a British statistician expresses the real meaning of defining a software architecture. It is hard to achieve a simple, complete and precise architecture, but it is not impossible. A good architecture "implies taking into consideration all kinds of requirements (performance, security, etc.), the organization of the system, how the system parts communicate with each other, whether there are some external dependencies, what the guidelines and implementation technologies are, what risks to take into consideration, and much more"[77].

This chapter describes the architecture and the overall decisions to implement the WIT Temporary Numbers Platform through four sections. The first section explains the model that will be used to document the architecture of the solution. The second section exposes the solution architecture according to the model of C4. The third section details the data model for the solution. Lastly, the fourth section, the security mechanisms of the architecture

### 6.1. Model C4

In order to define the architecture of the internship solution, the author of this document opted for using the model created by Simon Brown, the C4 model. According to this model "a software system is made up of one or more containers, each of which contains one or more components, which in turn are implemented by one or more code elements" [78]. This is the reason why the model stands for "Context, Containers, Components and Code".

Considered as an "abstraction-first" approach to diagramming software architecture and based on a small set of abstractions, this model allows a more natural level of learning and use. Nevertheless, before starting to present the system architecture, it is necessary to contextualise some core concepts that are part of the C4 model to help the reader to better understand the diagrams presented sections bellow.



The C4 model considers the static structures of a software system in terms of containers, components and code. The people that use the systems are considered as persons. [78]

- **Person:** represents human users of a software system (e.g. actors, roles, personas, among others).
- **Container:** represents an application or a data store. It is something that needs to be running in order for the overall software system to work (e.g Server-side web application, Client-side web application, Mobile app, Database, Shell script).
- **Component:** is a grouping of related functionalities encapsulated behind a well-defined interface (e.g. a collection of implementation Java classes).
- **Code:** one or more code elements (e.g. classes, interfaces, objects, functions).

With the aforementioned static structures, it is possible to create a collection of Context, Container, Component and (optionally) Code diagrams. Each one of the diagrams has a specific level of detail as well as the objective. The diagrams could be: [78]

- **Level 1 - System Context diagram:** is a diagram to start diagramming and documenting a software system. Drawing a Context diagram will allow seeing the big picture of the desired system. The detail is not essential on this level, meaning that the main focus is on identifying the people (actors, roles, personas) and software systems. This type of diagram is mostly used to show to non-technical people.
- **Level 2 - Container diagram:** shows the high-level shape of the software architecture and how responsibilities are distributed across it. A zoom-in into the system boundary is done, where the main goal is to identify the containers, the technology choices and how the containers communicate with one another. It is a simple, high-level technology focused diagram that is helpful for software developers.
- **Level 3 - Component diagram :** exposes how a container is made up of a number of "components", what each of those components are, their responsibilities and the technology/implementation details. Overall, this type of diagram is more technical because it decomposes each container in order to uncover the major structural blocks and their interactions.
- **Level 4 - Code:** This is an optional level of detail and is often available on-demand from tooling such as IDEs. This level of detail is not recommended for anything but the most important or complex components.

In order to draw these diagrams, there is no specific notation to use. Colours and shapes can be added to supplement the diagrams, either to add additional information or to make the diagram more aesthetically pleasing. However, the most common and generic way is presented in figure 6.1.

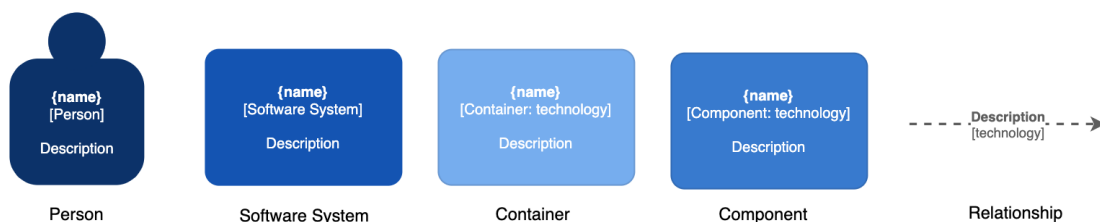


Figure 6.1: Notation used with C4 model

## 6.2. Architecture

In this section, the solution architecture will be detailed through the model described in the previous section. In order to present everything in a structured way, this section was subdivided into three subsections, each representing a level of detail of the C4 model. In each one of these subsections are explained the reasons that led to those decisions.

### 6.2.1. Context View

The first level of the C4 model requires that users and entities should be identified first, in order to create the contextual diagram. Therefore, the users that will interact with the system are the following:

- **Registered Users:** users that are registered on the platform and can use the services provided.
- **Unregistered Users:** users that are not registered on the platform but can access information about the services provided.
- **Administrators:** admins that are responsible for managing the platform.

After the users had been identified, it was necessary to determine the entities/services which the system will interact with. With that purpose, a meeting took place between the intern and his supervisors, resulting in the suggestion of the integration of a CPaaS provider (Twilio). To ensure this provider would own all the needed services for the system, the intern implemented a simple standalone Java application using some of the services provided by the Twilio Platform (Verify API, Phone Numbers API, Calls API and SMS API). This proof of concept helped conclude that Twilio had all the necessary services to be the service provider of the system.

In essence, the Twilio chosen services that will interact with the software system are:

- **Twilio Verify API:** allows to create OTP code sending services and corresponding validation.
- **Twilio Phone Numbers API:** provides the temporary phone numbers.
- **Twilio Voice API:** allows to make and receive calls through the temporary phone numbers.
- **Twilio SMS API:** allows to send and receive messages through the temporary phone numbers.

After the requirements for the first level had been met, the contextual diagram could be drawn. Figure 6.2 shows the contextual diagram.

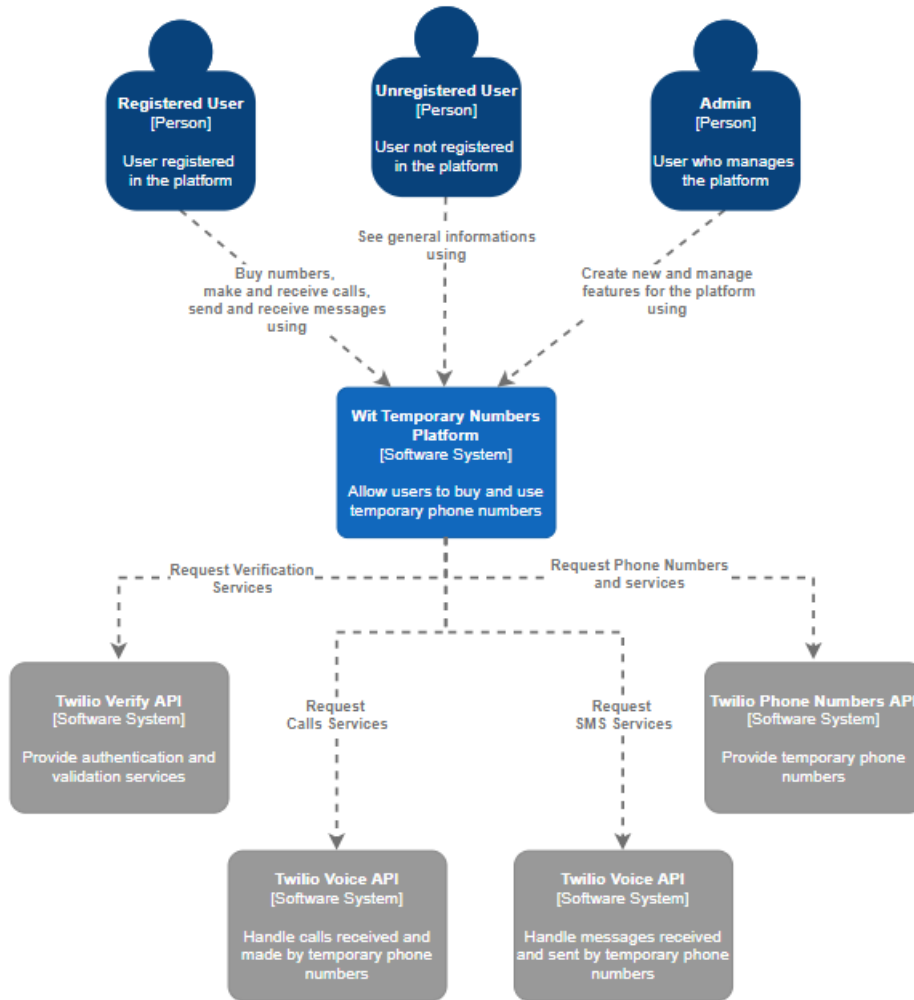


Figure 6.2: Contextual Diagram of WIT Temporary Numbers Platform

### 6.2.2. Containers View

Once the system context was defined, it was necessary to identify the containers, the technologies and the communication protocols. With that in mind, the containers identified were:

Container	Description
API Backend	RESTful Web Server that is responsible for performing all the business operations and exposes the services through endpoints.
Mobile APP	Client application that allows the users to buy and use temporary phone numbers.
Admin Dashboard	Client application that allows to manage the platform and gather statistics. Both operations are made through API Backend requests.
Database	Container that will store all the information related with the WIT Temporary Numbers Platform (users, calls, messages, phone numbers).

Table 6.1: Platform Containers

Followed by the identification of the containers, it was necessary to choose which of the technologies would be used to implement each one of the containers. The table below presents the technologies chosen as well as a small description.

Technology	Container	Description
<b>Spring Boot</b>	API Backend	Spring Boot is an open-source Java-based framework developed by Pivotal Team, used to build stand-alone and production-ready spring applications. [79] Through Spring Boot it is possible to reduce development, unit test and integration test time and increase productivity.
<b>React Native</b>	Mobile APP	React Native is an open-source JavaScript framework created by Facebook for writing real, natively rendering mobile applications for iOS and Android. [80] Built on top of React, the Facebook JavaScript library, React Native differs mainly in the DOM manipulation. Nowadays, it is a trendy framework that has broad community support.
<b>React</b>	Admin Dashboard	React is an open-source, component-based JavaScript library used to build user interfaces. [80] It was created by Facebook in 2012 and is one of the most popular Javascript libraries used to build apps front-end right now.
<b>PostgreSQL</b>	Database	PostgreSQL is a free and open-source relational database system created by a computer science professor Michael Stone and his team. [81] Supporting both SQL for relational and JSON for non-relational queries and backed by an experienced community of developers, PostgreSQL is one the best databases available on the market.

Table 6.2: Platform Containers Technologies

The choice of protocols had to be made carefully since the information will be exchanged between containers. Thus, it was necessary to find protocols that are "universal" in the sense of being compatible with the major technologies used on the market. In addition, it is also necessary that each protocol used has security mechanisms in order to be unable to compromise the exchange of information. As a result, the protocols chosen are exposed in the table below.

Protocol	Description
<b>HTTPS</b>	Hypertext transfer protocol secure (HTTPS) is the secure version of HTTP, which is the primary protocol used to send data between a web browser and a website [82]. HTTPS is encrypted in order to increase security of data transfer.
<b>JSON</b>	JSON (JavaScript Object Notation) is a lightweight data-interchange format for storing and transporting data. [83]

Table 6.3: Communication Protocols

After exposing the containers, the technologies, and the communication protocols to be used on the implementation phase, the intern did the containers diagram of the WIT Temporary Numbers Platform. This diagram represents a higher level of the software system to be developed, allowing to observe which components integrate the system and their relations.

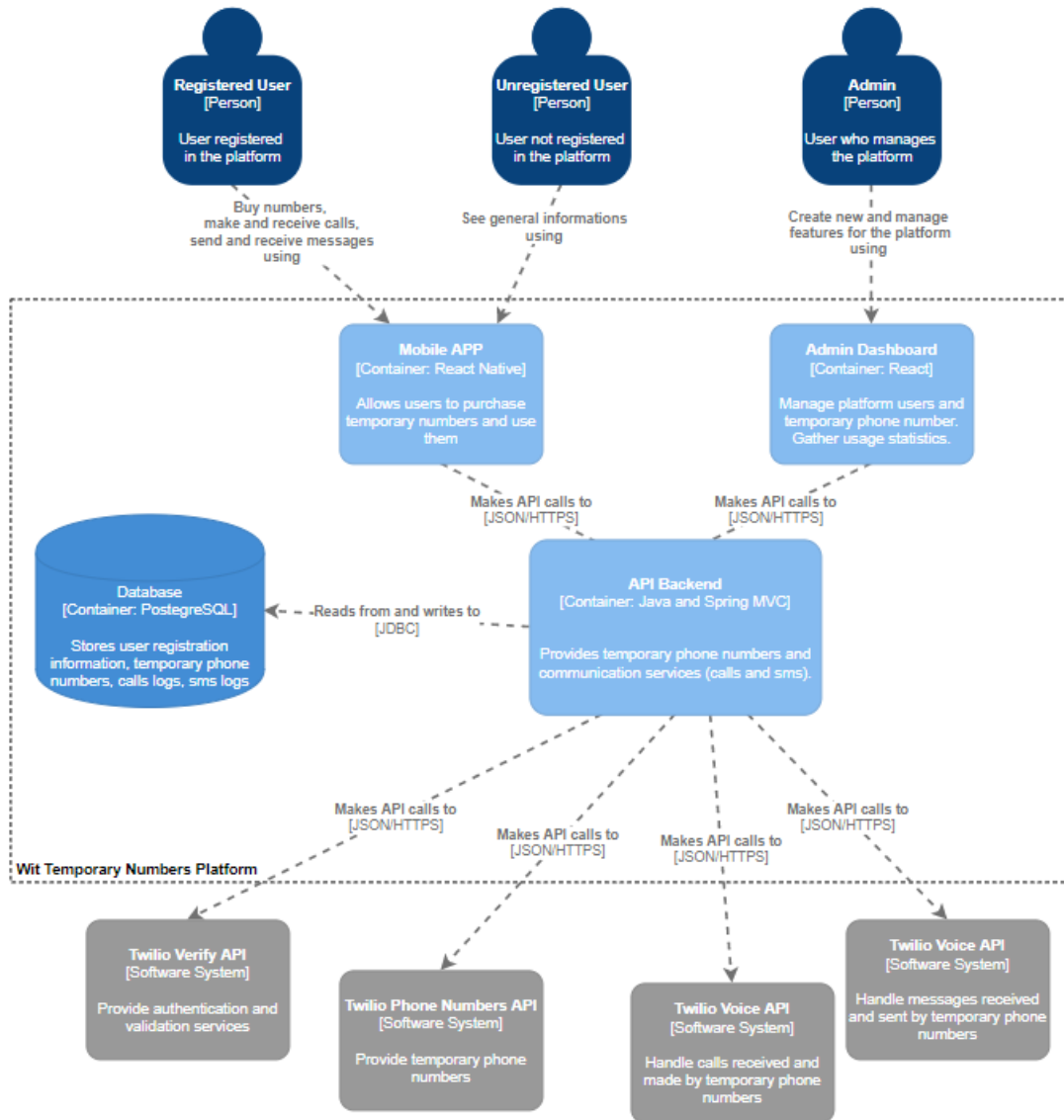


Figure 6.3: Containers Diagram of WIT Temporary Numbers Platform

### 6.2.3. Components View

After having defined, in the previous section, the containers that constitute the software system, it was necessary to zoom-in again, this time on each one of the containers. As the Mobile Application and Admin Dashboard will only consume the resources (via front-end) provided by the API Backend, the author did not think it necessary to zoom-in on these containers. On the other hand, the API Backend container is the core of the software system due to being there that providing and managing services will be performed. In this sense, it was necessary to understand which components will form the API Backend.

Essentially, it was necessary to define which components will form the API Backend. For that purpose, the intern analysed the Spring Boot architecture and subsequently identified the type of components that will constitute the API Backend container:

- **Controllers:** are classes that provide access to the application services by exposing REST endpoints.

- **Services:** are class files that are used to write business logic in a different layer, separated from the controllers class.
- **Repositories:** are classes that are written for encapsulating storage, retrieval, and search of data.

To streamline the component identification process, the author decided to represent the services that the platform will have to provide through modules. Thus, the modules identified are as follows:

- **Temporary Numbers Module:** is responsible for getting temporary numbers from the provider and for providing them to the client's application.
- **Messages Module:** is responsible for handling the messages sent and received by the users through the temporary phone numbers.
- **Calls Module:** is responsible for handling the calls made and received by the users through the temporary phone numbers.
- **Payment Module:** is responsible for handling the payments made by the user and some admin manipulations (e.g. update user balance).
- **Users Module:** is responsible for managing the users data stored in the platform.
- **Statistics Module:** is responsible for gathering systems statistics about the functioning of the system.

Although the payment module was mentioned, for the sake of simplicity, no payment mechanism will be implemented or integrated. Instead, each user account will have a specific amount in their balance to allow the purchase of temporary numbers and subscribing of plans. In case of a real product, a payment mechanism will have to be implemented, or third party API integration can be done such as PayPal, Stripes or Noodlio Pay.

After the modules were identified, the intern started making some sketches of the component diagram of the API Backend container and the final version is presented in figure 6.4. This diagram is more technical, allowing to uncover the major structural blocks that compose the container, the technologies and their interactions.

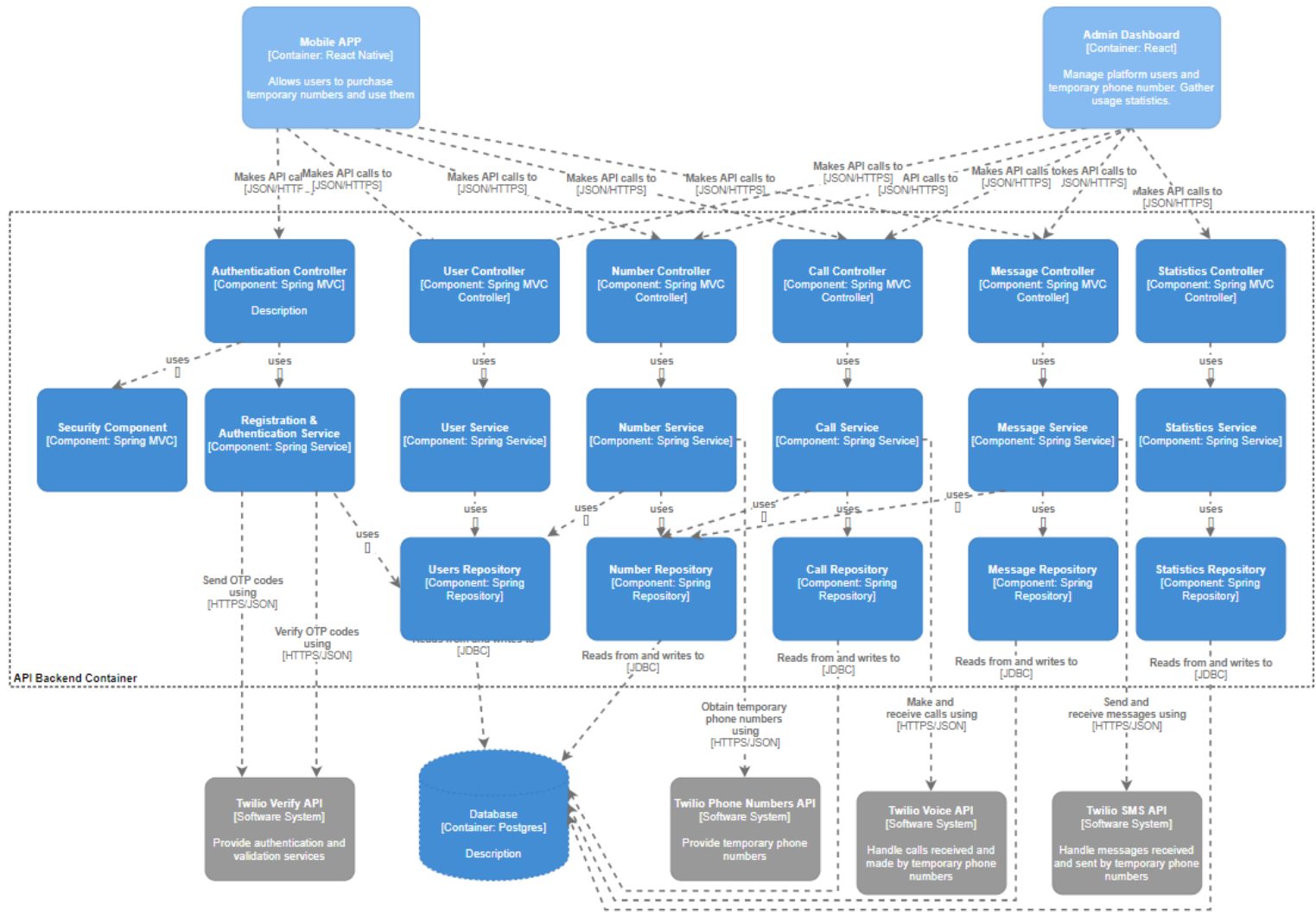


Figure 6.4: Components Diagram of API Backend container

### 6.3. Data Model

The database is one of the primary containers of the system's architecture because it is responsible for storing and securing data. Therefore, the WIT Temporary Numbers Platform will be composed of a data model that only the API Backend can manipulate or interact with. For this purpose was chosen a relational database (PostgreSQL) because of its quality in transactions management and the structure allows to link information from different tables through the use of foreign keys. In order to explain the data model defined, an entities table was created as well the entity-relationship diagram.

