

Mestrado em Engenharia Informática 2015/2016
Estágio
Relatório Final

Software development for remote mobile protection

Bruno Filipe Baptista Marques Pedroso
bpedroso@student.dei.uc.pt

Orientador DEI:
Marco Vieira

Orientador WIT:
João Pedro Silva

Data: 1 de Setembro de 2016



FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA



FCTUC

Departamento de Engenharia Informática

Faculdade de Ciências e Tecnologia
Universidade de Coimbra
Pólo II, Pinhal de Marrocos, 3030-290 Coimbra

Tel: +351239790000 | Fax: +351239701266 |
info@dei.uc.pt



WIT Software, S.A.

Centro de Empresas de Taveiro
Estrada de Condeixa, 3045-508 Taveiro, Coimbra

Tel: +351239801030 | Fax: +351239801039 |
info@wit-software.com

Estagiário:

Bruno Filipe Baptista Marques Pedroso

bpedroso@student.dei.uc.pt

Orientador do DEI:

Marco Vieira

mvieira@dei.uc.pt

Júris:

Carlos Fonseca

cmfonsec@dei.uc.pt

Tiago Baptista

baptista@dei.uc.pt

Orientador da WIT:

João Pedro Silva

joao.silva@wit-software.com

Acknowledgements

This report describes the work done during my internship, the final step of the Masters course, and now that it is complete I need to show my gratitude to some people that made it possible.

Even though this report is related to the internship, it represents my academic years as they have come to an end. For that reason, this report is dedicated to my parents who have always been by my side, guiding and supporting me through my entire life. To them, I owe everything and have an eternal gratitude.

Secondly, I would like to thank my WIT supervisor João Pedro Silva for his support, his help and for his good judgement always caring. Also, a big thanks to Pedro Vasco Pereira, my tutor, for his guidance and patience.

Many thanks to Marco Vieira my DEI supervisor, for his availability, for being always ready to clarify and remind me of how to manage my software project properly as well as helping me to create the best report possible.

To WIT Software for giving me this opportunity and the best conditions to accomplish a great work, always being supportive.

Finally, to my girlfriend who kept pushing me to exceed myself.

Thank you

Abstract

In a world where technology evolves rapidly and is taking every day more part in our lives, it becomes difficult to live without it. The regular computers are currently less sold than mobile devices such as mobile phones and tablets, which already exist in a bigger number than people on earth.

These devices are powerful tools from where we control part of our lives. Its not just for its price or emotional value but as the phone is nowadays an entry point to every virtual account we have and all our digital information kept online, it has an important role in our daily routine. So, when we are separated from them as a result of being lost or stolen, we get so worried.

In case our phone gets missing, an imaginary countdown starts and when it stops, as the phone gets turned off or its battery dies, the chance of recovering it or securing our private information from someone who might have it, gets scarce. As the time window is small, people need a fast and effective way to communicate with their device and get information that might help them to achieve those objectives.

The objective of this internship is to create a software solution and that focus on creating features in the Device Protection area. These features will be focused on controlling devices remotely, for instance, locating the phone, locking it, wiping it and getting usage info with the main objective of allowing the users to retrieve their lost or stolen devices, or at least, to retrieve or delete the private data they carry or give access.

Keywords: Android, Mobile Device, Mobile Phone, Protection, Security, Software, Tablets, Technology, Theft

Resumo

Num mundo em que a tecnologia evolui rapidamente e se envolve cada vez mais nas nossas vidas, torna-se difícil viver sem ela. Os normais computadores já são menos vendidos do que dispositivos móveis como telemóveis e tablets que já existem em maior número do que pessoas no mundo.

Estes dispositivos são poderosas ferramentas por onde controlamos parte da nossa vida. Não é apenas pelo seu preço ou valor sentimental mas por serem também hoje em dia um ponto de acesso a todas as contas virtuais e toda a informação que guardamos online, tem um papel muito preponderante no nosso dia-a-dia. Por isso, que quando ficamos sem eles por os perdermos ou serem roubados ficamos tão preocupados.

Quando ficamos sem o telemóvel por uma destas razões, começa uma contagem decrescente que quando pára, por o telemóvel ser desligado ou ficar sem bateria, a probabilidade de o recuperarmos ou de protegermos a nossa informação privada acessível através dele fica muito reduzida. Como esta janela temporal é pequena, as pessoas precisam de uma solução rápida para comunicar com o dispositivo e obter informação que pode ser vital para atingir estes objetivos.