Entity:	Description:
Pre-Registrations	Stores all the pre-registrations made through the mobile application by the users.
Users	Stores all information related to a user's account's.
Tokens	Stores the user's refresh and access tokens. Both tokens are used to ensure a user has access to the appropriate resources .
Phone Numbers	Stores all information related to a temporary phone number purchased by a user.
Calls	Stores all records of calls made and received by the temporary numbers owned by the users of the platform.
Messages	Stores all records of messages sent and received by the temporary numbers owned by the users of the platform.
Plans	Saves the plans that can be added to the phone numbers purchased by the users.

Table 6.4: Data model entities

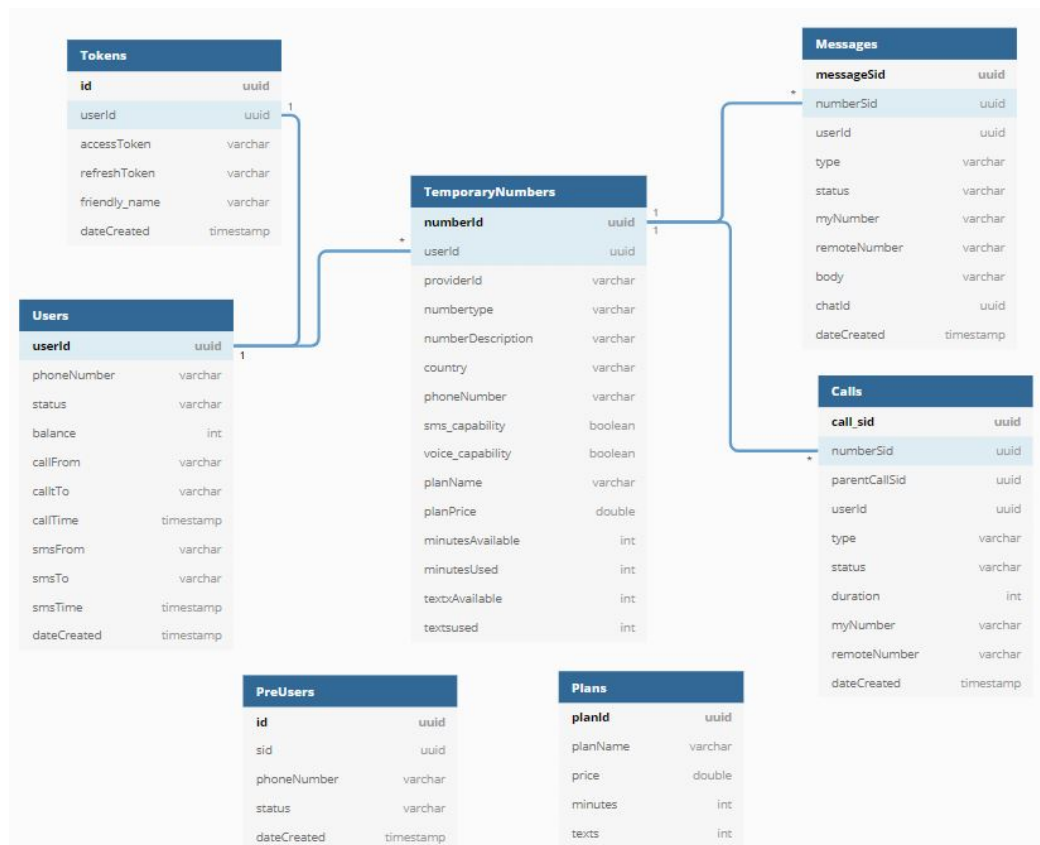


Figure 6.5: Entity Relationship Diagram



## 6.4. Security

Security Architecture is developed to provide guidance during the design of the system. Thus, Security Architecture is the design of artifacts that describe how the security mechanisms are positioned and how they relate to the overall systems architecture. These mechanisms serve the purpose to maintain the system's quality attributes such as confidentiality, integrity and availability.

On this project, the most crucial security aspects are the user authentication, as well as the confidentiality, integrity and authenticity of messages exchanged between the server and mobile clients. The following subsections explain how these aspects (Authentication, Authenticity and the Non Repudiation ) will be handled.

### 6.4.1. Authentication

The API Backend will manage the registration and sign-in of the users into the platform. Both processes are similar in a way that:

1. An OTP code must be sent to the user.
2. The user must insert the OTP code.
3. The API must validate the code inserted by the user.

The OTP code mentioned before is "a code that is valid for only one login session or transaction using a mobile phone" [84]. To implement this mechanism, the API will integrate the Twilio Verify API [85]. This API allows the creation of a user verification service that sends and verifies OTP codes.

Concluding, a user service verification will be created through the Twilio Verify API to manage the registration and sign-in of the users.

### 6.4.2. Authenticity and Non Repudiation

Users and businesses use APIs to use or connect to services. For security purposes, each API must have an authentication and authorization mechanism to protect the data exchanged among all parties involved. In order to protect the WIT API Backend, the intern will use JSON Web Token (JWT) technology. The JWT is "an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object"[86]. With JWT, it is possible to control API authentication and the exchange of information between parties, ensuring that the resources are made available only for the clients that have permission for it.

The process behind JWT is the following:[86]

1. The application or client requests authorization from the authorization server with their credential.

2. If the authorization is successful, the authorization server returns an access token to the application.
3. Whenever the application or client wants to access a protected resource from the API, it uses the access token.

Figure 6.6 resumes how a JWT is obtained and used to access APIs:



Figure 6.6: Diagram of JWT creation and usage [86]

After the architecture of the system to develop has been explained, some of the platform's services are going to be presented in more detail.

This page is intentionally left blank.

## Chapter 7

# The WIT Temporary Numbers Platform

In this chapter, the author describes in detail some of the core services of the WIT Temporary Numbers Platform. With this approach, the author intends to explain more technically the processes and mechanisms behind each service provided by the platform. Consequently, this chapter was divided into four sections. Section 7.1 presents the processes related to users accounts. Section 7.2 describes how temporary phone numbers will be obtained and managed. Next, section 7.3, details which processes will be used in order to gather platform statistics. Lastly, section 7.4 explains the processes related with the platform's management.

### 7.1. Accounts

Each customer of the platform must have an account to access WIT Temporary Numbers Platform's resources (temporary phone numbers). Therefore, an account needs to be created when the user first signs into the platform.

All the resources purchased by the platform's users are associated to a Twilio Master Account that was created to access them. Through this Master Account it is possible to access all the resources requested by the Temporary Phone Numbers Platform. Since the Mobile Application is an internal component of the platform, there is no need to distinguish between used resources. However, for a final product, there may be external client applications that intend to use the services supplied by the platform, in which case will make sense to distinguish between resources acquired by the internal mobile application component and by the external client applications.

With this in mind, it was verified that Twilio allows for that distinction, through the creation of subaccounts. In this case, each subaccount will represent a client application.

Although each subaccount has its own characteristics, the master account is the one that manages everything.

In order to illustrate this process, a diagram has been drawn, for further clarification of this approach. Figure 7.1 presents said diagram.

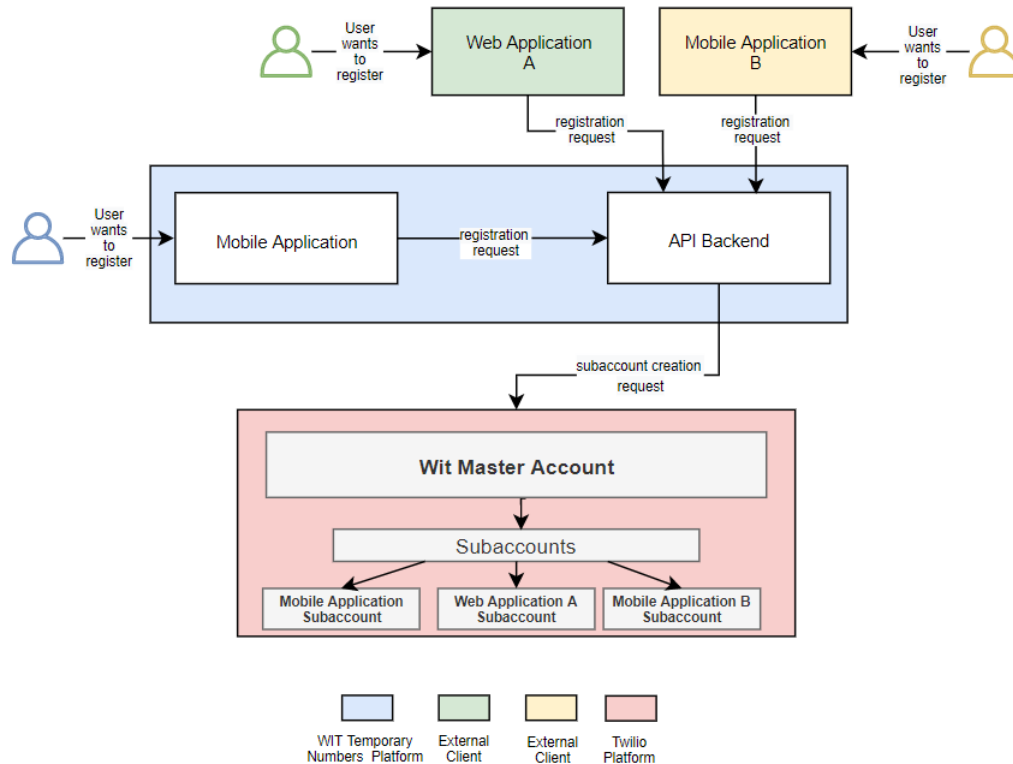


Figure 7.1: Diagram of account managing process

As can be seen from the previous diagram, this approach allows to achieve the segmentation of users by the client application. Also, it makes the processes of managing the resources used and gathering statistics more efficient. It is worth mentioning that using this approach means that we are dependent on Twilio for client application segmentation on resources and services usage. This means that, if in the future, a provider change needs to take place, then a new segmentation mechanism needs to be implemented or integrated.

### 7.1.1. Account Creation

The registration of new users on the platform is done through two processes. First, it is necessary to send an OTP code to the user via SMS. Second, the system must validate the inserted code.

Sending and verifying of OTP codes will be done through the Twilio Verify API. To this end, a service will be created on the Twilio platform, which will allow the sending and verifying the OTP codes. The authentication also shares this process.

The service created through the Twilio Verify API allows to define some characteristics such as the length (4 to 10 digits ) and the way of sending the OTP (SMS, voice call or email). The server of the platform checks the phone number provided by the user, and Twilio also performs a process in order to see if that number is well-formatted and can receive SMS's before a verification code is sent.

In order to illustrate the flow of the account creation process, the sequence diagram is presented in figure 7.2.

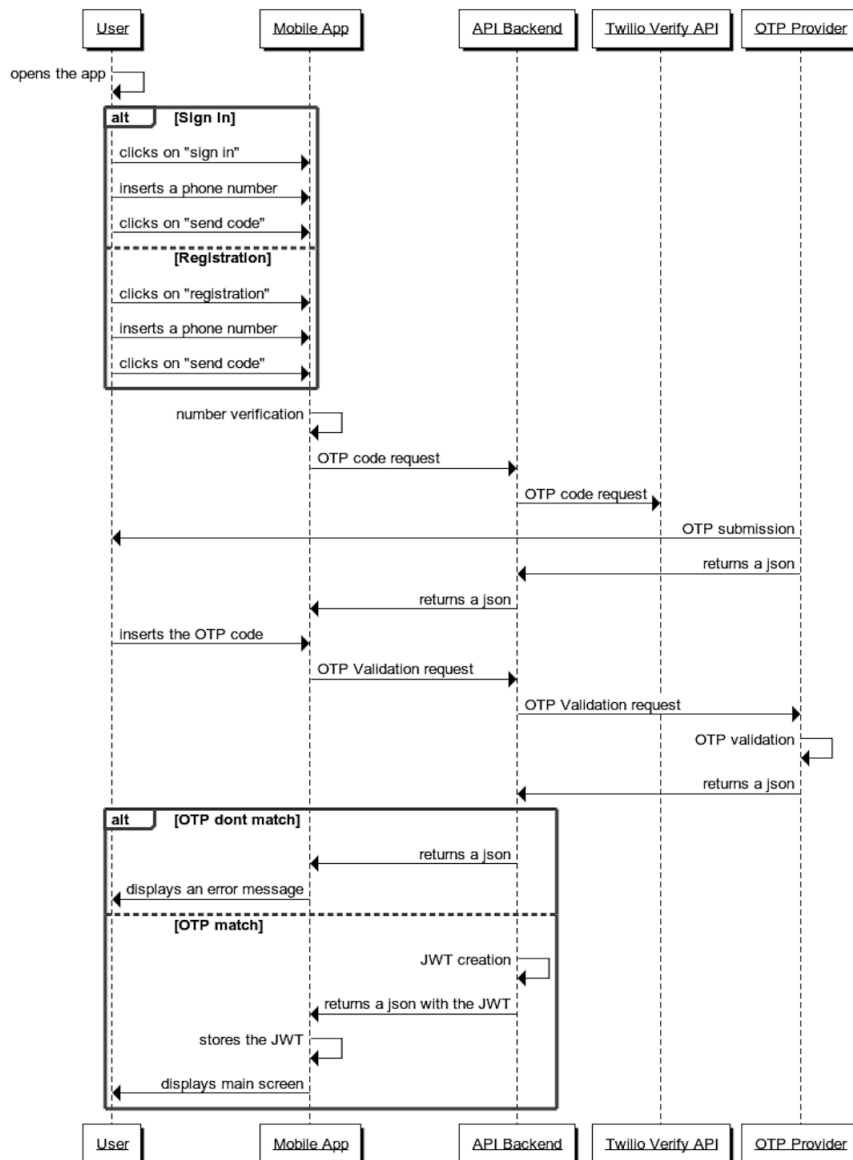


Figure 7.2: Diagram of account creation process

### 7.1.2. Account Cancellation

In addition to creating an account, a user is free to cancel or delete their account at any moment. In order to explain the processes, a sequence diagram (figure 7.3) was drawn.

The diagram exposed bellow displays two flows of account cancellation. The two flows are almost identical, contrasting only in a case of whether a user has phone numbers linked to their account to make a request to Twilio in order to cancel those phone numbers, or not. This cancellation is mandatory because those phone numbers need to be recycled before being made available once more. The process of recycling is handled by Twilio, meaning that the Wit Temporary Numbers Platform only needs to request the cancellation of the phone numbers.

Additionally to the cancellation of the numbers, it is also necessary to delete the user's information. On the one hand there is a legal responsibility of erasing all the data related to a user. On the other hand, there is an interest in maintaining the usage records (e.g.

calls, SMS) for matters related to statistics and user preferences. Therefore, this issue must be analyzed and discussed for the commercial version of the product.

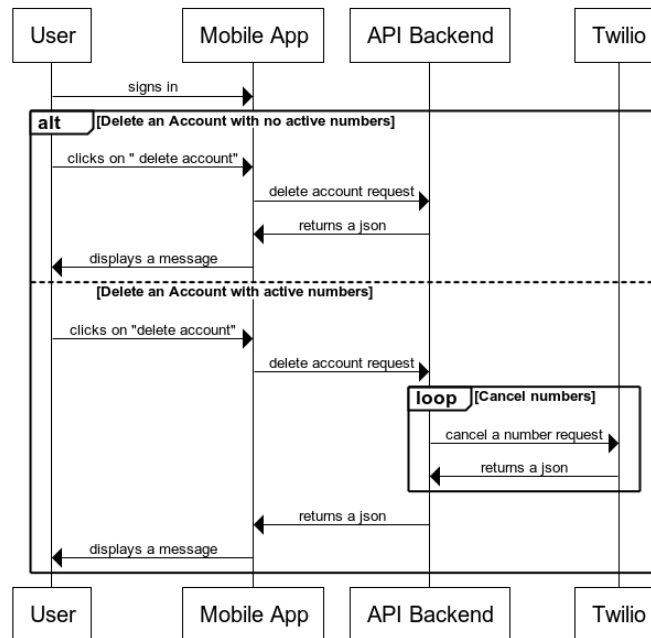


Figure 7.3: Diagram of account cancellation process

## 7.2. Temporary Phone Numbers

The temporary phone numbers provided by the WIT Temporary Numbers Platform will be supplied via Twilio Phone Numbers API. In order to be possible to obtain those phone numbers, it is necessary to create Regulatory Bundles on Twilio due to new changes made in February of 2020 related to country regulations. By creating Regulatory bundles allows Twilio to provide adequate identity documentation to the local regulators or carriers avoiding that the temporary phone numbers communications be rejected or disconnected.

In order to create the regulatory bundle, the first step is to indicate if the phone number will be used for personal or business purposes. Depending on the phone number purpose, user documentation is then required and needs to be validated by Twilio. Only after the documentation has been validated and consequently, the bundle has been created, it is possible to obtain phone numbers and use them.

The changes made by the Twilio platform mean that each user of the WIT Temporary Numbers platform must provide documentation (e.g. Government-issued id, residence permit, utility bill) in order associate the phone numbers provided with themselves. This is a critical aspect that will have to be analyzed for a real product because the end users may not want to provide this information and if they do not, WIT will have to bear responsibility, giving this information in order to acquire the temporary phone numbers.

### 7.2.1. Purchasing

The process of purchasing a temporary number affects three entities (the mobile application, the API Backend, and Twilio Numbers API) of the system.

The purchasing process is initiated by the mobile application, where the user, through the interface, chooses a temporary phone number type according to their preferences (e.g. mobile phone or local number). During the purchasing process, two distinct moments on the server-side happen. The first is the arrival of request from the mobile application asking to provide some phone numbers according to the user preferences. The second is when the user presses the button to obtain the phone number. In both moments, requests to Twilio Phone Numbers API are made.

From the moment that the search request is made to the server until the moment that the request is made to Twilio to acquire the number, two key problems can happen:

- **No funds:** the user is informed that he does not have enough funds to buy that phone number.
- **Phone number has already been acquired:** the user is informed that the purchase order has failed and will have to repeat the purchase phone number process.

Figure 7.4 shows the sequence diagram of purchasing a phone number.

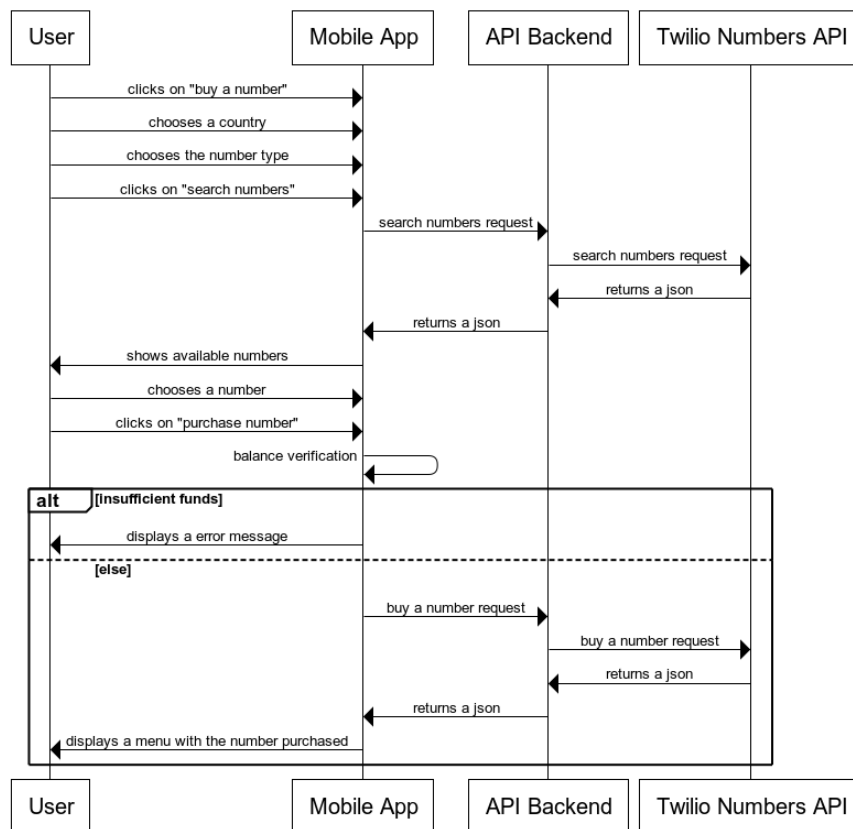


Figure 7.4: Sequence diagram of purchasing a number



### 7.2.2. Canceling

A number's cancellation process requires that a request is made to Twilio to cancel the intended number. This way, the number can be unbound from the user to which it was associated and advance to the recycling process. Note that Twilio is in charge of the whole recycling process for the canceled number.

Figure 7.5 represents the sequence of the cancellation process for a phone number.

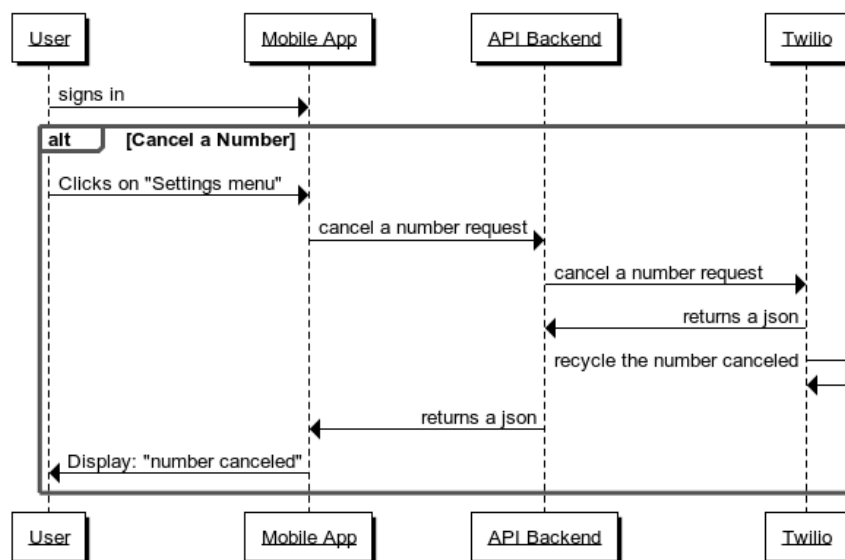


Figure 7.5: Sequence diagram of canceling a number

### 7.2.3. Calling

Making a call is one of the most important features of the platform. In order to achieve this feature, the intern used Twilio's Voice API. This API allows to use the phone numbers acquired with the Twilio Phone Numbers API to make and receive calls.

As mentioned in chapter two, there are several methods available to provide communications through temporary numbers. For this project, the forwarding method is used, meaning that the temporary phone number acts as a middle agent, redirecting all communications (incoming and outgoing) from and to the user's respective phone numbers. Through this method, data from the user's communications package is used for outgoing communications. It is worth mentioning that although this method has been chosen, Twilio provides the possibility to do it through VOIP. In this case, whenever a user wants to make an outgoing call, Twilio initiates a call to the user himself and only when the user answers, is the call redirected to the person whom he seeks. This method only requires a stable internet connection.

The sequence diagram presented on the figure 7.6 explains the process of making a call using a temporary phone number.

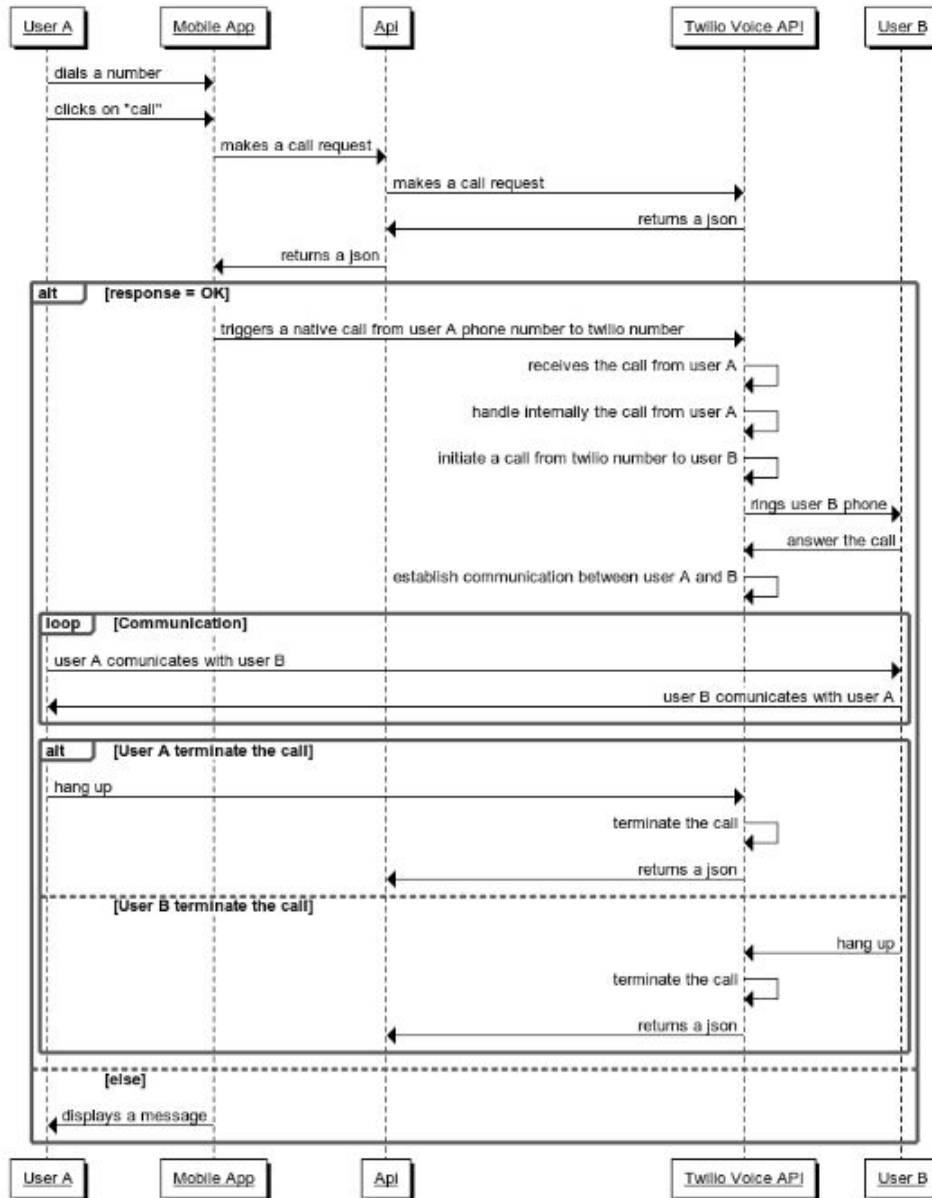


Figure 7.6: Sequence diagram of calls process

### 7.2.4. Messaging

In addition to calls, the sending of messages is another core feature of the platform. Thus, to achieve the sending and receiving of messages through the temporary phone number, the Twilio SMS API is used. The process of sending and receiving messages is fully managed by the API, ensuring that data from the user’s communications package is not consumed.

The process of sending and receiving messages is shown in figure 7.7 through a sequence diagram.

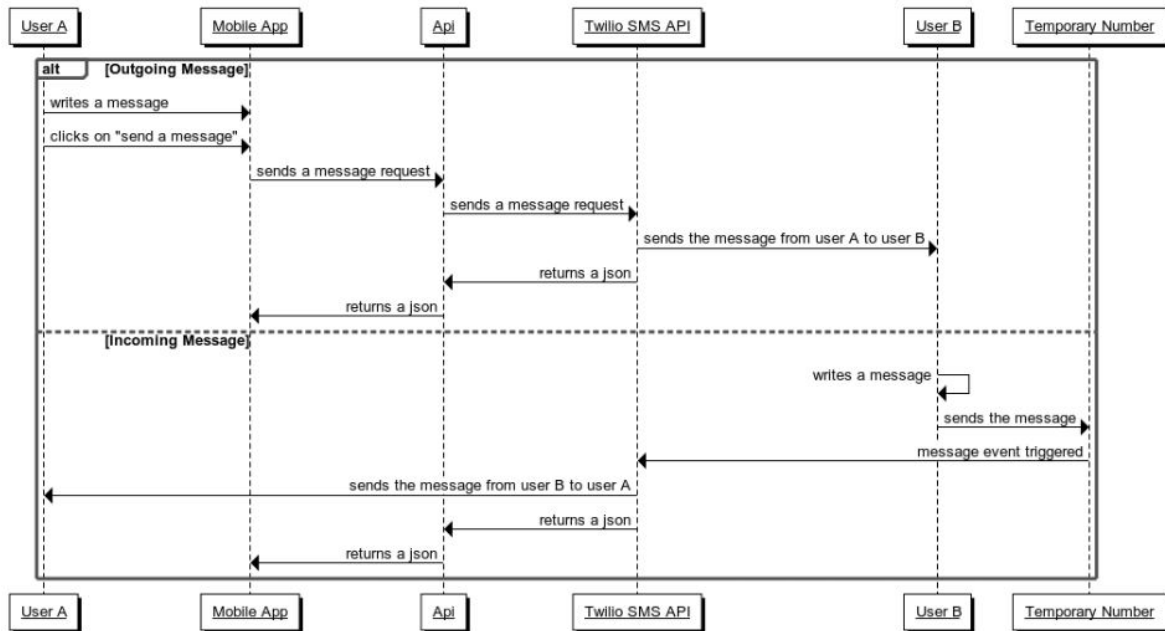


Figure 7.7: Sequence diagram of sending and receiving a message

### 7.3. Statistics

In the development of any software system, it is important to conceive mechanisms that allow to obtain statistics. These statistics are then used to study the system's performance, understand the user's preferences, system metrics and even find possible purchase patterns. Some functionalities were defined for this purpose, regarding the gathering and treatment of the platform's data for statistical purposes.

These are the types of statistics defined:

- **System:** Used to obtain system functioning statistics (e.g. number of successful requests)
- **User:** Used to obtain user statistics related to the platform's usage (e.g average number of calls per month)
- **Numbers:** Used to understand the statistics regarding the supplied phone numbers (e.g. most purchased type of number)
- **Calls:** Used to obtain statistics concerning the calls made and received by the platform's phone numbers (e.g. average number of calls made per day by the platform)
- **Messages:** Used to obtain statistics regarding the messages sent and received by the platform's phone numbers (e.g. average number of messages sent per day by the platform)
- **Revenue:** Used to understand the system's revenue over time.
- **Costs:** Used to understand the cost of the operations carried out by external entities.

In order to gather these statistics, the Admin Dashboard (Web portal) will be used, which will provide a simple interface, and the API Backend will perform the logic operations.

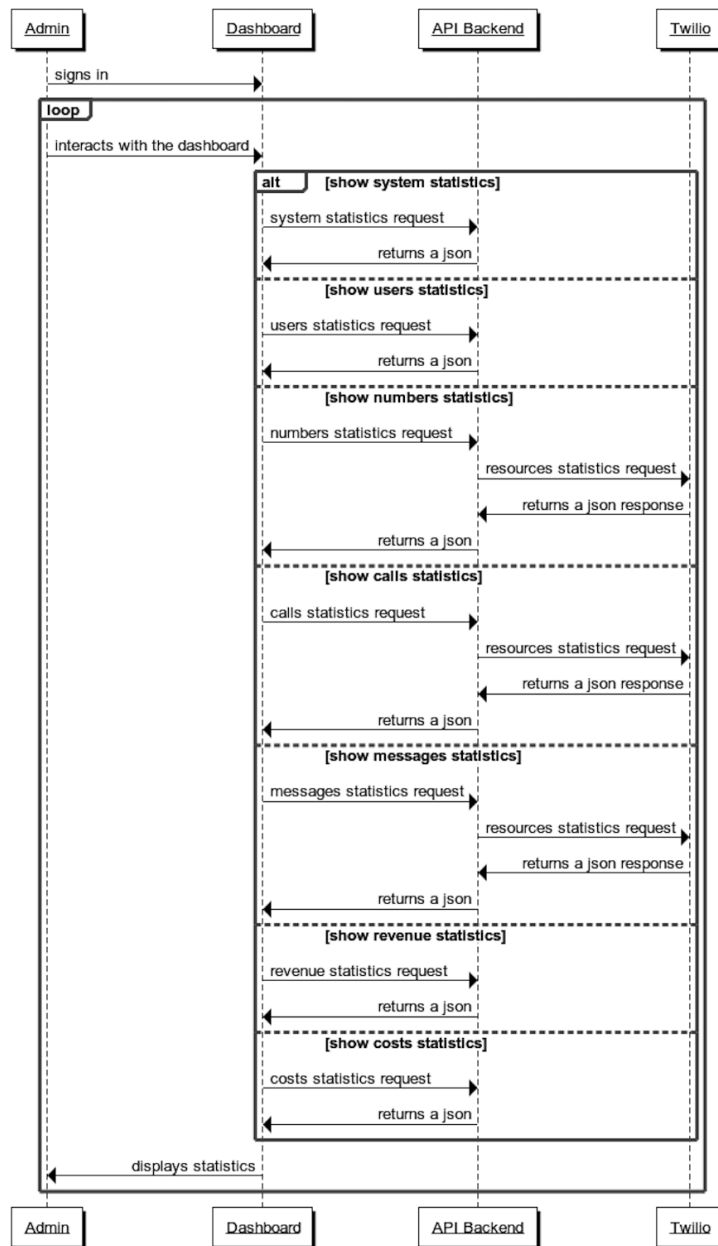


Figure 7.8: Sequence diagram of the statistics features

## 7.4. Management

One base aspect of any platform is that it possesses management mechanisms. In this case, this task will be carried out by the system’s administrators through the Admin Dashboard.

Thus, some basic functionalities were designed by the intern so that the system’s administrators can manage the platform:

- Notify or Alert a Client.
- Add/Remove funds to/from user balance.
- Suspend a user account.
- Close a user account.

The figure 7.9 presents the defined features.

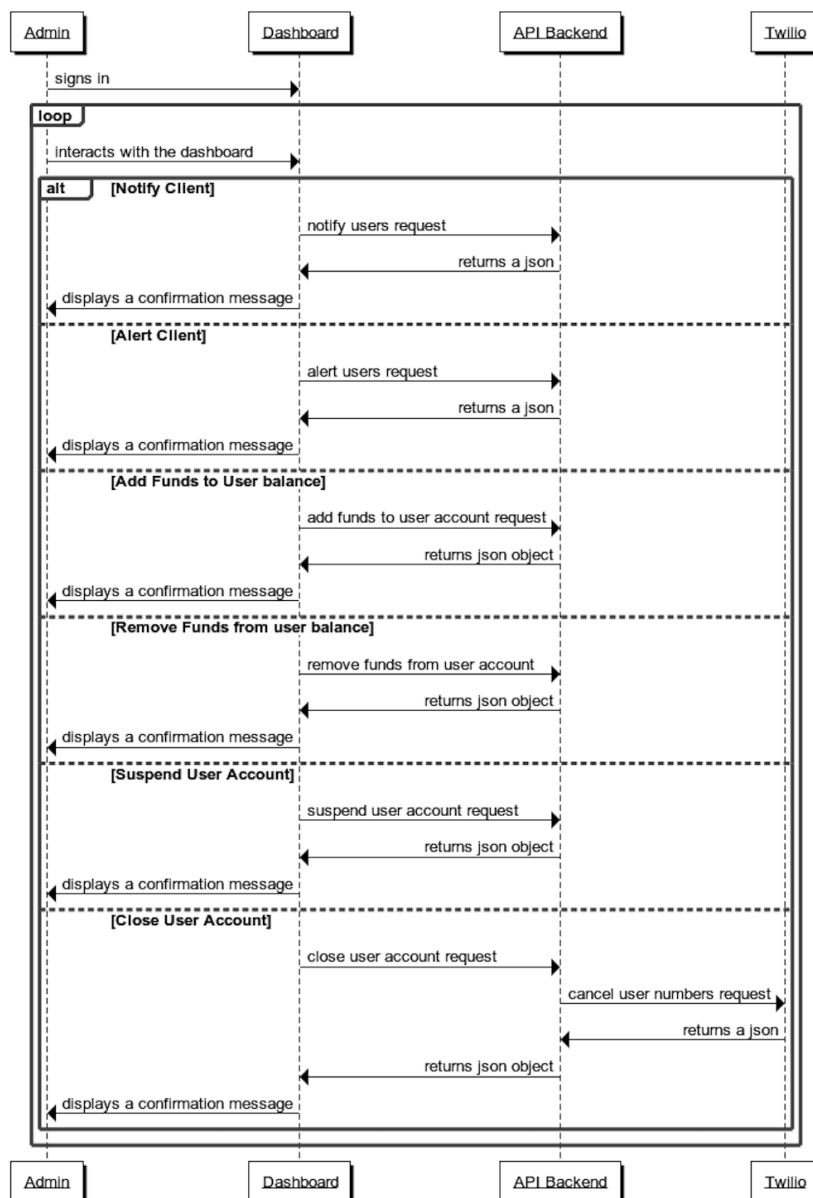


Figure 7.9: Sequence diagram of management features

In this chapter were described in detail some of the core services of WIT's Temporary Phone Numbers Platform. Next is presented the process of implementing the designed solution.

# Chapter 8

## Implementation

This chapter is dedicated to explaining the implementation phase of the project developed during the internship.

Two main components were implemented: API Backend and Mobile Application. Section 8.1 presents the server side which consists of the API Backend. Next, section 8.2 addresses the client side which encloses the Mobile Application component.

### 8.1. API Backend

The API Backend encompasses the whole environment related to the server part of the platform. It is this component that conducts all the business operations of the platform and exposes them to the client applications. Therefore, this section intends to show the processes, procedures and decisions that were made during the implementation of the server.

#### 8.1.1. Structure

In order to start developing the server side it was necessary to understand which layers make up the Spring Boot architecture. The figure 8.1 represents the 4 layers that make up Spring Boot. The figure demonstrates that each layer of the architecture is directly communicating with the adjacent layers, due to the workflow. It means each layer is interdependent with the adjacent layers, so if one layer is changed, the adjacent layers must be changed as well.