O objetivo deste estágio é criar um software que vise a prototipagem de um conjunto de funcionalidades na área de Proteção de Dispositivos, funcionalidades essas mais focadas no controlo remoto de dispositivos que permitam por exemplo, localizar o telemóvel, bloquear o telemóvel, limpá-lo remotamente e obter informação de utilização do mesmo, com o objetivo principal de que os utilizadores possam recuperar os seus dispositivos perdidos ou roubados, ou pelo menos, consigam recuperar ou apagar a informação que estes contêm.

Keywords: Android, Dispositivos Móveis, Roubo, Proteção, Segurança, Software, Tablets, Tecnologia, Telemóveis

Contents

Acknowledgements	iii
Chapter 1: Introduction	1
Chapter 2: State of Art	5
2.1 Features	5
2.2 Competitors	6
2.2.1 Eardroid Anti-Theft	7
2.2.2 Cerberus Anti Theft	7
2.2.3 Avast Anti Theft	8
2.2.4 Find My Phone - TrackView	8
2.2.5 Prey Anti Theft	9
2.2.6 McAfee - Security & Power Booster	9
2.2.7 Lookout Security and Antivirus	10
2.3 Comparative Analysis	11
Chapter 3: Structure and Objectives of the Updates project	13
3.1 Components	13
3.1.1 Backoffice	13
3.1.2 Server	14
3.1.3 Database	15
3.1.4 Android App	15
3.2 Security and Privacy	16
Chapter 4: Project Management	17
4.1 Methodology	17
4.2 Planning	19
4.2.1 First Semester	19
4.2.2 Second Semester	19
4.3 Risks	24
4.3.1 Risks definition	25
4.3.2 Risks description and conclusions	26
Chapter 5: Requirements	31
5.1 Requirements Analysis	31
5.2 Functional Requirements	33
5.3 Non-Functional Requirements	39
5.4 Requirement Changes	40
Chapter 6: Architecture	41
6.1 Context	42
6.1.1 Physical View	43

6.2	Containers	45
6.2.1	Web Portal	45
6.3	Server	47
6.3.1	Components	47
6.3.2	Data Storage	48
6.4	Android SDK	51
6.4.1	SDK Components	51
Chapter 7: Development and Final Product		57
7.1	Web Portal	58
7.2	Android App	66
Chapter 8: Evaluation		69
8.1	Functional Testing	69
8.2	Non-Functional Testing	72
8.2.1	Usability testing	72
8.2.2	Performance Testing	80
Chapter 9: Conclusions and Future Work		83
Bibliography		85

List of Figures

3.1	Updates High-Level Architecture	14
4.1	Gantt First Semester	20
4.2	Burn Down Chart First Semester	21
4.3	Gantt Second Semester Sprints 1-3	22
4.4	Gantt Second Semester Sprints 4-7	23
4.5	Burn Down Chart Second Semester	24
4.6	Risks Matrix - Danger Level	26
4.7	Risks Matrix	28
4.8	Risks Final Matrix	29
5.1	User Story	31
6.1	Box structure	42
6.2	Context	43
6.3	Physical View	44
6.4	Web Portal Container	46
6.5	Server Container	47
6.6	Protect Commands Table	49
6.7	File system hierarchy	51
6.8	Android Container	52
6.9	Android SDK container	53
6.10	Android Database	55
7.1	Web Portal Homepage	58
7.2	Web Portal Locate menu	59
7.3	Web Portal Lock menu	60
7.4	Web Portal Wipe Data menu	60
7.5	Web Portal Wipe Factory menu	61
7.6	Web Portal Calls menu	61
7.7	Web Portal Messages menu	62
7.8	Web Portal Browser menu	62
7.9	Web Portal Gallery menu	63
7.10	Web Portal Camera menu	63
7.11	Web Portal Network menu	64
7.12	Calls menu content mobile version	65
7.13	Mobile version image zoom	65
7.14	Mobile version responsiveness	65
7.15	Welcome screen	66
7.16	Registration screen	66
7.17	Confirmation screen	67
7.18	Admin Permission requesting	67

8.1	Actions and time taken for Use Case 1	74
8.2	Actions and time taken for Use Case 2	75
8.3	Actions and time taken for Use Case 3	75
8.4	Actions and time taken for Use Case 4	76
8.5	Actions and time taken for Use Case 5	77
8.6	Actions and time taken for Use Case 6	78
8.7	Actions and time taken for Use Case 7	79