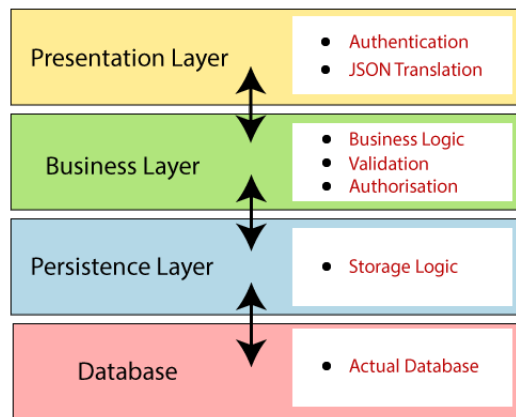


Figure 8.1: Spring Boot structure [1]

A brief description of the layers is given below [1, 87]:

1. **Presentation Layer:** handles the HTTP requests.
2. **Business Layer:** comprises of the service classes and uses services provided by data access layers. Handles all the business logic.
3. **Persistence Layer:** holds all the storage logic and translates business objects from and to the database.
4. **Database Layer:** performs CRUD operations (create, retrieve, update, delete).

After this brief introduction about the architecture used by Spring Boot, it is necessary to mention that in the presentation layer of the API Backend developed no front end component was implemented. Instead, an API interface is exposed, containing all the services provided by the platform. This means that the server only provides the resources and it is the client side that must parse, treat and display the information to the users.

The figure 8.2 represents the Spring Boot flow architecture.

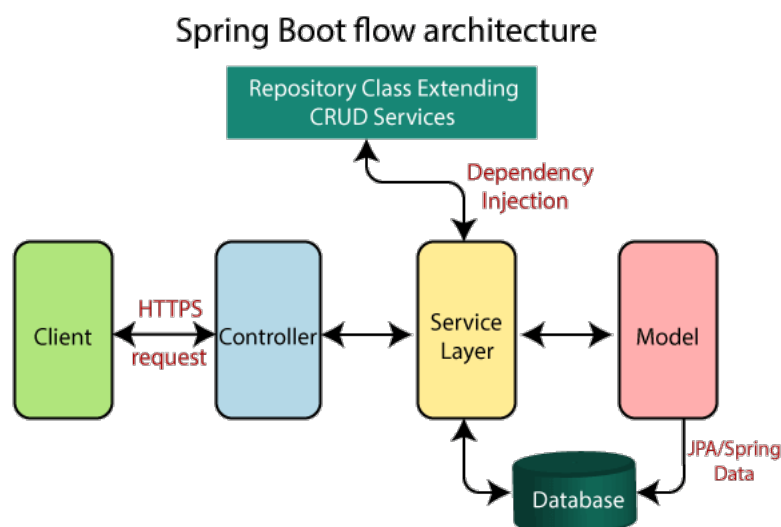


Figure 8.2: Spring Boot flow architecture [1]

The HTTP request or web requests are handled by the Controllers in the presentation layer, the business logic is controlled by the services, and persistence (storage logic) is handled by the repositories. A controller can handle multiple services, a service can handle multiple repositories, and a repository can handle multiple databases.

Since the Spring Boot architecture is based on the Spring framework it uses mostly the features and modules of Spring there is no need for the Data Access Object (DAO) classes. In order to clarify the flow of the API, this is an exemplification:

1. The mobile application client makes an HTTP request (GET or POST) to the server.
2. The request is forwarded to the controller, and it maps the request and processes it. It also calls the service logic if needed.
3. The business logic is performed in the service layer, and the logic is performed on the data from the database that is mapped with the entity class through JPA.
4. A JSON object is returned as a response to the client.

In accordance to what was previously stated, the intern implemented the controllers and services stated in chapter 6 figure 6.4, as well as:

- **JWT Controller and Service:** Responsible for exposing the service used to access the resources (access token).
- **Plan Controller and Service:** Responsible for exposing the services related to the plans that can be acquired by the users when purchasing temporary phone numbers.

Regarding the structure exposed above, the dependencies used to initiate the implementation were:

Dependencies	Description
<b>spring-boot-starter-web</b>	Used for building RESTful applications, using Spring MVC. Tomcat is the default embedded container.
<b>spring-boot-devtools</b>	Used for fast application restarts, LiveReload, and configurations for enhanced development experience.
<b>spring-boot-stater-jpa</b>	Used for persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
<b>spring-boot-starter-security</b>	Used to customize authentication and access-control in Spring applications.
<b>org.postgresql</b>	JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.
<b>io.jsonwebtoken</b>	Used to create and verify JSON Web Tokens (JWT) with java.
<b>com.twilio.sdk</b>	Used to interact with the Twilio API's.
<b>org.flyway</b>	Used for version control for database.

Table 8.1: Dependencies used on the API Backend implementation

The dependencies mentioned above are automatically managed by Spring Boot. It manages itself and the dependencies are upgraded automatically when a new version is available.



### 8.1.2. Architectural Constraints

For an application to be considered as RESTful, there are standards that must be followed and constraints are put on the system. It was decided that the project would follow a REST architectural pattern therefore, these standards and constraints also apply to the WIT Temporary Numbers Platform, namely to the API Backend component.

The constraints that are a part of the used architectural style and how they were applied to the implementation of the API Backend (RESTful Services), are as follows [88, 89]:

- **Client Server:** This constraint requires a separation between the client and the server. Through this separation it is possible to improve the clients' mobility across platforms and improve scalability. In this project, this separation was put in place and the server will be in charge of all the business operations whereas the client will be in charge of providing the frontend (user Interface and User Experience).
- **Stateless:** This constraint requires that the communication between the server and the client be done stateless. A stateless communication implies that all of the information is kept on the client's side, and the server will only receive the information pertaining to each sent request. This constraint was applied to the project and user-server communication follows this condition, meaning that session state is kept entirely on the clients' side and the server only needs to verify if the requests have the information regarding the request itself.
- **Cache:** This constraint requires that responses from the server be marked as cacheable or non-cacheable. This was not implemented, which can mean loss in performance, but its importance is acknowledged.
- **Uniform Interface:** This constraint means that the author must define the API Interface for the resources inside the system which are exposed to the clients through endpoints, thus uniformizing, as the name implies, the way of interacting with the server, regardless of device or type of application (Mobile App or website). This constraint encloses four guidelines, of which the author followed three:
  - **Resource-Based:** Requests identify individual resources (e.g WitApi/accounts).
  - **Manipulation of Resources Through Representations:** Enough information is included in every message to describe how to process it, to facilitate the analysis of the request by the server.
  - **Self-descriptive Messages:** resources are represented for the client and with them, enough information to alter or eliminate said resource on the server, as long as there is permission. Example: With the creation of an account, an ID is associated with the account and with that ID it is possible to buy phone numbers, delete the account, among other actions.
- **Layered System :** This constraint allows the use of a layered system architecture, where each layer can be composed by a server, a proxy or a load balancer but the client does not know to which one they are connected and which one answered the request. For the project, this constraint was not verified, meaning that the clients interact directly with the platform's main server.
- **Code-on-Demand:** This constraint is optional, and allows that executable code be provided to the client. For this project, this constriction was not needed nor followed.

All these constraints make up a “truly” RESTful API, but the violation of one or more constraints still allows for the RESTful denomination, it is simply not a “truly” RESTful API. Following this thought, the web service implemented by the author is simply defined as RESTful.

### 8.1.3. Web Security

In terms of web security, it is known that web applications are vulnerable to security threats since they are exposed to the open world of the internet. Therefore, the access to certain web pages, files or data must be restricted to authorized personnel only. For this purpose the intern used the Spring Security, a spring framework "that focuses on providing both authentication and authorization to Java applications" [90].

In the context of this project, this framework was used for authorization purposes in order to ensure that requests contain the necessary permissions to be satisfied. This mechanism was achieved through the use of Spring filters.

A Spring filter is "an object used to intercept the HTTP requests and responses" [91] meaning that it is possible to perform two operations at two instances: before sending the request to the controller and before sending the response to the client.

Resuming, the intern implemented a filter that allows to define which URI's need to be protected and the ones that are supposed to be available for external entities. The process of authorization carried out, is based on checking if the requests follow the Bearer authentication scheme where a security token is passed (JSON WebToken).

### 8.1.4. Exception Handling

One of the key procedures that can be implemented in any software product is the exception handling. This is "to handle the runtime errors so that normal flow of the application can be maintained"[92].

Since the API Backend will deal with all the requests performed by the clients it is imperative to have a mechanism that allows not only the identification of failures without compromising the system but also to facilitate the process of sending the responses derived from the exceptions.

This is an important aspect seeing that as the API grows, the degree of consistence needed within the app grows too and, to avoid the process of creating successive Response Entities for the responses, an exception handler can be implemented.

For that reason and aware of this problem the author decided to implement an exception handler in order to centralize the error handling logic. This was an easy task, since Spring Boot provides different options to do exception handling in apps.

### 8.1.5. Data Management System

The data management process consists in several tasks, and managing a database involves designing, implementing and supporting stored data, to maximize its value. Out of several types of database management systems, the one used was the centralized method, that assures that the data resides in one system and place, and all clients access data through that one place.

After defining the type of database management, from the options available on the market to develop the database, the intern used PostgreSQL (free and open-source relational database management system)[93] as previously established and through Docker, reducing the task of installing and running software to two simple commands: `docker [94] run` and `docker pull`. For starters, the intern did a pull on the most recent and stable image of PostgreSQL and from then on began developing the database model.

To make the database management process more fluid, a GUI (Graphical User Interface) application was used, DBeaver. This GUI is a free “multi-platform database tool for developers, database administrators, analysts and all people who need to work with databases”[95]. To make the connection between Docker and DBeaver it was only necessary to provide some parameters of connection such as host, port, user, password and driver. To integrate with the server it was only needed the addition of the Postgres maven dependency.

The intern was advised to pay particular attention to was the database migrations process. In order to do that, the intern used Flyway, an “Apache v2 licensed open-source tool that makes database migrations easy”[96]. Using migrations, Flyway updates a database to the next version, and said migrations are written using SQL database-specific syntax. The integration of this tool with the server was done through the addition of the maven dependency.

This approach revealed useful, seeing as the intern managed to build the database through constant versioning.

### 8.1.6. Integration with Twilio

The platform to be developed, from the beginning of its conception, required the integration with external entities in order to be able to fulfill the functional requirements defined for the platform. Thus, this project included the integration of four APIs provided by Twilio (a CPaaS), namely the Twilio Verify API, Twilio Phone Numbers, Twilio Voice API and, at last, the Twilio Messages API.

In order to be able to use the resources provided by Twilio’s web service APIs, the user created an account on the respective platform to which was associated an Account SID and AUTH TOKEN (represented in figure 8.3 ). Although in Twilio’s ecosystem, each product has its own API, the way of interacting with each one of them is the same, meaning that they can be used through HTTP or using Twilio’s helper libraries.

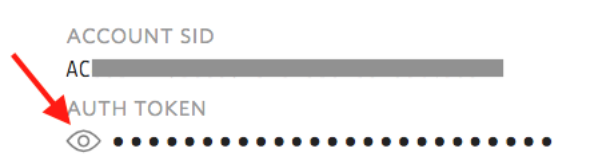


Figure 8.3: Twilio credentials generated after the Twilio account creation

With that in mind, the process of authentication carried out by Twilio to access the resources involves the HTTP basic Authentication where the SID Account works as username and the AUTH TOKEN as the password. This allows to protect the URLs on the server side, meaning that only the server and Twilio can access them. An example of a request using the account credentials is as shown below:

```
curl -G https://api.twilio.com/2010-04-01/Accounts \
-u '[YOUR ACCOUNT SID]:[YOUR AUTH TOKEN]'
```

Figure 8.4: Example of a HTTP request

In addition to the type of authentication used to access the resources, it is also necessary to explain that Twilio uses Webhooks to asynchronously notify when some events happen, like receiving an incoming call or receiving an SMS message.

This way, when an event such as the arrival of a message occurs, Twilio makes an HTTP request (usually a POST or GET) to the API Backend at the URL configured for that Webhook. This request includes details about the event that was triggered. As the server receives the request, it must reply to Twilio with the instructions of the actions to perform.

Considering that the intern must use Webhooks to interact with Twilio, in the case of calls, it was necessary to define two Webhooks: one for when an incoming call arrives and another for when there is a state change in a call (e.g. call disconnection, reject call). In the case of messages, one Bundle was created for when a message is received.

A critical aspect is how the temporary phone numbers are supplied to the platform by Twilio, because since February 2020, Twilio mandates to provide documentation regarding the person that will use those numbers. For this purpose, the intern had to create two Regulatory Bundles, one for each number type in a country. Figure 8.5 presents the two Bundles created through Twilio console.





REGULATORY BUNDLE & SID	COUNTRY	TYPE	END-USER	STATUS
<a href="#">Bundle Mobile Numbers PT</a> BU43050898e7654500f0fa61b0757c636a	 Portugal	Mobile	Individual	 Twilio Approved
<a href="#">Bundle National Numbers PT</a> BU71e40f39d9d27fcc79e54946646a6378	 Portugal	National	Individual	 Twilio Approved

Figure 8.5: Twilio credentials generated after the Twilio account creation

Each created Bundle has an acknowledgment section (Figure 8.6) associated to it, that informs the user about how the documents they provided are used by Twilio.

Acknowledgement ^

---

I declare that the information provided is accurate. I acknowledge that Twilio will be processing the information provided for the purpose of identity verification, and will be sharing this information with my local telecommunications providers or authorities where required by local law. I understand that Twilio phone numbers may be taken out of service due to inaccurate or false information.


 We are required to collect this information to fulfill local regulatory obligations regarding identity verification for telephone numbers. We are committed to safeguarding any data we collect, and we will keep your personal data safe and confidential. Please see [Twilio's Privacy Statement](#) for details about how we protect and secure your personal data.

Figure 8.6: Acknowledgement message showed in every Regulatory Bundle

After the Bundles were created for mobile and national Portuguese numbers, the intern was apt to buy temporary phone numbers supplied by Twilio. From that moment forward, the intern started implementing the functionalities relating to communications that will be made by those phone numbers, the calls and messages.

## 8.2. Mobile App

The Mobile Application is the second most important component of the system. In relation to this module, two implementations were made. The first implementation used Expo (an open-source platform for making universal native apps for Android, iOS, and the web with JavaScript and React), due to being a widely used platform for the development of small projects. However, since Expo did not support native calls, the project had to be exported to a native version.

This way, this section will present the structure, the approach for navigation between screens, the obtained interfaces, and the data management and permissions.

### 8.2.1. Structure

This subsection presents the details of the client-side implementation related to the project, developed using the React Native framework.

The Mobile Applications structure is composed by four main folders:

- **assets:** this stores static files (e.g. images used in the application).
- **android:** this stores the native code (written in java) needed to run the app in an android platform.
- **node\_modules:** this stores NPM or Yarn installed packages.
- **src:** this stores the dynamic files.
- **ios:** this stores the native code (written in Swift).

From these guidelines onwards, the intern had total freedom to structure the code. With that in mind, the intern created, inside the src folder, the following directories:

- **components:** where the components created by the intern that will be used by the layouts are (e.g. custom buttons, flat lists, cards).
- **screens:** where all the application layouts, which are also divided into different folders depending on the related requirements are (e.g. profile, messages, calls, dialpad).
- **stacks:** where the stack Navigators that provide transitions between screens are (e.g. authentication, home, messages).

Even though React is multiplatform, the difference between ios and android functionalities made the intern focus solely on the android operating system.

Seeing as React Native is already a framework that excels for reusing components, and as it possesses a very active community, the intern made use of some of the community's libraries. Although libraries were used, there was always a need to explore and adapt them to the project. The libraries the intern used throughout the project are identified in the following table.

Library	Description
<b>react-native-axios</b>	A promise based HTTP client for the browser.
<b>react-native-responsive-screen</b>	A component that makes UI elements fully responsive.
<b>react-native-contacts</b>	A component that allows to access and manage the device's contacts.
<b>react-native-immediate-phone-call</b>	A component that allows to initiate an immediate phone call without further user interaction.
<b>react-native-gifted-chat</b>	A component that provides chat UI features.
<b>react-native-responsive-fontsize</b>	A component that allows for fonts to be fully responsive.
<b>react-native-raw-bottom-sheet</b>	A component that allows to create and custom bottom sheets.
<b>react-native-snap-carousel</b>	A swipe component that provides previews and multiple layouts for handling animations.
<b>react-native-progress-wheel</b>	A component used to create natively animated circular progress wheel.

Table 8.2: Platform Components

### 8.2.2. Navigation

Currently, there are several types of navigation that can be used in the development of a React Native application. Each one of these types has different objectives and implementation processes.

The intern had to examine the mockups made by the designers from WIT in order to understand which types of navigation would be used. The chosen navigation types were: Stack Navigator and Bottom Navigator, as shown in figure 8.7

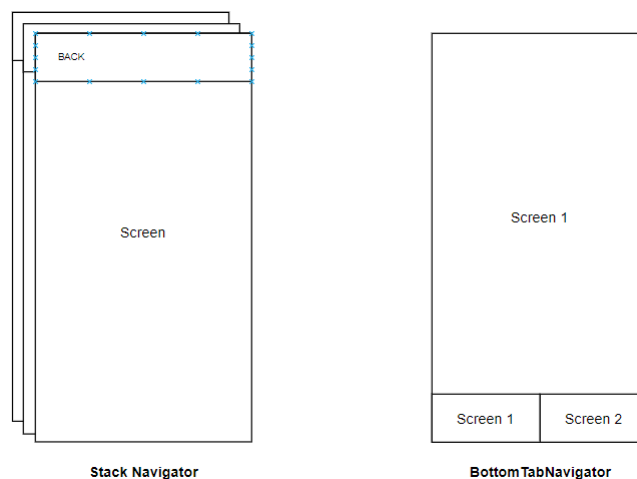


Figure 8.7: Types of navigators chosen

The project required that both navigators be used together. This falls into the Nesting Navigators category, meaning that one navigator is rendered inside the screen of another navigator. This concept is demonstrated in the following diagram (figure 8.8).

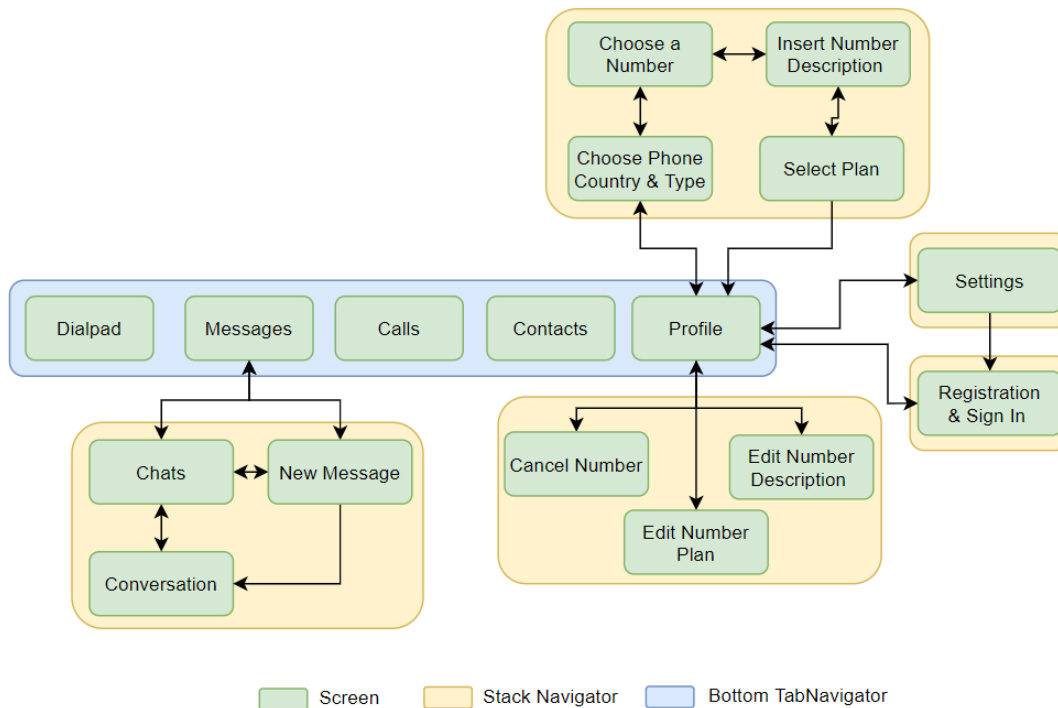


Figure 8.8: Navigation diagram

The implementation process for this concept proved hard, since this type of navigation affects the application's behaviour. Each navigator has its own navigation history, each one has its own options, parameters, nested navigators do not receive parent's events, and all of these are traits that interfere with the implementation process.

### 8.2.3. Data Management and Permissions

Concerning the data, the application will only store the access Token and the refresh Token that are used to make requests to the server (API Backend).

Regarding the permission, it is necessary that the user provides authorization for the application to access the contacts and make calls. Only then can they completely use the functionalities provided by the Mobile Application.

### 8.2.4. Interface

In this subsection are presented the interfaces that were developed for the Mobile Application component. Note that the final design is a result of the mockups (see Appendix C) done by the intern that were then sent for Wit's design team to decide on the final aspect. Then the intern followed and tried to match the design team's decision. Spiik was the named decided by the design team for the Mobile App.

The biggest challenges found in this section were regarding the handling of data and navigation management between screens.

In all of the following figures, phone numbers will be censored for privacy purposes, since the numbers can be recycled and reassigned to other people in the future.

## Authentication

The process of authentication requires first of all that a valid phone number is inserted (figure 8.9), for a valid OTP code to be sent to the user via SMS. After this code is received by the user, they must insert it for validation (figure 8.10). If the user does not receive an OTP code, they may request the resending of it through the "Resend Code" button.

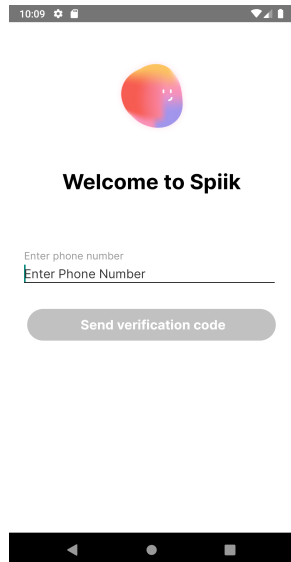


Figure 8.9: Phone number insertion screen

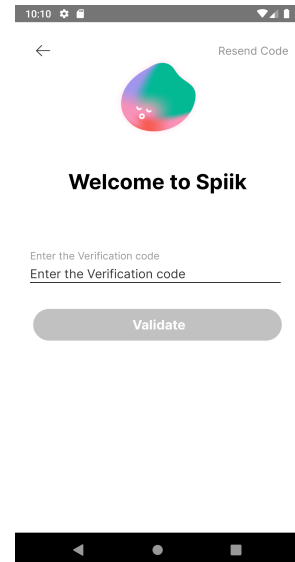


Figure 8.10: Code verification screen

The OTP code is sent by the service created in Twilio, using the API Verify. The user receives a message like the following:

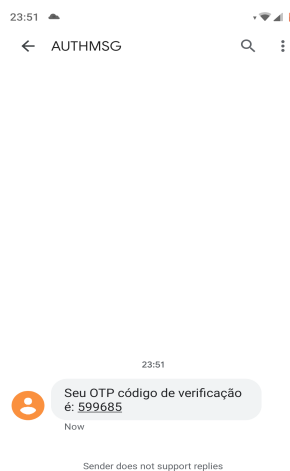


Figure 8.11: Message received containing the OTP code

## Profile

After the user manages to finish the authentication process, the app stores the access token and the refresh token, in order to be able to access the resources supplied by the API Backend component (through access token) and to refresh the access token (through refresh token).

The users are automatically redirected to the profile screen after the OTP code has been



validated. In this screen they are shown their profile, containing information regarding the temporary numbers they possess, as well as some of the characteristics of these phone numbers, such as the number of calls made from that phone number or the number of messages sent.

The following pictures show both the display for when the user does not yet have any purchased numbers (figure 8.12) or display the number(s) they have purchased (figure 8.13)

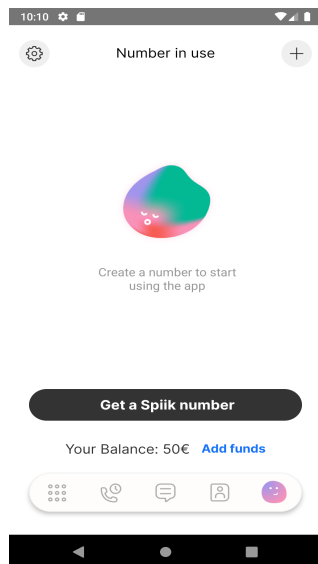


Figure 8.12: Account without purchased temporary phone numbers screen

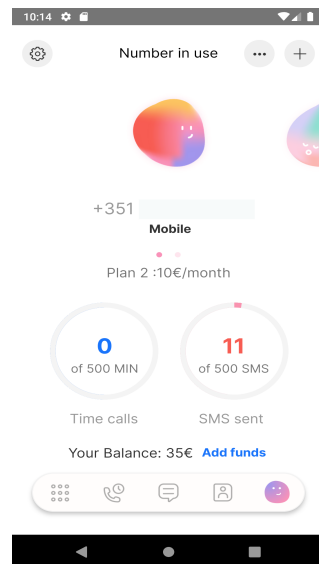


Figure 8.13: Account with purchased temporary phone numbers screen

In case the user possesses any temporary phone number, they may change the plan or the description associated with that or any other number they purchased. They can also at any time erase any purchased temporary phone number. The figure 8.14 shows the bottom sheet where all these options appear.

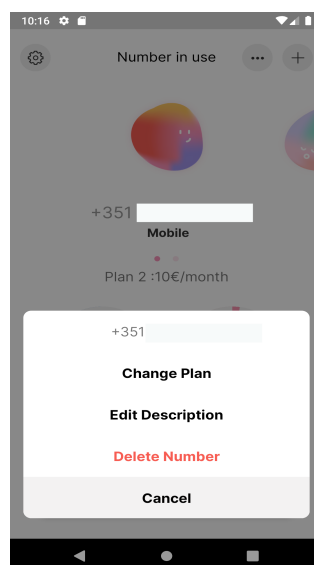


Figure 8.14: Phone number settings screen

The next two images will depict the description change screen (figure 8.15) and the subscribed plan change screen(figure 8.16).

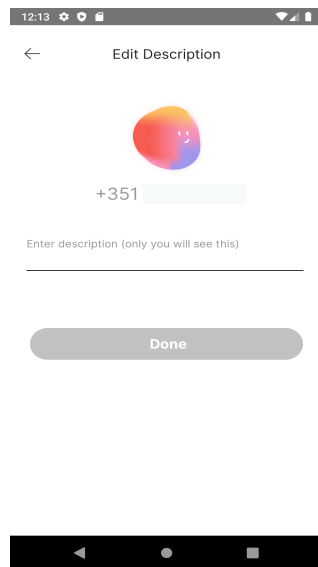


Figure 8.15: Edit phone number description screen

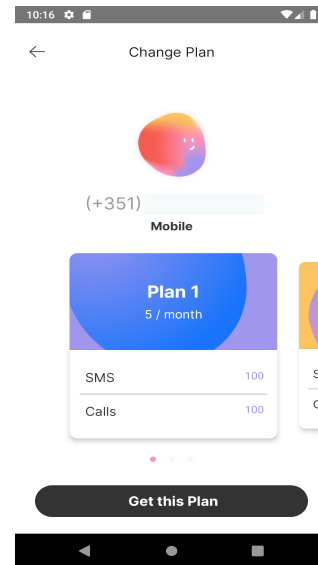


Figure 8.16: Change phone number plan screen

In case of cancellation of a temporary phone number, a new bottom sheet appears, where the user has to confirm the choice.

### Temporary Phone Number Purchase

The purchasing process is made up of four steps. First, the user has to choose the country of origin of the telephone number they intend to buy and the type of phone number (figure 8.17). Next, a determined amount of numbers is presented (figure 8.18), from which they can choose. If they do not like any of the options presented, they can request other options.

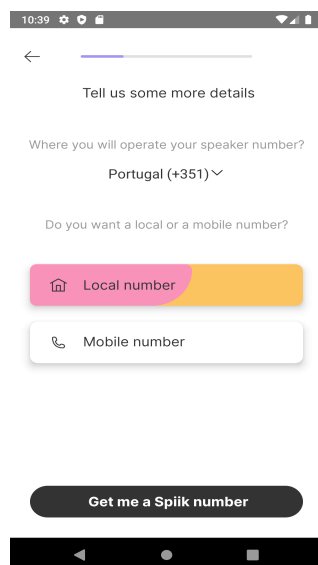


Figure 8.17: Selecting phone number characteristics screen

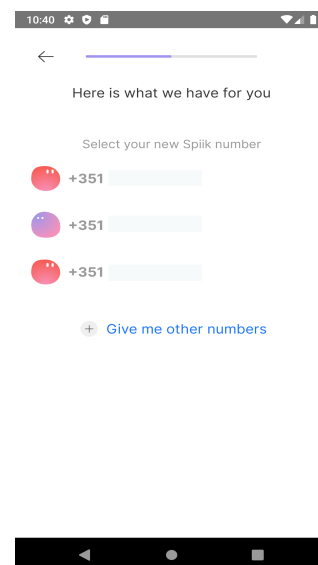


Figure 8.18: Example of temporary phone numbers screen

Thirdly, they can add a description to the number they intend to acquire (figure 8.19). If they do not want to add a description, they can skip this step.

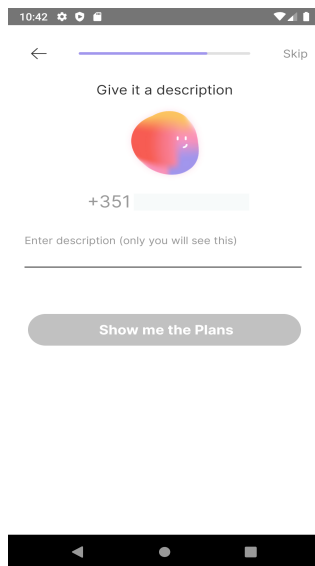


Figure 8.19: Naming temporary phone number screen

Lastly, they must select which plan they would like to associate with the phone number they intend to acquire (figure 8.20). Each plan has a set number of minutes and messages available. It is also shown to the user which plans they can get, according to their balance.

After the user selects the plan, they are automatically redirected to their profile, where the newly acquired phone number will already also appear (figure 8.21).

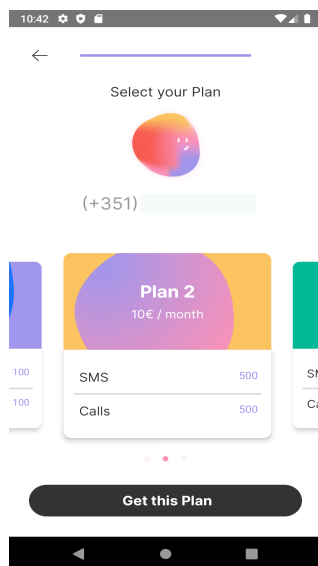


Figure 8.20: Choosing a plan screen

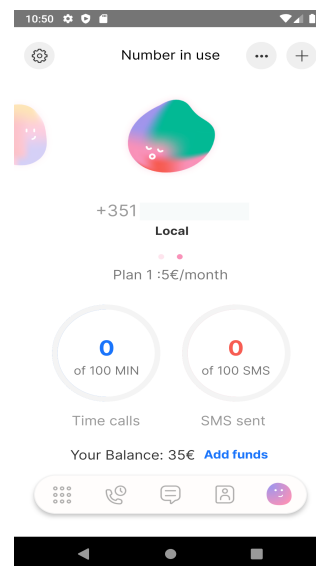


Figure 8.21: User profile screen

## Dialpad

The dialpad (figure 8.22) is a screen in which the user can type in a phone number they intend to contact. Since the users can have more than one temporary phone number, they can choose which one to use for which communications (figure 8.23).

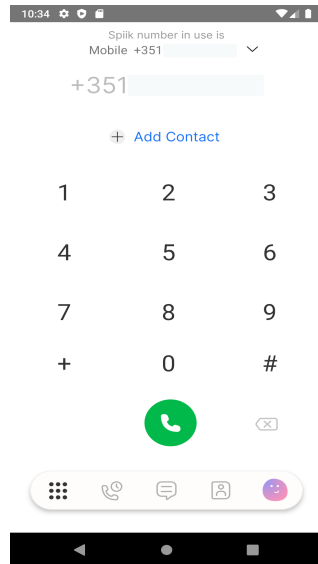


Figure 8.22: Dialpad screen

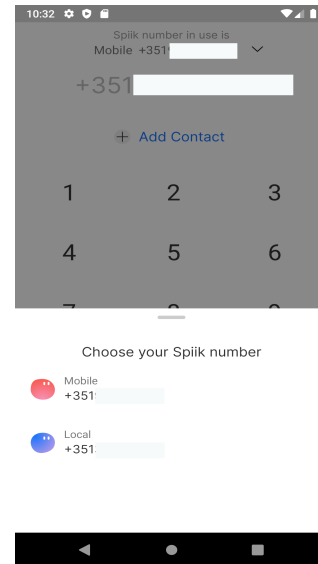


Figure 8.23: User profile screen

As soon as the user presses the call button, a native call is initiated from the user's phone number to the temporary phone number selected at the top of the screen (figure 8.24).

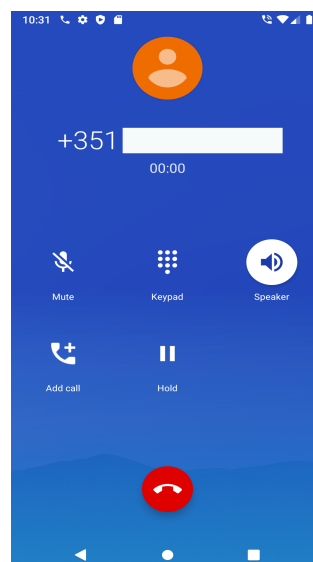


Figure 8.24: Call in progress screen

## Contacts

The user can access their contacts list in the app. For a user to be able to access their contacts, they have to give authorization for the application to access their contacts (figure 8.25). After this authorization is granted, the contacts are automatically imported and displayed (figure 8.26).

In this screen, the user can search for the contact they wish to communicate with and use the call or message buttons to immediately initiate communications.

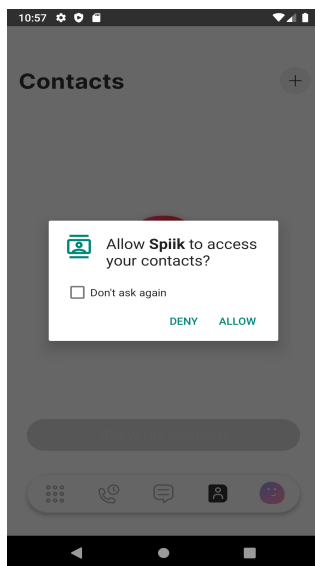


Figure 8.25: Allow access to contacts message screen

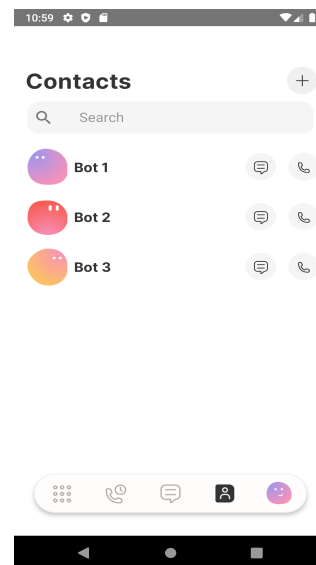


Figure 8.26: Imported contacts screen

## Messages

Other than the calls, the user can also send and receive messages through the temporary phone numbers. For this purpose, when the user accesses the messages tab, they will come across a screen where their chats are displayed (figure 8.27).

In the context of this project, a chat is a conversation room of sorts between the user and the other person, where all exchanged messages are shown.

In the first screen is displayed the information of who the conversations are with, as well as the date of the last message. If there are no conversations, the screen will display a message encouraging the sending of messages (figure 8.28).

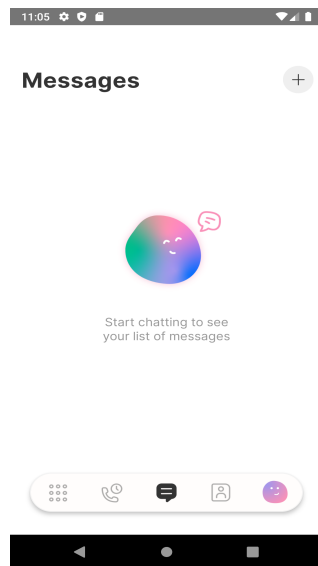


Figure 8.27: Message history with no conversations screen

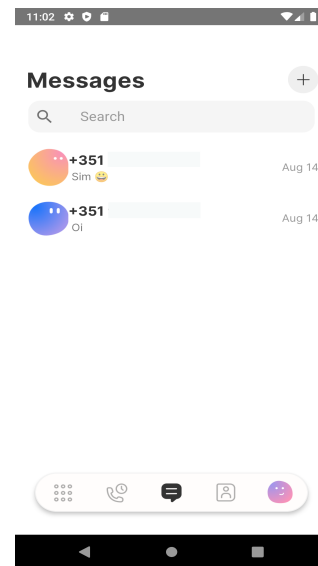


Figure 8.28: Message history with conversations screen

Other than accessing the messages history, the user can press the button on the top right corner to create a new message and be redirected to a menu where their contacts appear and from which they can choose to who they want to send the message (figure 8.29). After selecting the message's recipient, the user is redirected to a screen where they can write the message they want to send (figure 8.30).

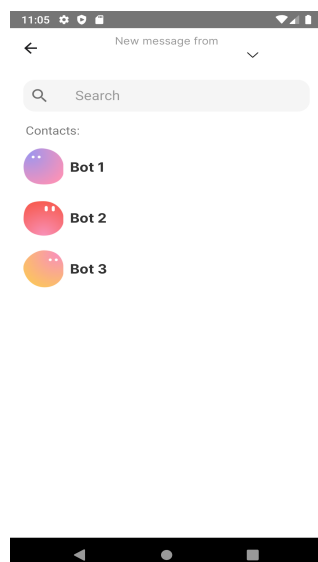


Figure 8.29: Create new message: select contact screen



Figure 8.30: Create new message: write message screen

If the user already has some exchanged messages with the other individual, pressing on the contact will direct the user to a screen where the conversation, with all the messages exchanged with that person, is displayed. (figure 8.31)



Figure 8.31: Conversation screen

## Calls

Since the purpose of this application is to facilitate communications, it is important to keep records of them. This way, the user has the possibility of checking all calls made and received through the temporary phone numbers they possess. In this case, a screen was created to allow the user to see all the records or filter through the unanswered calls (figure 8.32). If the user has no call records, the screen will appear as in figure 8.33.

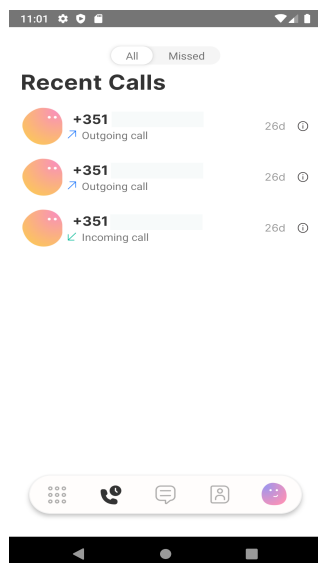


Figure 8.32: Calls history screen

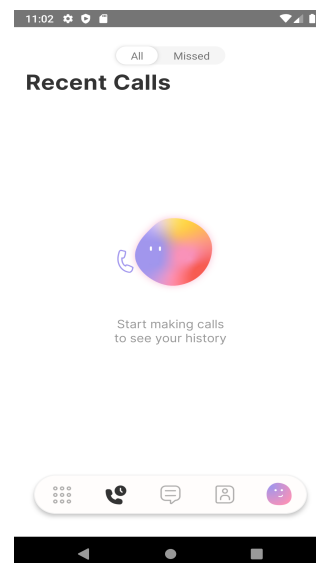


Figure 8.33: Empty call history screen

## Settings

Through the user's profile screen, the user can access the settings, where they can logout or cancel their account. The Terms of Service and Privacy Policy appear on the menu but were added for aesthetic purposes only. The image 8.34 shows the settings menu.