List of Tables

2.1	Competitor - Eardroid Anti-Theft [12]	7
2.2	Competitor - Cerberus Anti Theft [14]	7
2.3	Competitor - Avast Anti Theft [16]	8
2.4	Competitor - Find My Phone - TrackView [18]	8
2.5	Competitor - Prey Anti Theft[20]	9
2.6	Competitor - McAfee Security & Power Booster [22]	9
2.7	Competitor - Lookout Security and Antivirus [24]	10
2.8	Competitors comparison	11
4.1	Risk 01	26
4.2	Risk 02	26
4.3	Risk 03	27
4.4	Risk 04	27
4.5	Risk 05	28
5.1	User Stories	33
5.2	Functional Requirements	38
5.3	Non Functional Requirements	39
8.1	Functional Testing Results	71
8.2	Non Functional Requirements	81
8.3	Performance Times per action	81

Acronyms

ACID Atomicity, Consistency, Isolation, Durability

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

App Application

BLOB Binary Large Object

DB Database

FR Functional Requirement

GCM Google Cloud Messaging

GPS Global Positioning System

HTTPS Hyper Text Transfer Protocol Secure

ID Identification

IoC Inversion of Control

IT Information Technology

IMEI International Mobile Station Equipment Identity

JSP JavaServer Pages

MMS Multimedia Messaging Service

MSISDN Mobile Station International Subscriber Directory Number

MVC Model–view–controller

NFR Non Functional Requirement

NTFS New Technology File System

OpCo Operating Company

POJO Plain Old Java Object

REST Representational State Transfer

SDK Software Development Kit

SIM Subscriber Identity Module

SMS Short Message Service

UI User Interface

XML Extensible Markup Language

XMPP Extensible Messaging and Presence Protocol

Chapter 1

Introduction

Nowadays people's mobile phone or tablet is not just a piece of technology that they can detach to. Long are the days where phones were the heaviest object people carried around and the only thing they could do with them was make calls. Now, make calls is just another feature amongst many. A phone gives help organizing people's lives, organizing their work, having fun, connecting with family, friends and the whole world. Almost all their's private info is there or can be accessed through it which makes the phone an essential part of their lives. As it can be a great strength, it can also be a vulnerability when not in their possession.

When a phone gets missing it can be due to two reasons: either it was lost and there is still a chance of someone with good intentions to find it and return it, or it gets missing because someone stole it. If that happens, certainly, that person has malicious intentions. Either way, the most common approach to solve this problem, is to report the case to the police and wait for good news. However, such approach involves waiting and hoping for the best and as time goes by, the chances of having success decreases. Waiting is a painful process and brings anxiety to people as they are powerless to do something to recover their phone. This problem still has no optimal solution found, and certainly should get more attention.

This internship took place at WIT Software, S.A., "a software company that creates advanced solutions and white-label products for the mobile telecommunications industry" [2]. The software to be developed was integrated in the Updates project, a solution that manages and monitors remotely mobile applications.

WIT Software, S.A. developed a solution which purpose is to manage and monitor mobile devices remotely. This solution called "Updates" performs management actions on the device: when its operating company, Vodafone, has a new Android app to announce to its clients, Updates communicates with the device and through its app, the device owner can choose to install the new app, update that app or other apps, and finally to delete them. Also, info related actions can be performed such as showing advertisements selected by Vodafone and retrieving usage info from the device. All the work done by the author had the objective of being integrated in the Updates solution, complementing it and making it a more complete product to the end users.

As the Updates project is a device management solution, it performs several actions related to managing the device and it had a perfect structure to be the foundation of a solution which goals would be to solve the problem of people getting back their missing phones. This solution, meant to be a device protection proof of concept product lead to

the creation of this internship, with the objective of developing a solution to help people to find their lost or stolen device or at least to protect their meaningful private data accessed through it.

The present report reflects the work done by the author at WIT Software, S.A. during the annual internship included in the Master's degree in Informatics Engineering in the Department of Informatics Engineering of the University of Coimbra. This work was supervised by João Silva, Senior Software Engineer at WIT Software, S.A. and Marco Vieira, PhD Professor at Department of Informatics Engineering of the University of Coimbra.

This project has two main goals, which are interconnected:

- Personal - Improvement of the intern's skills during the internship;
- Professional - Finish the internship with a satisfying result for the intern and for the company.

As this internship is the last step of the intern's academic career and the first real-world project that he works on, it is expected to consolidate what he has learnt during the previous years attending Bachelor's and Master's course at University of Coimbra, and to gain a lot of knowledge, improving personal and professional skills.