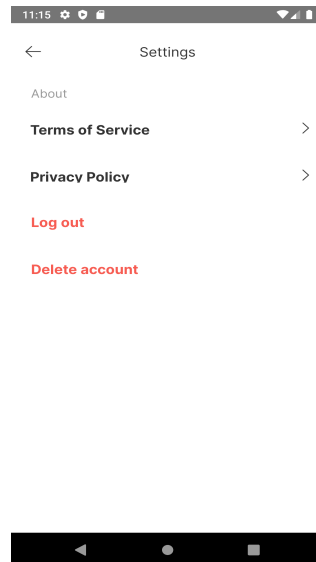


Figure 8.34: Settings screen

After having described how the system's components were implemented and demonstrating their interface (Mobile App only), the author will present how the developed software was verified and validated, in the next chapter.



This page is intentionally left blank.

## Chapter 9

# Verification and Validation

In the context of building a software product, quality assurance is one of the several phases that is associated with the process of software engineering. According to Ian Sommerville [97], quality assurance is "a planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements". Hence, this phase can occur during or after the implementation phase and is of high importance since it is where the verification and validation take place. [98]

Despite being seemingly identical, verification and validation are two different processes that are usually executed by distinctive entities and with different objectives as well, sharing only the fact that both are conducted through tests.

The difference between Verification and Validation is that Verification tries to answer the question "Have we built the product well?", and Validation tries to answer the question "Have we built what was expected?".

For a better understanding of these processes, one needs to introduce the concepts of failure, error, and fault: A failure is an instantaneous event that occurs as a result of accumulated situations and errors throughout the system execution time and causes unexpected behaviour. An error consists in an inconsistent state of the computer, usually due to bad resource management practices, generating the appearance of failures. Lastly, a fault is an abnormal condition or defect at the component level that can lead to a failure. [99]

In short, in a development process, it is necessary to identify the faults that are at the origin of the errors in order to avoid that the system shows failures. If this is not done, there is a risk of obtaining unexpected behaviours that may lead to the system malfunctioning. This way, in the context of software engineering, the processes of verification and validation are used for debugging and investigating the causes for the faults, through testing.

Each test reveals how the software product behaves when faced with a predetermined condition. Since it is not enough to execute merely one single test, as a rule, a set of tests is executed with all the previously determined conditions. The final result after each test allows for a later assessment of if the product has the expected behaviour or not.

Aside from the number of tests, one other quite important factor is the environment in which the tests are conducted, which, being varying, influence the results. This difference in behaviour that happens between environments then requires that the verification and validation steps feature tests that have various perspectives and techniques, with the purpose of maximizing the number of conditions tested. Hence, in terms of taxonomy, there

are two distinctive types of tests that can be performed:

- **Dynamic Testing:** represents all the tests that depend on execution environments. Resorts to compiled software and it is through these that failures and errors are identified.
- **Static Testing:** represents all the tests that are executed on the artifacts produced by the implementation process. They do not resort to any type of operation environment and are responsible for identifying failures.

According to the distinction made previously, it is noteworthy that the dynamic testing can be accomplished by any entity (e.g end-user or developer) of the development process, unlike the static testing that can only be executed by the entities that have technological knowledge of said artifacts.

Accomplishing all of this, this chapter intends to explain what types of testing the author carried out so as to be able to identify and validate the MVP he developed during the internship. For that, the chapter was divided in two sections, the first exposes the verification process and the second the validation process.

## 9.1. Verification

Verification is a subphase of the software development process that has the purpose of making sure that the system is in accord with what was specified [99]. In order to be able to do so, both static and dynamic testing are used. Of the tests defined by classical literature [100, 101], the ones chosen for the verification process were unit tests, integration tests and functional tests.

The unit and integration tests were conducted throughout the MVP's development, after the Sprint corresponding to each requirement implemented, whereas the functional tests were used in the final stage of the project, to show that the requirements predetermined during the first semester were accomplished or not.

### 9.1.1. Unit Testing

The unit testing is where tests are written and run by software developers to guarantee that a section of an application (known as the "unit") meets its design and behaves as intended. These must be executed continuously all throughout the development time. During the implementation phase, the intern always resorted to unit testing to ensure that the components he was implementing worked according to expectations. Although it is highly recommended to automate unit testing, the intern did it manually. In terms of strategy used to choose the unit tests, the author opted to follow the Guideline-based testing strategy, based on guidelines that reflect previous experience of the kind of errors that programmers often make when developing components.

Some of the guidelines used stemmed from Whittaker's book [102], such as:

- Choosing inputs that force the system to generate error messages.

- Forcing invalid outputs to be generated.
- Designing inputs that cause input buffers to overflow.
- Repeating the same input numerous times.

### 9.1.2. Integration Testing

After the unit tests were conducted, the author proceeded to integration tests. An integration test is “a level of software testing where individual units are combined and tested as a group”[103]. This sort of test only begins when the unit tests are seen as finished and successful. For this next stage, out of three alternatives of execution (Big Bang, Top-down or Bottom-up), the intern chose to use the Bottom-up, which considers that only when more specific components are successfully tested, can they be integrated with other broader components.

Before performing this testing, detailed design documentation where interactions between each unit are clearly defined is needed. For this reason, the author used the sequence diagrams drawn up during the system architecture phase.

The steps to integration testing were as follows:

- The Integration Test Strategy that could be adopted was determined. As stated, the intern chose the Bottom-up strategy.
- The Architecture design of the Application was studied and the Critical Modules were identified.
- The test cases were created to verify all of the interfaces in detail and test data was also defined. The intern performed this step in a rather informal manner, without creating proper documentation.
- The tests were conducted, and when defects were found, they were tracked, corrected and re-tested, until the integration tests were complete.

This whole procedure revealed quite useful, seeing as it allowed to detect interface-related errors, as well as that integrated modules worked properly. On the other hand, it was time-consuming, since it needed to verify and ensure the previously stated advantages. The integration tests done were not documented, hence they are not in this report.

### 9.1.3. Functional Testing

Functional testing is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input and cross checking the output against the functional requirements list.

Functional testing mainly involves black box testing and is not concerned about the source code of the application. This type of testing was used to check the user interface, the API, the database, the client/server communication and other functionalities of the WIT Temporary Numbers Platform. This was performed during the implementation phase although it was formally documented only at the end of the internship.

The strategy to define the Functional tests followed these steps:

- Understanding the Software Engineering Requirements.
- Identifying test input (test data).
- Computing the expected outcomes with the selected test input values.
- Executing test cases.
- Comparing of actual and computed expected result.

Table 9.1 represents an example of a functional requirement described as mentioned earlier. The remainder of the requirements are detailed the Appendix D.

<b>Id</b>	<b>Test Scenario</b>	<b>Test Steps</b>	<b>Expected Result</b>	<b>Result</b>
TC01	Registration	1) Insert valid phone number 2) Insert the OTP received	User is registered into the platform	PASS

Table 9.1: Functional test case example

## 9.2. Validation

For the purpose of validating the final output, acceptance tests are usually used, that have the purpose of validating if the final users and clients' requirements are implemented in the software product made available. This act of validation is fully supported by dynamic testing and is only conducted by the clients or final users [99].

Since the objective of the internship was to develop a prototype for the internship proposal. It was established from the beginning that the Scrum Masters and Product Owner would represent the client. Therefore, monthly meetings were conducted to make it so that the intern could present the work done and validate it.

Besides the monthly meetings, there was a presentation at the end of the internship (June 30th) for WIT's Business Development Manager with the purpose of showing the value proposal developed by the intern during the school year.

From the final meeting, the Business Development Manager showed interest in the solution developed by the intern, yet it is now up to WIT to decide on the next steps.

Having now explained the processes of verification and validation that took place during the internship, the next and last chapter will expose the conclusions and future work.

## Chapter 10

# Conclusions and Future Work

At the end of the internship, it is necessary to reflect and review the entire path that was taken by the intern during the academic year. Therefore, this last chapter contains an overview of the work done during the internship, some considerations regarding the project success, what can be done to improve this project in the future and for last, a personal analysis of the internship.

### 10.1. Overview

This internship focused on designing and implementing an MVP of a solution capable of providing temporary phone numbers. The prototype developed in the context of this internship involved several different aspects such as searching and understanding, learning of new technologies, integration with Twilio APIs.

The first challenge of the internship came with the fact that no software related to the internship's proposal had been implemented at WIT Software, S.A Therefore, there was a need to search and analyse what was available on the market in order to start designing the solution. State of the art had a major importance in this project because it allowed to understand which features the competitors used in order to start designing the MVP.

After that analysis, there was a need to define which features would compose the prototype. With that definition, the requirements phase followed, starting by user stories, use cases definition and the mockups.

An architecture for the solution was designed based on those requirements and the output didn't suffer any alterations. This was one of the longest phases of the internship because it was needed of the intern to understand which components the solution would need, which technologies, know how communications processes are done, how they flow, among others.

The implementation of the solution only took place in the second semester and despite the difficulties that arose over time, there was always the objective of validating and verifying what was implemented through meetings with the supervisors.

Although the intern did not manage to implement everything that was expected, the final result allowed to have a functional prototype of the desired solution.

## 10.2. Success Evaluation

With the end of the internship, it was essential to understand if the project was successful. In the context of software engineering, there is no specific formula that allows to calculate whether a project has been successful or not. However, there are some metrics that can be used in order to find it, such as: if the project met business requirements, if it was delivered and maintained on schedule, if it was delivered and maintained within budget, if the project delivered the expected business value [104].

From this point of view, it was defined from the start that what was expected was a functional prototype that was capable of providing and allowing communications through temporary phone numbers. Following this line of thought, the solution designed by the intern was made up of three components, and even though not all of the platform's components were implemented, the final result was a functional prototype that allows the purchase and use of temporary numbers, fulfilling the requirements defined for the mobile app. Only the platform's management was left out, which would be done through a Web Portal (Admin Dashboard). This was agreed on by WIT, prioritizing other parts of the project.

In sum, the expected outcome of the internship was not fully accomplished. Yet, the artifacts produced during the internship can be useful for WIT to understand the next steps to take in the future.

## 10.3. Future Work

There are opportunities for future work derived from unimplemented requirements, as well as from the evolution of the MVP. Since the Admin Dashboard was not implemented but was fully planned, that is the starting point for work to be done in the future to complete the proposed MVP.

The following steps could be to implement possible improvements to the prototype, some of which were already identified by the intern together with his supervisors, during the implementation phase.

Lastly, the MVP should be evaluated by possible external clients and if it proves that this value proposition has a place in the market, then a transition to the commercial product should take place.

## 10.4. Final Considerations

This internship was an extraordinary opportunity for professional and personal improvement. In the face of what was undoubtedly the greatest challenge the intern was ever posed with, he is pleased to say that the majority of the goals set in the beginning of this

task were carried out. However, several problems emerged during this unexpected moment of lockdown. The difficulties were discussed with the project team and actions were designed to minimize its negative impact (e.g. project extension). There is still a feeling of accomplishment in bringing a project from its creation all the way to its end, overcoming obstacles and taking every step until it becomes a finished product.

Such a venture required that the author put to use the knowledge acquired over time while studying and that he learn even more, deepening his understanding in several technologies and getting acquainted with new ones, and this is sure to be a valuable experience to take as he makes his way as a software engineer.

The intern had the opportunity to conduct the project at WIT with an extensive degree of autonomy, in an entirely new product, with the valuable guidance provided by the company experts.

Overall, it was a rewarding experience that built the author's confidence in his own abilities, challenging him and making him evolve in this field of expertise.



This page is intentionally left blank.

# References

- [1] JavaTPoint. Spring boot architecture. <https://www.javatpoint.com/spring-boot-architecture>. Accessed on July.
- [2] techopedia. Service provider. <https://www.techopedia.com/definition/22021/service-provider>. Accessed on December.
- [3] Wit-Software SA. *Wit Software Bluebook* , 2015. Accessed on October.
- [4] M-STAT. The evolution of telecommunications. <https://www.m-stat.gr/the-evolution-of-telecommunications>. Accessed on December.
- [5] Frederic Gonzalo. When is a mobile used? <http://fredericgonzalo.com/wp-content/uploads/2012/04/Screen-Shot-2012-04-21-at-3.32.38-PM.png>. Accessed on December.
- [6] Jackie Dove. The best apps for a second phone number. <https://www.digitaltrends.com/mobile/best-apps-for-a-second-phone-number/>. Accessed on December.
- [7] techopedia. What does minimum viable product (mvp) mean? <https://www.techopedia.com/definition/27809/minimum-viable-product-mvp>. Accessed on December.
- [8] Alan Quayle. The growing importance of the phone number beyond the call. <http://alanquayle.com/2015/04/importance-phone-number-beyond-the-call/>. Accessed on Jun.
- [9] FREEZVON. Classification of phone numbers. <https://freezvon.com/services/phone-numbers>. Accessed on June.
- [10] Microsoft. Different kinds of phone numbers used for calling plans. <https://docs.microsoft.com/en-us/microsoftteams/different-kinds-of-phone-numbers-used-for-calling-plans>. Accessed on May.
- [11] Ofcom. Geographic telephone numbers. <https://www.ofcom.org.uk/consultations-and-statements/category-1/geographic-numbers>. Accessed on May.
- [12] Ofcom. Non-geographic call services. <https://www.ofcom.org.uk/phones-telecoms-and-internet/information-for-industry/policy/non-geo-call-services>. Accessed on May.
- [13] Simple Textin. What is a landline number? <https://simpletexting.com/what-is-a-landline-number/>. Accessed on June.

- [14] Formulate Information Design. Landline phone numbers in electronic forms. <https://www.formulate.com.au/blog/landline-phone-numbers-in-electronic-forms/>. Accessed on May.
- [15] Sonetel. Are there different types of numbers? <https://sonetel.com/pt-pt/help/help-topics/phone-numbers/are-there-different-types-of-phone-numbers/>. Accessed on May.
- [16] Dial9 Communications. Internal numbers. <https://www.dial9.co.uk/support/dial-9-connect/numbers/internal-numbers>. Accessed on Jun.
- [17] Twilio. Short code. <https://www.twilio.com/docs/glossary/what-is-a-short-code>. Accessed on June.
- [18] Justia. Unwanted telemarketing. <https://www.justia.com/consumer/deceptive-practices-and-fraud/unwanted-telemarketing/>. Accessed on December.
- [19] Oag. Overwhelmed by robocalls? <https://oag.dc.gov/blog/overwhelmed-robocalls-here-are-5-tips-help>. Accessed on December.
- [20] Sean Captain. Stop giving companies your phone number. <https://www.fastcompany.com/90415625/stop-giving-companies-your-phone-number-do-this-instead>. Accessed on December.
- [21] Tech Suport. Benefits of disposable temporary phone numbers. <https://www.davestechsupport.net/blog/benefits-of-disposable-temporary-phone-numbers/>. Accessed on October.
- [22] Hushed. What are disposable numbers and how do they benefit us. <https://hushed.com/features/disposable-numbers>. Accessed on October.
- [23] Twilio. Guidelines. <https://www.twilio.com/guidelines/pt/regulatory>. Accessed on June.
- [24] Plivo. Local regulations. <https://www.plivo.com/docs/numbers/api/verification/address#delete-an-address>. Accessed on June.
- [25] Keepsafe Software Zouhair Belkoura, CEO Co-Founder. Can someone find out your real number if you're using a burner app number? <https://www.quora.com/Can-someone-find-out-your-real-number-if-youre-using-a-burner-app-number>. Accessed on October.
- [26] Telesign. Number deactivation and the recycled phone number dilemma. <https://www.telesign.com/blog/post/number-deactivation-and-the-recycled-phone-number-dilemma>. Accessed on June.
- [27] Twilio. Cancel or release a twilio number. <https://support.twilio.com/hc/en-us/articles/223183028-Cancel-or-release-a-Twilio-numbea>. Accessed on Jun.
- [28] giffgaff. Why is my phone number deactivated? <https://www.giffgaff.com/help/articles/why-is-my-phone-number-deactivated>. Accessed on June.
- [29] Twilio. Emergency calling. <https://www.twilio.com/docs/sip-trunking/emergency-calling>. Accessed on October.

- 
- [30] Nexmo. Can i call emergency services (e.g. 911, 999, 112). <https://help.nexmo.com/hc/en-us/articles/207424127-Can-I-call-Emergency-Services-e-g-911-999-112-using-Nexmo-E911->. Accessed on october.
- [31] Vonage. Traditional 911 and vonage 911 dialing. <https://support.vonage.com/articles/answer/Traditional-911-and-Vonage-911-Dialing-1110>. Accessed on October.
- [32] RingCentral. Ringcentral emergency services policy. <https://www.ringcentral.com/legal/last-update-October-15-2019/emergency-services.html>. Accessed on October.
- [33] Mike Chu. Can a burner phone number be traced + 19 steps to get one properly? <https://dataoverhauleders.com/can-a-burner-phone-number-be-traced/>. Accessed on NOvember.
- [34] techtarget. Private branch exchange (pbx). <https://searchunifiedcommunications.techtarget.com/definition/private-branch-exchange>. Accessed on October.
- [35] Atlantech. Pbx providers. <https://www.atlantech.net/blog/the-5-best-hosted-pbx-providers-for-businesses-of-all-sizes>. Accessed on October.
- [36] RingCentral. Ringcentral. <https://www.ringcentral.com>. Accessed on October.
- [37] Wikipedia. Vonage. <https://en.wikipedia.org/wiki/Vonage>. Accessed on October.
- [38] Atlantech. Atlantech. <https://www.atlantech.net>. Accessed on October.
- [39] Techtarget. Cpaas definition. <https://searchunifiedcommunications.techtarget.com/definition/Communications-platform-as-a-service-CPaaS>. Accessed on October.
- [40] RadioStudio. The most popular cpaas providers of 2019. <https://radiostud.io/the-most-popular-cpaas-providers>. Accessed on October.
- [41] Twilio. Twilio. <https://www.twilio.com>. Accessed on October.
- [42] Nexmo. Nexmo. <https://www.nexmo.com>. Accessed on October.
- [43] Plivo. Plivo. <https://www.plivo.com>. Accessed on October.
- [44] Talkroute. Diference between carrier and service provider. <https://talkroute.com/carriers-service-providers-whats-the-difference>. Accessed on October.
- [45] Twilio. Twilio. <https://www.twilio.com>. Accessed on October.
- [46] Nexmo. Nexmo. <https://www.nexmo.com>. Accessed on October.
- [47] Plivo. Plivo. <https://www.plivo.com>. Accessed on October.
- [48] AffinityClick. Hushed. <https://affinityclick.com/hushed/>. Accessed on October.
- [49] Wikipedia. Line2. <https://en.wikipedia.org/wiki/Line2>. Accessed on October.

- [50] Cloud SIM. Cloud sim app. <https://www.cloudsimapp.com>. Accessed on October.
- [51] Wikipedia. Pinger. <https://en.wikipedia.org/wiki/Pinger>. Accessed on October.
- [52] Sideline. Sideline. <https://www.sideline.com>. Accessed on October.
- [53] Swapp. Swapp. <https://www.swapp.pt>. Accessed on October.
- [54] Wikipedia. Meo. [https://en.wikipedia.org/wiki/Meo\\_\(telecommunication\\_service\)](https://en.wikipedia.org/wiki/Meo_(telecommunication_service)). Accessed on October.
- [55] Susan E. Reid and Ulrike de Brentani. *Software Architecture in Practice*, 2010. Accessed on June.
- [56] Scrum.org. What is scrum? <https://www.scrum.org/resources/what-is-scrum>. Accessed on December.
- [57] Ken Schwaber. *SCRUM Development Process*, 1997. Accessed on September.
- [58] Scrum.org. Scrum framework. [https://s3.amazonaws.com/scrumorg-website-prod/drupal/2016-06/ScrumFramework\\_17x11.pdf](https://s3.amazonaws.com/scrumorg-website-prod/drupal/2016-06/ScrumFramework_17x11.pdf). Accessed on December.
- [59] ScrumGuide.org. Scrum guides. <https://www.scrumguides.org/scrums-guide.html>. Accessed on December.
- [60] Cast. Risk management in software development and software engineering projects. <https://www.castsoftware.com/research-labs/risk-management-in-software-development-and-software-engineering-projects>. Accessed on December.
- [61] ProfessionalQA.com. Risk management activities. <http://www.professionalqa.com/risk-management-activity>. Accessed on December.
- [62] Project Management Institute. Risk analysis and management. <https://www.pmi.org/learning/library/risk-analysis-project-management-7070>. Accessed on December.
- [63] Julian Talbot. What's right with risk matrices? <https://www.juliantalbot.com/post/2018/07/31/whats-right-with-risk-matrices>. Accessed on December.
- [64] stakeholdermap.com. Risk management risk analysis templates and advice. <https://www.stakeholdermap.com/risk/risk-assessment-matrix-simple-3x3.html#risk>. Accessed on December.
- [65] geeksforgeeks.org. Software engineering | requirements elicitation. <https://www.geeksforgeeks.org/software-engineering-requirements-elicitation/>. Accessed on December.
- [66] Raul Sidnei Wazlawick. *Object-Oriented Analysis and Design for Information Systems*, 2014. Accessed on December.
- [67] MOUNTAIN GOAT SOFTWARE. User stories. <https://www.mountangoatsoftware.com/agile/user-stories>. Accessed on December.
- [68] Marwan H. Soliman. The skeleton of user stories. [https://medium.com/@marwan\\_hamdy/the-skeleton-of-user-stories-718c7e78dd83](https://medium.com/@marwan_hamdy/the-skeleton-of-user-stories-718c7e78dd83). Accessed on December.

- 
- [69] usability.gov. Use cases. <https://www.usability.gov/how-to-and-tools/methods/use-cases.html>. Accessed on December.
- [70] Product Plan. Moscow prioritization. <https://www.productplan.com/glossary/moscow-prioritization/>. Accessed on December.
- [71] <http://faculty.washington.edu/>. Use case template by alistair cockburn. <http://faculty.washington.edu/jtenenbg/courses/360/f01/project/usecasetemplate.html>. Accessed on December.
- [72] Ravi Prakash Gorthi Vikas Bajpai. *On Non-Functional Requirements: A Survey*, 2012. Accessed on July.
- [73] Nikolay Ashanin. Quality attributes in software architecture. <https://medium.com/@nvashanin/quality-attributes-in-software-architecture-3844ea482732>. Accessed on Jun.
- [74] Tonya Smyrnova. Software quality attributes and their impact on your business. <https://syndicode.com/2019/06/14/software-quality-attributes-and-their-impact-on-your-business/>. Accessed on Jun.
- [75] Paul Clements and Rick Kazman. *Market Vision and Market Visioning Competence: Impact on Early Performance for Radically New, High-Tech Products*, 2010. Accessed on June.
- [76] Wen Tao. What makes good software architecture. <https://medium.com/software-engineering-problems/what-makes-up-the-software-20f607da9155>. Accessed on December.
- [77] Lea Maya Karam. The importance of good software architecture. <https://dzone.com/articles/the-importance-of-a-good-software-architecture>. Accessed on December.
- [78] Simon Brown. The c4 model for visualising software architecture. <https://c4model.com>. Accessed on June.
- [79] toturialspoint. Spring boot - introduction. [https://www.tutorialspoint.com/spring\\_boot/spring\\_boot\\_introduction.htm](https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm). Accessed on August.
- [80] CCorner. What and why react. <https://www.c-sharpcorner.com/article/what-and-why-reactjs/>. Accessed on Jun.
- [81] toturialspoint. Postgresql. [https://www.tutorialspoint.com/postgresql/postgresql\\_overview.htm](https://www.tutorialspoint.com/postgresql/postgresql_overview.htm). Accessed on Aug.
- [82] CloudFlare. What is https? <https://www.cloudflare.com/learning/ssl/what-is-https/>. Accessed on Jul.
- [83] w3schools. What is json? [https://www.w3schools.com/whatis/whatis\\_json.asp](https://www.w3schools.com/whatis/whatis_json.asp). Accessed on August.
- [84] infobit. Otp (one-time pin) code. <https://www.infobip.com/glossary/otp-one-time-pin-code>. Accessed on December.
- [85] TwilioDocs. Verify api. <https://www.twilio.com/docs/verify/api>. Accessed on December.

- [86] JWT. Introduction to json web tokens. <https://jwt.io>. Accessed on December.
- [87] Tutorial and Example. Spring boot architecture. <https://www.tutorialandexample.com/spring-boot-architecture/>. Accessed on July.
- [88] restfulapi.net. Rest architectural constraints. <https://restfulapi.net/rest-architectural-constraints/>. Accessed on August.
- [89] GeeksforGeeks. Rest api architectural constraints. <https://www.geeksforgeeks.org/rest-api-architectural-constraints//>. Accessed on August.
- [90] spring. Spring security. <https://spring.io/projects/spring-security>. Accessed on August.
- [91] totutorialspoint. Spring boot - servlet filter. [https://www.tutorialspoint.com/spring\\_boot/spring\\_boot\\_servlet\\_filter.htm](https://www.tutorialspoint.com/spring_boot/spring_boot_servlet_filter.htm). Accessed on August.
- [92] javaTpoint. Exception handling in java. <https://www.javatpoint.com/exception-handling-in-java>. Accessed on August.
- [93] PostgreSQL. Postgresql: The world's most advanced open source relational database. <https://www.postgresql.org/>. Accessed on June.
- [94] Docker. Docker overview. <https://docs.docker.com/get-started/overview/>. Accessed on June.
- [95] DBeaver. Dbeaver community. <https://dbeaver.io/>. Accessed on June.
- [96] Flyway. Version control for your database. <https://flywaydb.org/>. Accessed on June.
- [97] Ian Sommerville. *Software Engineering*, 2011. Accessed on July.
- [98] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*, 2010. Accessed on June.
- [99] Sérgio Guerreiro. *Introdução à Engenharia de Software*, 2015. Accessed on June.
- [100] Ian Sommerville. *Software Engineering*, 2010. Accessed on August.
- [101] Roger S. Pressman. *Software Engineering A Practitioners Approach*, 2010. Accessed on August.
- [102] James A. Whittaker. *Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design*, 2011. Accessed on June.
- [103] softwaretestingfundamentals.com. Integration testing. <https://softwaretestingfundamentals.com/software-testing-levels/>. Accessed on August.
- [104] DoIT. What makes a successful project? <https://doit.maryland.gov/SDLC/Documents/What%20Makes%20a%20Successful%20Project.pdf>. Accessed on August.

# Appendices



This page is intentionally left blank.

# Appendix A

## Requirements

### A.1. Functional Requirements

In this section are detailed the functional requirements of the two major components of the system, the mobile application and dashboard, following the notation described in Chapter 5, section 5.2.

FR01 - Registration	
<b>Primary Actor</b>	Non Registered User
<b>Description</b>	User enters his phone number in order to make the registry
<b>Priority</b>	Must have
<b>Pre-conditions</b>	User has the application installed User has access to an internet connection User has a valid phone number
<b>Post-condition success</b>	User registers and can start using the application
<b>Main Success Scenario</b>	<ol style="list-style-type: none"> <li>1. User inserts his phone number</li> <li>2. System verify the inserted number</li> <li>3. The System sends a verification code (via SMS) to user</li> <li>4. The user is redirect to verification screen</li> <li>5. User inserts the code received</li> <li>6. The system validates the code</li> <li>7. User is redirect to the main page</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>2a Number inserted by user already exists               <ol style="list-style-type: none"> <li>2a1. User is informed that number is already registered</li> </ol> </li> <li>2b Number inserted is not valid               <ol style="list-style-type: none"> <li>2b1. Error message is shown</li> </ol> </li> <li>5a. Code inserted don't match               <ol style="list-style-type: none"> <li>5a1. Error message is shown</li> <li>5a2. The user tries again</li> <li>5a3. User presses a button to receive another code</li> </ol> </li> <li>5a. Code inserted don't match               <ol style="list-style-type: none"> <li>5a1. Error message is shown</li> <li>5a2. The user tries again</li> <li>5a3. User presses a button to receive another code</li> </ol> </li> </ol>

Table A.1: Functional Requirement 01 - Registration

RF02 - Sign In	
<b>Primary Actor</b>	Registered User
<b>Description</b>	User enters his phone number in order sign in
<b>Priority</b>	Must have
<b>Pre-conditions</b>	User has the application installed User has access to a internet connection
<b>Post-condition success</b>	User signs in and is able to start using the application
<b>Main Success Scenario</b>	<ol style="list-style-type: none"> <li>1. User inserts his phone number</li> <li>2. System verify the number</li> <li>3. System sends a verification code (via SMS) to user</li> <li>4. User is redirect to verification screen</li> <li>5. User inserts the code received</li> <li>6. System validates the code</li> <li>7. User is redirect to the main page</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>2a Number inserted is not registered               <ol style="list-style-type: none"> <li>2a1. Error message is shown</li> </ol> </li> <li>2b Number inserted is not valid               <ol style="list-style-type: none"> <li>2b1. Error message is shown</li> </ol> </li> <li>5a. Code inserted don't match               <ol style="list-style-type: none"> <li>5a1. Error message is shown</li> <li>5a2. The user tries again</li> <li>5a3. User presses a button to receive another code</li> </ol> </li> <li>5a. Code inserted don't match               <ol style="list-style-type: none"> <li>5a1. Error message is shown</li> <li>5a2. User tries again</li> <li>5a3. User presses a button to receive another code</li> </ol> </li> </ol>

Table A.2: Functional Requirement 02 - Sign In

RF03 - Show Main Menu	
<b>Primary Actor</b>	Registered User
<b>Description</b>	User requests the main menu to see the available options.
<b>Priority</b>	Must have
<b>Pre-conditions</b>	User has the application installed User has access to a internet connection User has to be logged into the app
<b>Post-condition success</b>	User is presented the main menu and is able to select any desired option
<b>Main Success Scenario</b>	<ol style="list-style-type: none"> <li>1. User can see the numbers that he purchased</li> <li>2. User presses one of the numbers</li> <li>3. User is redirected to a new screen</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>1a User does not have any numbers purchased.               <ol style="list-style-type: none"> <li>1a1. User buys one or more numbers</li> </ol> </li> </ol>

Table A.3: Functional Requirement 03 - Show Main Menu

RF04 - Buy a Number	
<b>Primary Actor</b>	Registered User
<b>Description</b>	User asks for the main menu in order to see the available options
<b>Priority</b>	Must have
<b>Pre-conditions</b>	User has the application installed User has access to a internet connection User has to be logged into the app
<b>Post-condition success</b>	User is able to buy a number
<b>Main Success Scenario</b>	1. User presses the button to buy a number 2. User chooses the country 3. User chooses the type of number (mobile or local) 4. User chooses the plan 5. User reviews the purchase 6. User finishes the process clicking on a button
<b>Extensions</b>	6a User do not have enough balance to make the purchase 6a1. User adds funds and tries again

Table A.4: Functional Requirement 04 - Buy Number

RF05 - Make a Call	
<b>Primary Actor</b>	Registered User
<b>Description</b>	User dials a number through a keypad screen in order to establish a call with other person
<b>Priority</b>	Must have
<b>Pre-conditions</b>	User has the application installed User has access to an internet connection User has to be logged into the app User has minutes available to spend
<b>Post-condition success</b>	User makes a call
<b>Main Success Scenario</b>	1. User dials a number 2. User presses the call button 3. A call is initiated
<b>Extensions</b>	2a The number format is not correct 2a1. Error message is shown 2b The user does not have minutes available to spend. 2b1. Error message is shown

Table A.5: Functional Requirement 05 - Make a Call

RF06 - Show Calls History	
<b>Primary Actor</b>	Registered User
<b>Description</b>	The user should be able to see all the call logs from the number he had chosen in the main menu
<b>Priority</b>	Must have
<b>Pre-conditions</b>	User has the application installed User has access to an internet connection User has to be logged into the app
<b>Post-condition success</b>	User is able to call a person
<b>Main Success Scenario</b>	1. User accesses the "Calls" menu 2. User sees the logs 3. User scrolls up and down to see the logs
<b>Extensions</b>	2a User does not see any logs 2a1. User refreshes the page

Table A.6: Functional Requirement 06 - Show calls history

RF07 - Send a Message	
<b>Primary Actor</b>	Registered User
<b>Description</b>	The user should be able to write and send a message
<b>Priority</b>	Must have
<b>Pre-conditions</b>	User has the application installed User has access to an internet connection User has to be logged into the app User has SMS available to spend
<b>Post-condition success</b>	User sends a message
<b>Main Success Scenario</b>	1. User accesses the "Messages" menu 2. User presses a button to write a message 3. User writes a message 4. User presses a button to send the message 5. Message is sent
<b>Extensions</b>	5a User plans do not allow the operation 5a1. Error message is shown

Table A.7: Functional Requirement 07 - Send a Message

RF08 - Messages History	
<b>Primary Actor</b>	Registered User
<b>Description</b>	The user should be capable to see all the call logs from the number he had chosen in the main menu
<b>Priority</b>	Must have
<b>Pre-conditions</b>	User has the application installed User has access to an internet connection User has to be logged into the app
<b>Post-condition success</b>	User is able to call a person
<b>Main Success Scenario</b>	1. User accesses the "Messages" menu 2. User sees the messages history 3. User scroll up and down to see the logs
<b>Extensions</b>	2a User can not see any logs 2a1. User refreshes the page

Table A.8: Functional Requirement 08 - Messages History

RF09 - Show Contacts	
<b>Primary Actor</b>	Registered User
<b>Description</b>	The user should be capable to see all imported contacts in order to send messages or make calls
<b>Priority</b>	Must have
<b>Pre-conditions</b>	User has the application installed User has access to an internet connection User has to be logged into the app User has to give permission to access contacts
<b>Post-condition success</b>	User imports his contacts
<b>Main Success Scenario</b>	1. User accesses the "Contacts" menu 2a. User sees his contacts 2b. User imports his contacts to the App
<b>Extensions</b>	2b User does not give the permission needed 2a1. Warning message is shown

Table A.9: Functional Requirement 09 - Show Contacts

RF010 - Show Settings	
<b>Primary Actor</b>	Registered User
<b>Description</b>	In the "Settings" menu user should be capable to cancel a number, delete his account, buy a new number, see the capabilities of the number we own and the plan which he is subscribed
<b>Priority</b>	Must have
<b>Pre-conditions</b>	User has the application installed User has access to an internet connection User has to be logged into the app
<b>Post-condition success</b>	User is able to see the settings menu
<b>Main Success Scenario</b>	1. User accesses the "Settings" menu 2. User settings are shown
<b>Extensions</b>	2a User does not see anything 2a1. User refreshes the page

Table A.10: Functional Requirement 10 - Show Settings

RF11 - Add Funds	
<b>Primary Actor</b>	Registered User
<b>Description</b>	The user should be capable of adding funds to his account
<b>Priority</b>	Could have
<b>Pre-conditions</b>	User has the application installed User has access to an internet connection User has to be logged into the app
<b>Post-condition success</b>	User is able to add funds into his account
<b>Main Success Scenario</b>	1. User accesses the "Settings" menu 2. User presses the "Add Funds" button 3. User chooses the payment method 4. A confirmation message is shown to the user
<b>Extensions</b>	4a. Confirmation message is not shown 4a1. User tries again

Table A.11: Functional Requirement 11 - Add Funds

RF012 - Cancel a Number	
<b>Primary Actor</b>	Registered User
<b>Description</b>	The user should be capable to cancel temporary numbers
<b>Priority</b>	Must have
<b>Pre-conditions</b>	User has the application installed User has access to an internet connection User has to be logged into the app
<b>Post-condition success</b>	User is able to cancel the temporary number
<b>Main Success Scenario</b>	1. User accesses the "Settings" menu 2. User presses the "Cancel Number" button 3. Number is canceled 4. A confirmation message is shown to the user
<b>Extensions</b>	4a. Confirmation message is not shown 4a1. User tries again

Table A.12: Functional Requirement 12 - Cancel a Number

RF13 - Cancel an Account	
<b>Primary Actor</b>	Registered User
<b>Description</b>	The user should be capable to cancel his account
<b>Priority</b>	Must have
<b>Pre-conditions</b>	User has the application installed User has access to an internet connection User has to be logged into the app
<b>Post-condition success</b>	User is able to cancel his account
<b>Main Success Scenario</b>	1. User accesses the "Settings" menu 2. User presses the "Cancel Account" button 3. Account is canceled 4. User is redirect to the Sign In/ Registration screen
<b>Extensions</b>	4a. User is not redirected 4a1. User tries again

Table A.13: Functional Requirement 13 - Cancel an Account

RF14 - System Overview Menu	
<b>Primary Actor</b>	Admin
<b>Description</b>	The admin should be able to see statistics about the system
<b>Priority</b>	Must have
<b>Pre-conditions</b>	Admin has access to an internet connection Admin has to be logged into the Dashboard
<b>Post-condition success</b>	Admin sees the system statistics
<b>Main Success Scenario</b>	1. Admin accesses the "System Overview" menu 2. Admin scrolls up and down to see the system statistics
<b>Extensions</b>	2a Admin does not see anything 2a1. Admin refreshes the page

Table A.14: Functional Requirement 14 - System Overview Menu

RF15 - Managing Menu	
<b>Primary Actor</b>	Admin
<b>Description</b>	The admin should be able to see a page where he can perform managing operations (e.g. notify a user, delete a user account, etc)
<b>Priority</b>	Must have
<b>Pre-conditions</b>	Admin has access to an internet connection Admin has to be logged into the Dashboard
<b>Post-condition success</b>	Admin sees the managing page
<b>Main Success Scenario</b>	1. Admin accesses the "Manage System" menu 2. Admin uses the managing features provided
<b>Extensions</b>	

Table A.15: Functional Requirement 15 - Managing Menu

RF13 - Add/Remove Funds to/from a User Account	
<b>Primary Actor</b>	Admin
<b>Description</b>	Admin is able to add/remove funds from the user account
<b>Priority</b>	Must have
<b>Pre-conditions</b>	Admin has access to an internet connection Admin has to be logged into the Dashboard
<b>Post-condition success</b>	The admin adds/removes funds from a user balance
<b>Main Success Scenario</b>	1. Admin accesses the "Manage System" menu 2. Admin inserts the amount to add/remove to user balance 3. User balance is updated
<b>Extensions</b>	

Table A.16: Functional Requirement 13 - Add/Remove Funds to/from a User Account

RF15 - Send a Notification/Alert	
<b>Primary Actor</b>	Admin
<b>Description</b>	The admin could send alerts and notifications to the users
<b>Priority</b>	Must have
<b>Pre-conditions</b>	Admin has access to an internet connection Admin has to be logged into the Dashboard
<b>Post-condition success</b>	Admin sends a notification/alert to a user
<b>Main Success Scenario</b>	1. Admin accesses the "Manage System" menu 2. Admin writes the notification/alert 3. Admin presses a button to send the notification/alert 3. Notification is sent to the user selected
<b>Extensions</b>	

Table A.17: Functional Requirement 15 - Send a Notification

RF19 - Delete User Account	
<b>Primary Actor</b>	Admin
<b>Description</b>	The admin can search for an user and then close his account
<b>Priority</b>	Must have
<b>Pre-conditions</b>	Admin has access to an internet connection Admin has to be logged into the Dashboard
<b>Post-condition success</b>	Admin sends a notification to a user
<b>Main Success Scenario</b>	1. Admin accesses the "Manage System" menu 2. Admin search for the user 3. Admin selects the user 4. Admin presses the button to close user account 5. Account is closed
<b>Extensions</b>	

Table A.18: Functional Requirement 19 - Delete User Account



## A.2. Non Functional Requirements

In this section are detailed the quality attributes scenarios for the quality attributes defined (Availability, Security , Interoperability and Modifiability) in Chapter 5, section 5.3.