Keeping in mind the problem stated, this internship's proposal is to create a solution to protect people's mobile devices and data. This solution is divided into three areas:

- Web Portal – This represents the creation of a website from scratch, creating the front-end, business logic and all data structures needed to give people the possibility of requesting this solution to take actions on their missing device. So, when someone for example, loses their device, all actions that the person can take are triggered through this web application which afterwards shows the action's result;
- Server – As the solution will be integrated in the Updates, a base server already exists. This server already communicates with the devices and to upgrade it in order to meet this internship objectives, new communication systems with the Web Portal and the devices will be built, as well as multiple services, data structures and many new logic modules. All this, to support new features and new communications;
- Android – If people want to get information from their devices, an app must exist to gather it. As the target client OS of the internship was Android, there will be created a SDK which will be embedded in a new android prototype application. This SDK will execute the requested actions in the device and return their results.

Building these two new systems, the Web Portal and the Android SDK, and improving the existing one, the Server, a new device protection solution will be created that later in time can be added to the existing Updates product and offer to the final user a easy to use way of solving the previously stated problem. During this whole process, a lot of knowledge is expected to be gained in multiple areas.

In addition to the personal gain, the second goal is to complete successfully this project, creating an excellent software that meets WIT Software, S.A. and its clients expectations. In order to accomplish that, not only all the requirements must be achieved, concerning about quantity of work done and its quality, practicing good programming habits that increase the product value not only to the costumer but also to the project team and company.

This document is divided into nine chapters, including this Introduction and the following:

- **State of the Art** – This section begins with the description of all features to be implemented and follows with the presentation of an analysis of the competitors to this project alongside a comparison between them;
- **Structure and Objectives of the Updates project** - In order to fully understand the project and what will be done, some background information must be understood. This chapter presents an explanation on the Updates project, stating what exists and what will be developed by the author to create the final solution named Updates Protect;
- **Project Management** – This section presents project management information of the approach taken, such as the software development methodology, time planning and risks;
- **Requirements** - Chapter dedicated to the requirements of the project;
- **Architecture** – This section presents different views of the multiple components and interactions between them;
- **Development and Final Product** - After all the development is finished, a retrospective should be done to analyze which were the main problems, and to present the final product;
- **Evaluation** - To prove the solution effectiveness and quality, it must be evaluated. This chapter shows which processes were designed and performed to guarantee the final solution's quality;
- **Conclusions and Future Work** – As the final chapter, it is presented a retrospective of the past year and what could be done in the future.

As some sections of this report are shortened due to its size, some appendices are available in the end of the report to provide a full explanation on some topics. The **appendices** are:

- **Architecture**
This document shows a detailed description of all diagrams of the project components and communication between them;
- **Project Management**
This document corresponds to the specification of all existing risks, how harmful are they to the project and their changes over time. Also, it contains information regarding the development phase of the second semester;
- **Requirements**
Document that intends to present a description of the functional requirements, complementing its respective section.
- **Development**
Appendix which purpose is to present details about the implementation of the whole solution;
- **Evaluation**
Specific information regarding the testing phase of the solution.

Chapter 2

State of Art

This chapter intends to present an analysis of the android applications available in the market that compete with the features identified to be developed during this internship. It is an important step to make such an analysis as it shows the stakeholders many indicators on the project like its viability or how much of a innovation it represents. This is a crucial step in any project as this market study can reveal if there are any solutions available to the public and if so, if this solution has any chance of success. Otherwise, it may not be reasonable to be developed.

The chapter is divided into three sections: the first presents the features defined for this project and the second which are the competitors and a comparison between them.

2.1 Features

In the interest of analyzing the possible competitors of this project, we must define beforehand which are the features the author proposes to achieve so that we can compare afterwards the completeness of the final product with possibly existing solution already available on the market. These features were settled by the stakeholders of the project and the author had no participation on the research or definition of which features were the best to this project.

- Locate - This feature is one of the most important and relates to the capacity of retrieving the device's GPS location;
- Lock - The Lock feature represents the capability of remotely locking the screen of the device;
- Wipe - Produces a deletion of files on the device. Can be done by two ways:
 - Data Wipe - The user wants to delete specific data from his device and selects the files or applications he wants to delete and orders it;
 - Factory Reset - The user wants to delete everything from the device, also known as Factory Reset, the act of restoring a electronic device to its original system state by erasing all of the information stored on the device in an attempt to restore the device's software to its original manufacturer settings. [3]
- Spy - Group of features which give the owner information regarding what someone might be doing in their phone. These features can be:
 - Get Calls - Feature representative of retrieving the list of last incoming and outgoing calls on the device;