NFR01 - Availability	
<b>Source of Stimulus:</b>	Internal to the system
<b>Stimulus</b>	A component responds but the response late.
<b>Environment Condition(s):</b>	Normal operation
<b>System Response:</b>	The fault is detected and logged.
<b>Significant Measures:</b>	Time to detect the fault.

Table A.19: Quality Attribute scenario 01 - Availability

NFR02 - Security	
<b>Source of Stimulus:</b>	External to the system - malicious entity
<b>Stimulus</b>	An unauthorized user tries to access system services, resource or data.
<b>Environment Condition(s):</b>	The system is online and fully functional.
<b>System Response:</b>	Transactions are carried out in a fashion such that services are protected from unauthorized access. Record the attempts.
<b>Significant Measures:</b>	Number of attacks resisted.

Table A.20: Quality Attribute scenario 02 - Security

NFR03 - Interoperability:	
<b>Source of Stimulus:</b>	The system initiates a request to interoperate with the CPaaS Provider.
<b>Stimulus:</b>	A request to make a call.
<b>Environment Condition(s):</b>	Both systems are in runtime.
<b>System Response:</b>	The request is accepted and the call is initiated.
<b>Significant Measures:</b>	The percentage of calls correctly processed.

Table A.21: Quality Attribute scenario 03 - Interoperability

NFR04 - Modifiability	
<b>Source of Stimulus:</b>	End User
<b>Stimulus</b>	Wants to add a new CPaaS Provider to the system.
<b>Environment Condition(s):</b>	Runtime
<b>System Response:</b>	The change is made, tested and deployed.
<b>Significant Measures:</b>	Number of system modules affected by the new modification.

Table A.22: Quality Attribute scenario 04 - Modifiability

This page is intentionally left blank.

# Appendix B

## Project Risks

### B.1. Risks

In this section are detailed the project's risks as described in Chapter 4, section 4.3.

Schedules	
<b>ID</b>	R02
<b>Date of Identification</b>	January
<b>Description</b>	A functional prototype of the platform is expected by July of 2020. The deadline could be too ambitious to deliver a prototype containing all the requirements defined.
<b>Impact</b>	High
<b>Probability</b>	High
<b>Mitigation Plan</b>	Ensure that the tasks assigned to the intern are viable to complete during the estimated time. In last resort delay the delivery to September.

Table B.1: Risk 02 - Schedules

R3 - Integration with External Entities	
<b>ID</b>	R03
<b>Date of Identification</b>	February
<b>Description</b>	Since the WIT API requires integration with Twilio APIs, the data will be obtained through requests. The form of how the requests are made to the external APIs could be changed any moment by the provider, making the WIT API broken. Also, changes in legislation could happen, leading provider to make changes to the way it provides phone numbers.
<b>Impact</b>	High
<b>Probability</b>	Medium
<b>Mitigation Plan</b>	Read the changelog of Twilio and make the necessary changes to assure that integration is not compromised. If any change occurs, arrange a meeting with the Project Owner and Scrum Masters to decide the actions to set in motion.

Table B.2: Risk 03 - Integration with External Entities

R4 - Technologies Learning	
<b>ID</b>	R04
<b>Date of Identification</b>	December
<b>Description</b>	The intern has to deal with new technologies he is not familiarized with (Spring Boot, React, React Native and Twilio API's) to build the platform. Difficulties in using these technologies can cause delays in the execution of the tasks.
<b>Impact</b>	High
<b>Probability</b>	High
<b>Mitigation Plan</b>	Do some tutorials online and ask for help from more experienced colleagues at WIT Software that are available.

Table B.3: Risk 04 - Technologies Learning

Requirements Change	
<b>ID</b>	R05
<b>Date of Identification</b>	February
<b>Description</b>	Since the author will use an agile methodology (Scrum), some requirements may face some changes during the project.
<b>Impact</b>	High
<b>Probability</b>	Low
<b>Mitigation Plan</b>	Arrange meetings with the Product Owner to avoid possible misleadings and to assure that the project goes accordingly to the demands.

Table B.4: Risk 05 - Requirements Change

Tests	
<b>ID</b>	R06
<b>Date of Identification</b>	March
<b>Description</b>	As the trainee is implementing functionalities that integrate external components, it is necessary to test them and ensure that they work. Testing the features will, therefore, be a time-consuming task since external services used required some attention from a legal standpoint.
<b>Impact</b>	High
<b>Probability</b>	High
<b>Mitigation Plan</b>	Read and understand the documentation of the APIs used. In case of doubts regarding external services, contact the supplier to clarify them. Make a script with all test cases for a feature before start testing.

Table B.5: Risk 06 - Tests

Pandemic Situation	
<b>ID</b>	R07
<b>Date of Identification</b>	March
<b>Description</b>	The worldwide pandemic of COVID-19 led the intern to work remotely. The fact that the intern must be away from the work environment can affect productivity and communication during the implementation of the solution.
<b>Impact</b>	High
<b>Probability</b>	Low
<b>Mitigation Plan</b>	Ensure that the Scrum Master and the respective Product Owner are aware of the status of the project. Report delays and problems encountered.

Table B.6: Risk 07 - Pandemic Situation

Project Validation	
<b>ID</b>	R08
<b>Date of Identification</b>	June
<b>Description</b>	A meeting will be held between the Scrum Team and the Business Development Director to present the solution designed and implemented by the intern. The MVP presented may not correspond to expectations and requirements.
<b>Impact</b>	High
<b>Probability</b>	Medium
<b>Mitigation Plan</b>	Realize which requirements or expectations are not met and then arrange a meeting between the Development Team and the Scrum Masters to decide the actions to be taken.

Table B.7: Risk 08 - Project Validation

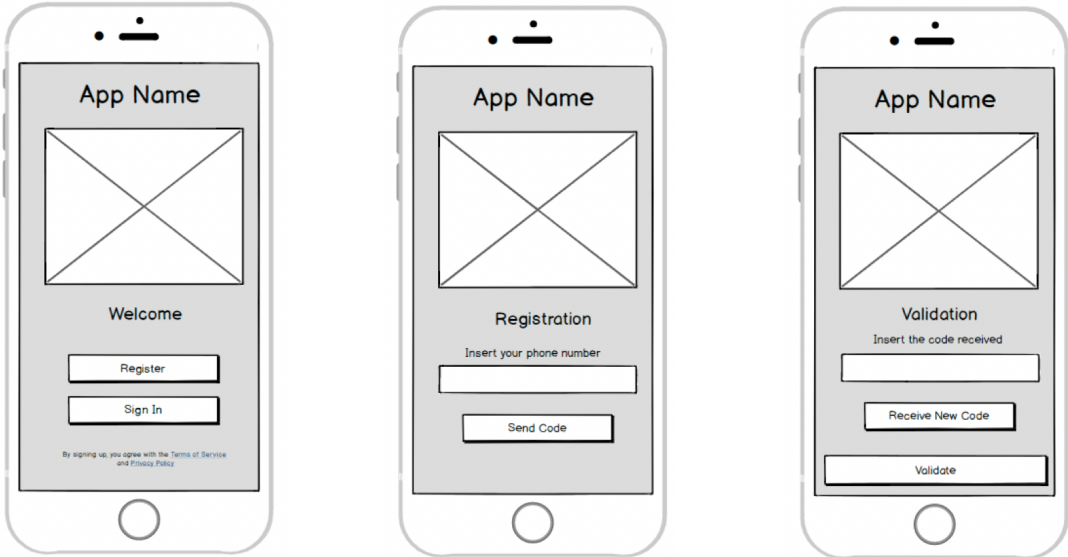
This page is intentionally left blank.

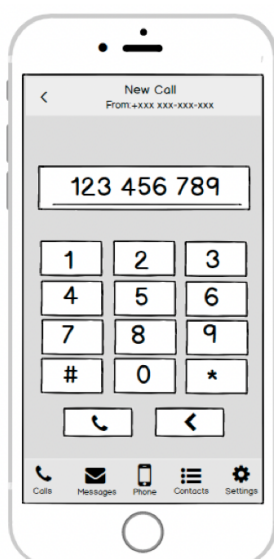
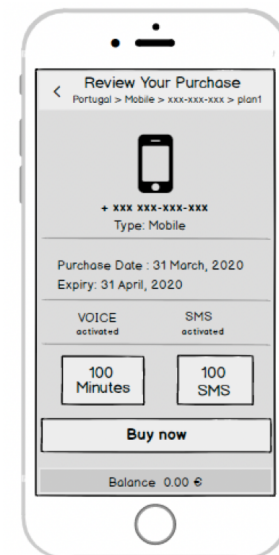
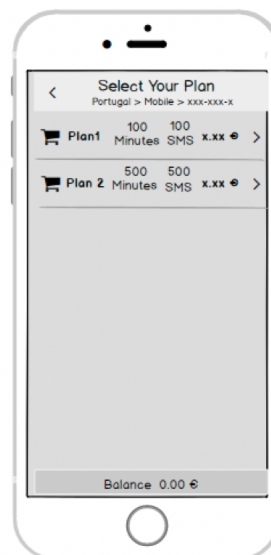
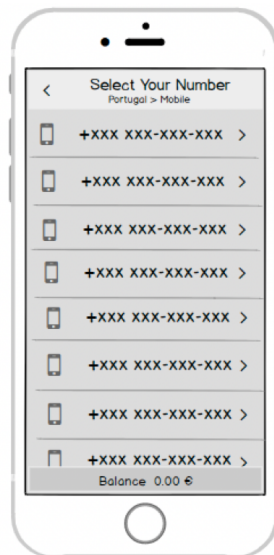
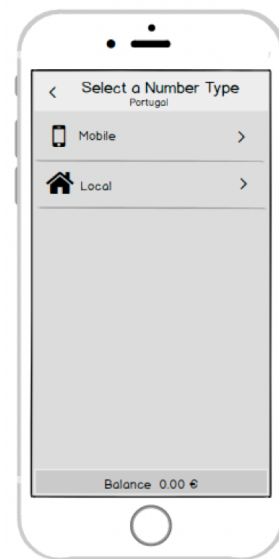
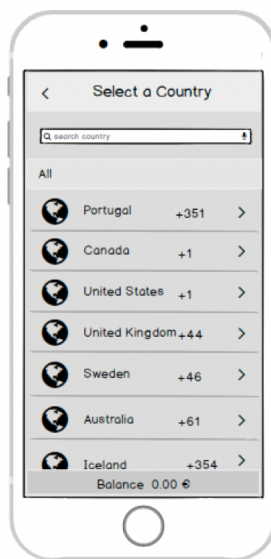
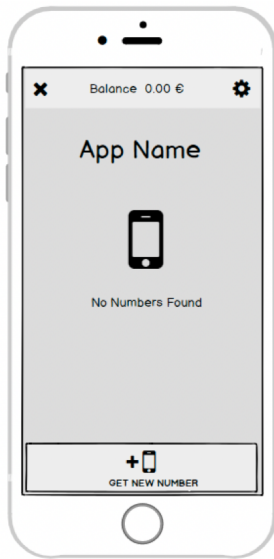
# Appendix C

## Mockups

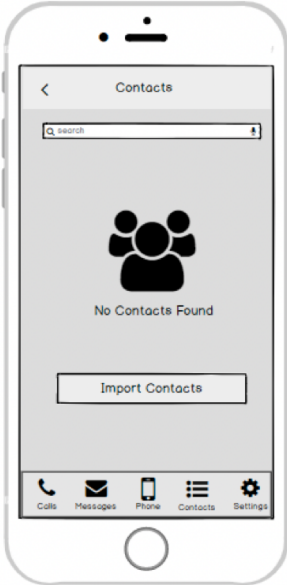
### C.1. Mobile Application Mockups

In this section are illustrated the UI's proposed for the Mobile Application by the intern.



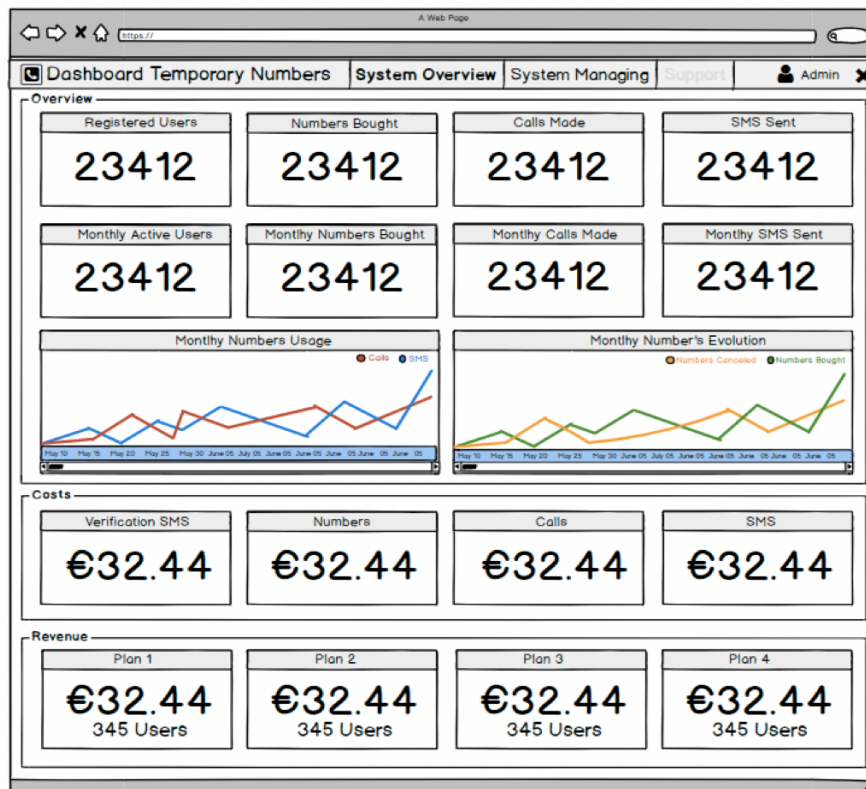
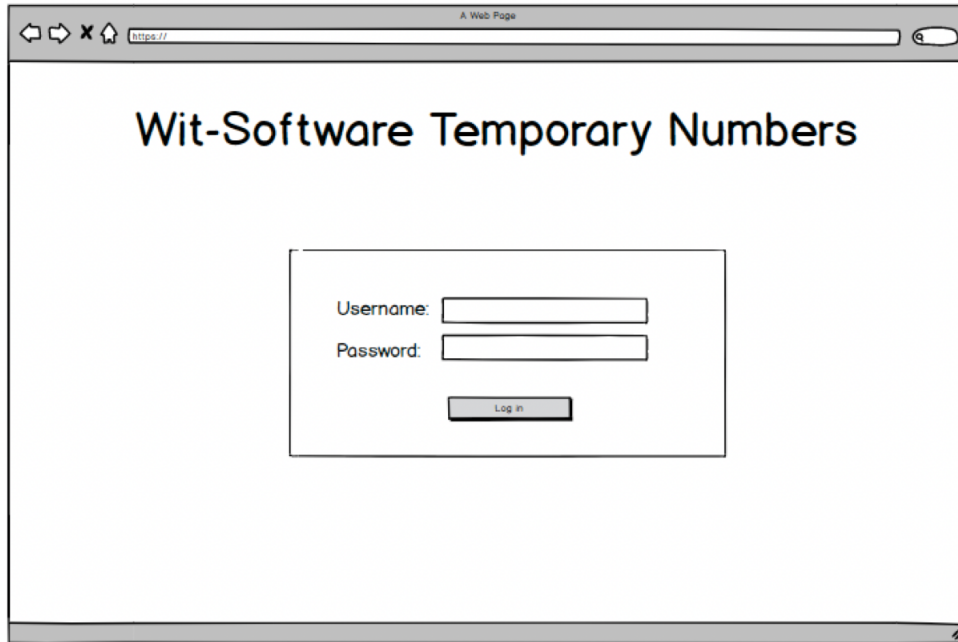






## C.2. Dashboard Mockups

In this section are illustrated the UI's proposed for the dashboard by the intern.



The screenshot shows a web browser window with a navigation bar containing 'Dashboard Temporary Numbers', 'System Overview', 'System Managing', 'Support', and 'Admin'. The main content area is divided into three sections:

- All Users:** A table with 15 rows, each representing a user named 'John David'. Each row contains a phone number (xxx-xxx-xxx), a balance (€ xx.xx), and two columns of 'Some text'. A search bar is at the top, and checkboxes are on the right.
- Alerts/Notifications:** A text area for writing alerts, with 'Send Notification' and 'Send Alert' buttons below.
- Disable Accounts:** A dropdown menu for 'Reason' (set to 'Calls to 911'), with 'Delete Account' and 'Suspend Account' buttons below.
- Add/Remove From User Balance:** An input field for 'Amount' (€xx.xx), with 'Add' and 'Remove' buttons below.

This page is intentionally left blank.

# Appendix D

## Tests

### D.1. Functional Tests

In this section are detailed the functional tests that were made for verification purposes, as stated in Chapter 9, section 9.1.

<b>Id</b>	<b>Test</b>	<b>Steps</b>	<b>Expected Result</b>	<b>Result</b>
TC01	Registration	1) Insert valid phone number 2) Insert the OTP received	User registers into the platform	PASS
TC02	Sign In	1) Insert valid phone number 2) Insert the OTP received	User logs on to the platform	PASS
TC03	Buy Phone Number	1) Choose the Country 2) Choose Number type 3) Choose a Number 4) Choose a Plan	User buys a Temporary Phone Number	PASS
TC04	Edit Phone Number Description	1) Select the desired Phone Number in the Profile Menu 2) Press Settings button for the selected Phone Number 3) Choose the Edit Number Description option 4) Insert desired Description	User changes Phone Number Description	PASS
TC05	Change Phone Number Plan	1) Select the desired Phone Number in the Profile Menu 2) Press Settings button for the selected Phone Number 3) Choose the Change Plan option 4) Select desired Plan	User changes the Temporary Phone Number's Plan	PASS
TC06	Make a Call	1) Go to Dialpad screen 2) Select Temporary Pnhone Number to make the call 3) Type in/select desired Number/Contact 4) Press Call button	User makes a Call	PASS

<b>Id</b>	<b>Test</b>	<b>Steps</b>	<b>Expected Result</b>	<b>Result</b>
TC07	Send a Message	1) Go to Messages screen 2) Press the New Message button 3) Type in/select desired Number/Contact 4) Type in the Message desired 5) Press the Send button	User sends a Message	PASS
TC08	Messages History	1) Access Message screen	User views Message History	PASS
TC09	Conversation Messages History	1) Access Message screen 2) Select desired Conversation	User views Conversation Messages History	PASS
TC10	Calls History	1) Access Calls History screen	User views Calls History	PASS
TC11	Access Contacts	1) Access Contacts screen	User views Contacts	PASS
TC12	Cancel Temporary Phone Number	1) Select the desired Phone Number in the Profile Menu 2) Press Settings button for the selected Phone Number 3) Choose the Cancel option 4) Confirm operation	User Cancels Temporary Phone Number	PASS
TC13	Cancel Account	1) Access the Profile screen 2) Access the Settings button 3) Select the Cancel Account option	User Cancels the Account	PASS
TC14	Logout	1) Access the Profile screen 2) Access the Settings button 3) Select the Logout option	User Logout	PASS