- Get SMS - Feature representative of retrieving the list of last incoming and outgoing text messages on the device;
- Get MMS - As the previous one, this feature gets the list of multimedia messages made and received by the device;
- Get Media - Also referring to multimedia, this feature represents the ability of getting the images and videos made by the device;
- Get Browser History - Ability of obtaining the visited webpages through the device;
- Take Pictures - Capacity of taking pictures through frontal and back camera of the device, in case the cameras exist;
- Film Videos - Capacity of filming videos through frontal and back camera of the device, in case the cameras exist;
- Get Video Stream - When requested, this feature creates a live transmission of audio and video from the device microphone and cameras to the browser;
- Get Device's Info - Capacity of retrieving the device information such as the device identifier number or SIM card info;
- Get Wi-Fi's list - Feature representative of retrieving the available Wi-Fi's at the moment of request;
- Start Wi-Fi and Data - Although it is not a feature directly available to the user, the system must have the capacity of turning on the device's Wi-Fi or Data Plan connector in case the device is not connected to the internet.

2.2 Competitors

In this section will be presented and detailed some existing applications that compete with the application developed during this internship. As the solution includes many features and for each one of them there are multiple applications available on the market that include those features, competitors must be analyzed through different perspectives.

In order to analyze and compare the identified applications, a set of properties was defined for the competitors:

- Relevant features included
- How the information is presented
- App is free, paid, has in-app billing
- Completion of information retrieved

2.2.1 Eardroid Anti-Theft


	Play Store Rating	4.3/5
	Play Store Installs	10.000 – 50.000
	Play Store Reviews	400
	Paid	Yes, with free trial
	Main Focus	Anti-Theft
	Remote Control Type	Web Application

Table 2.1: Competitor - Eardroid Anti-Theft [12]

Eardroid Anti-Theft as the name mentions is an application which purpose is to remotely monitor data in a mobile device to retrieve info on who has the device at some moment. This app only supported on Android devices has few Play Store installs in comparison with other solutions but it has a lot to offer. Its web application includes a simple dashboard to use its features, which are very similar to the ones of this project. Eardroid allows to see different kinds of information retrieved from the person's device such as device related info, call and sms history, pictures and videos, GPS and so on. An interesting feature is the ability to control de device remotely by SMS.

2.2.2 Cerberus Anti Theft

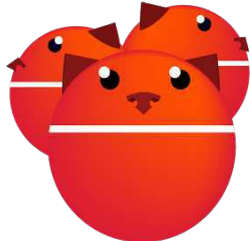
	Play Store Rating	4.4/5
	Play Store Installs	1.000.000 – 5.000.000
	Play Store Reviews	80.000
	Paid	Yes, with free trial
	Main Focus	Anti-Theft
	Remote Control Type	Web Application

Table 2.2: Competitor - Cerberus Anti Theft [14]

Cerberus is an app made by LSDroid and one of the most complete on the market as an anti-theft solution. Provides a major number of the identified features and presents useful information about the device and user actions on the website. Their ability to retrieve very complete information regarding the features in sight, is complemented with other characteristics that make user experience with this app very satisfying. In addition to the retrieval of the requested info, they present a list of all commands requested by the user, their status, if the system is currently working in any of those commands, the device's battery percentage, and so on.

2.2.3 Avast Anti Theft


	Play Store Rating	4.2/5
	Play Store Installs	10.000.000 – 50.000.000
	Play Store Reviews	210.000
	Paid	Yes, with free trial
	Main Focus	Anti-Theft
	Remote Control Type	Web Application

Table 2.3: Competitor - Avast Anti Theft [16]

Avast Anti Theft is an app from Avast, security Software Company with worldwide known products, with a 21.4% market share for antivirus applications on January 2015[4]. It is a good solution considering our goal, including the basic features identified (locate, lock and wipe) and some others such as the device's info, audio, private data retrieval. Allows to access info on the device related to private messages, calls, access accounts and accessed web pages. Despite not being the most complete app in this analysis, probably due to its company success, it's a very used app.

2.2.4 Find My Phone - TrackView


	Play Store Rating	4.1/5
	Play Store Installs	1.000.000 – 5.000.000
	Play Store Reviews	13.000
	Paid	No
	Main Focus	Anti-Theft
	Remote Control Type	Desktop Application

Table 2.4: Competitor - Find My Phone - TrackView [18]

Find My Phone Antitheft also known as TrackView, is an application made by Cybrook Inc., a high-tech company located in Silicon Valley that develops mobile security products and services [5]. Their product is a cross-platform that offers us the possibility of taking pictures/film videos and live stream from both cameras of a mobile device or a computer. Runs on Android/iOS devices and also provides an app for Windows. When it comes to the basic features, this app only has the ability of getting the device's location. However, it has many strengths related to the audio/video features as they offer the possibility of creating audio and video streaming, detect and record events, great control over front and rear cameras, and some others.

2.2.5 Prey Anti Theft


	Play Store Rating	4.2/5
	Play Store Installs	1.000.000 – 5.000.000
	Play Store Reviews	52.000
	Paid	No
	Main Focus	Anti-Theft
	Remote Control Type	Web Application

Table 2.5: Competitor - Prey Anti Theft[20]

Prey is a theft protection software that provides the basic features in sight. It has a simple and friendly interface from where the user can mark his device as stolen or missing. From it, the device owner can take pictures and retrieve a complete list of the device's info, the network it's connected to and the available Wi-Fi's. Once the owner set the device as lost or stolen, the app continuously retrieves useful reports, providing continuous support for the user until the device is set to found, feature that proved to be very interesting.

2.2.6 McAfee - Security & Power Booster


	Play Store Rating	4.3/5
	Play Store Installs	10.000.000 – 50.000.000
	Play Store Reviews	350.000
	Paid	No, in-app billing
	Main Focus	Antivirus/Security
	Remote Control Type	Web Application

Table 2.6: Competitor - McAfee Security & Power Booster [22]

McAfee, a company owned by Intel Security Group, is known worldwide for its security products having a 6.2% market share on the antivirus applications[4]. The product they present offers the three basic features and some extra ones: get call logs, text messages logs, new media and takes pictures (after the request is sent, the pictures is taken only when someone touches the screen). Although it does not provide many features, it offers the chance of contacting the device not only through the internet, but also by SMS, a great alternative to situations where establishing an internet connection is not possible.

2.2.7 Lookout Security and Antivirus


	Play Store Rating	4.4/5
	Play Store Installs	100,000,000 - 500,000,000
	Play Store Reviews	841,924
	Paid	No, in-app billing
	Main Focus	Antivirus/Security
	Remote Control Type	Web Application

Table 2.7: Competitor - Lookout Security and Antivirus [24]

Lookout is a mobile security company located in San Francisco, USA, that attracted many investors over the last years and has more than 70 million users[6]. This app is a powerful tool to protect a mobile device, offering many features to achieve their goals. They cover both security and antivirus.

Regarding their security features, Lookout provides the basic ones like Locate, Lock and Wipe, but also some additional ones such as retrieving new media and taking pictures. In addition, other interesting capabilities are offered: this app retrieves the GPS location when the battery is low without direct action from the user and retrieves both GPS location and a frontal picture when the device is turned off, also without the user requesting it.

2.3 Comparative Analysis

In order to compare the identified apps with the features, three metrics were chosen: features that are present in the application are colored as green, features partially available are colored as yellow and features colored as red, are not available in the app.

Feature	WIT Solution	Eardroid Anti-Theft	Cerberus Anti Theft	Avast Anti-Theft	Find My Phone - Trackview	Prey	McAfee Antivirus & Security	Lookout Security and Antivirus
Locate Device	Green	Green	Green	Green	Green	Green	Green	Green
Lock Device	Green	Green	Green	Green	Red	Green	Green	Green
Wipe Data	Green	Green	Green	Green	Red	Green	Green	Green
Get new call logs	Green	Green	Green	Green	Red	Red	Green	Green
Get new SMS logs	Green	Green	Green	Green	Red	Red	Green	Red
Get new MMS logs	Green	Red	Red	Red	Red	Red	Red	Red
Get new Media	Green	Red	Red	Red	Red	Red	Green	Yellow
Get new browser history	Green	Green	Green	Green	Red	Red	Red	Red
Take pictures	Green	Green	Green	Green	Green	Green	Yellow	Yellow
Film video	Green	Green	Green	Red	Green	Red	Red	Red
Get video stream	Green	Red	Red	Red	Green	Red	Red	Red
Get device's info	Green	Green	Green	Green	Yellow	Green	Green	Green
Get available Wi-Fi list	Green	Red	Green	Red	Red	Green	Red	Red
Start Wi-Fi	Green	Green	Red	Red	Red	Red	Red	Red
Start Data	Green	Green	Red	Red	Red	Red	Red	Red

Table 2.8: Competitors comparison

After analyzing all the apps and their final comparison, some conclusions may be taken about this market:

- **Security is in first place** – these apps that care about mobile security, when it comes to theft all include the basic features Locate, Lock and Wipe. These are the main features and are used as a banner to mobile security.
- **Cameras are everything** – Aside from the main features, the single one that all apps have in common is the ability of taking pictures. This feature is a crucial one, as is an easy way of identifying the possible thief. However the methods used by some apps to take the pictures are a bit limited: some only have the possibility of taking pictures when the Lock option is triggered, others when a general report is requested. This means that even though the feature is available, in some apps it is not given freedom to the user to decide when to take the picture from which camera to do so without involving other unwanted or unnecessary processes.
- **Is the lockdown the best solution?** – All apps seem to agree that locking down and wiping the device is the best way to secure the personal data and getting the device's location. In fact, getting the geographic location of the device is a great help in the attempt to retrieve it. However, if the thief lives in a twenty floors building in a residential area and the precision of the location is inaccurate, the chances of retrieving the device drop to zero. Only few apps realize that spying the device and what is around it is the best choice, so features like getting the sent messages, getting the surrounding Wi-Fi's list and streaming from the device's cameras are rare to find.
- **Internet trust** – This sort of solutions rely on principle that the device will be connected to the internet. However, as it is known, Wi-Fi and Data features, when are on are great battery consumers, which may lead the device's owners to keep them turned off. If this happens, the app becomes powerless unless it has a backup system. Few of these apps actually have a backup system, which is contacting the devices through SMS, avoiding this flaw of internet dependency.

Chapter 3

Structure and Objectives of the Updates project

In this section will be described the existing project - Updates - and which are its characteristics, in order to differentiate what already exists and what has to be implemented by the author.

Updates is a large-size project developed and commercialized to the group Vodafone. It's goal is to provide a way to the company to command or suggest the installation of some Android applications on a set of devices, while retrieving info and presenting campaigns. To do so, Updates makes use of its four components:

- Backoffice - Web application from where administrators can send new commands to their clients' devices.
- Server - Server which controls all the data flows to and from the devices.
- Database - Oracle databases where commands regarding customers and related information is recorded.
- Android App - Application installed in the customer devices which receives and presents new information to the final user.

Despite all components being available to the author to develop his work, some were more relevant than others as it will be explained in the following section. Figure 3.1 shows the high level architecture of the Updates including these components that will be described in the following section.

3.1 Components

This section will describe how relevant they are to the internship and which changes the author had to do to each one of them.

3.1.1 Backoffice

The Backoffice, web application used to control information on this project, is possibly the least important component to this internship. It's only purposes to the author were viewing information existing in the databases and using some technical functions available through it, such as encryption algorithms.

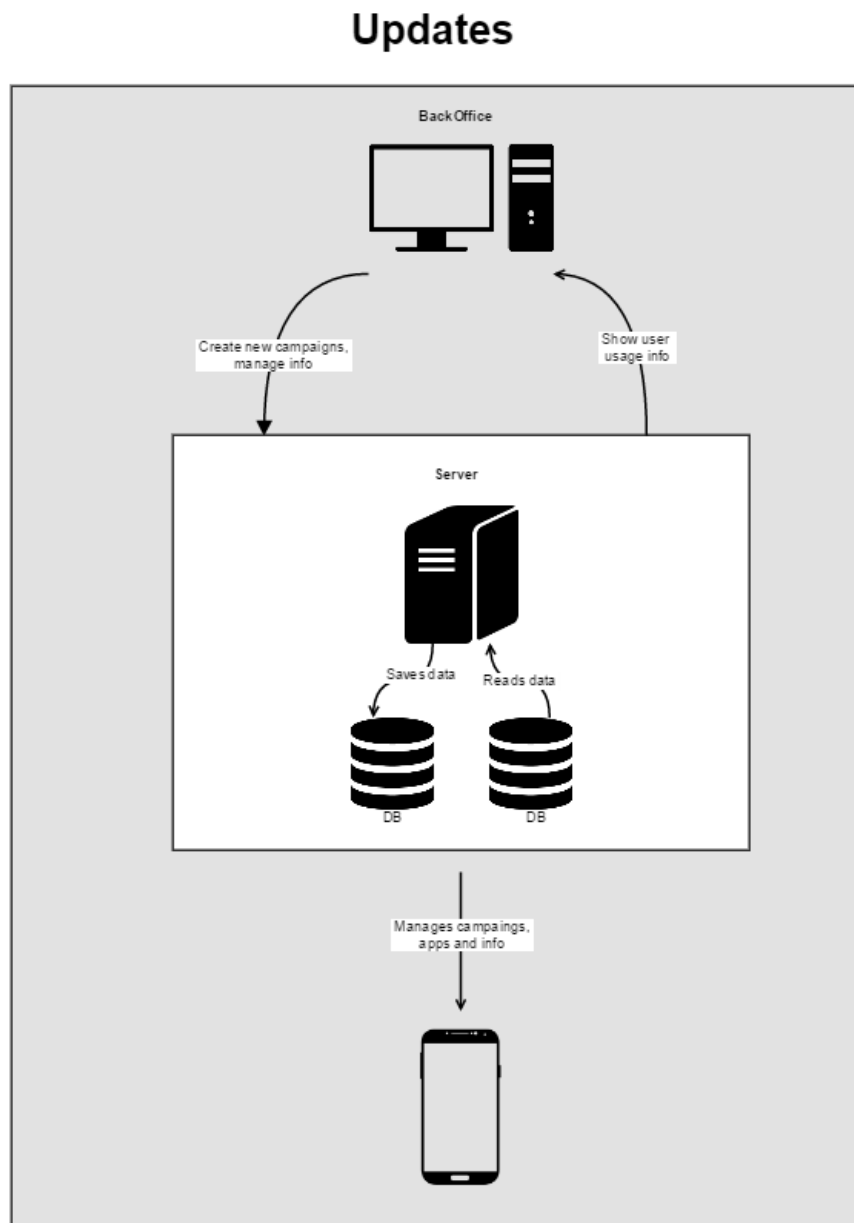


Figure 3.1: Updates High-Level Architecture

3.1.2 Server

The Server is the most important component as it was used by the author to implement the new features and manage all information. It is formed by some smaller projects and its complexity is considerable.

Some components of this server are of the author's interest. Firstly, its connection to the databases as they are necessary to access the information. Secondly, the communication process used by Vodafone, the Push System which contacts the Android device through a Nudge SMS, a silent text message sent to the device that is not perceived by the device's owner and this way, communication server-device can be performed without an internet connection.

In addition to these available already implemented functionalities, it is the author's responsibility to add:

- New server-device communication methods related to the internship. Alongside

with the already existing communication system a new one has to be implemented as an alternative. The chosen one was Google Cloud Messaging;

- A new database access layer;
- New data management layer;
- A REST interface to communicate with the Web Portal to be developed by the author;
- All methods necessary to control the layers mentioned above and to create the goal features of this internship.

3.1.3 Database

This component is not only one Oracle database, but several ones. Firstly, there is a major database related to the whole project and containing its the general information and settings. Then, there are a set of databases related to each OpCo. Regarding this internship, the author was given access to the general database and the one related to the portuguese OpCo, the one on which it was expected to work on.

As all databases were functional and suitable to this internship, it was the author responsibility to create new tables and scripts in order to accomplish his goals, storing the user's info.

3.1.4 Android App

The last component is the Updates Android app. This app executes all the Updates features on the device and communicates to the server. This last component had few interest to the author, as its purpose is different from the internship's goals and its overall structure was useless. However, some particular parts were of interest, which are enumerated ahead.

As this app is useless to the internship, a new one had to be created to store the SDK which would execute the features proposed. Considering the Android apps' structure, a SDK can be seen as a library that executes certain functionalities and is independent from the rest of the app. This way, in order to create the SDK, a base Android app must exist with the only purpose to store the SDK. Once it is completely developed, the SDK can then be integrated in the Updates app, simply by being imported to the project. For this reason, the objective is to develop an Android SDK and not an Android app.

Concluding, regarding this section, it's the author responsibility to:

- Create a new dummy android app with system privileges so all features can be executed successfully;
- Create an Android SDK that executes the features proposed to this internship;
- Create a communication system with the server's existing structure and with the Google Cloud Messaging servers. Regarding the communication with the server, some implementation methods of the Updates app were used;
- Persist all requests' information on the device until executed.

3.2 Security and Privacy

Security is by nature a top priority in informatics and information systems and this case is no exception. The features that are proposed to develop during this internship involve managing, transferring and saving users private data, very sensitive data. As all developments to be made by the author will be joined into the Updates, they will benefit of several security mechanisms granted by the Updates infrastructure which protects data confidentiality and integrity. These mechanisms include data encryption during communications, data encryption on storage, servers and databases protected by several layers of load balancers and much more. All together provide all necessary security to such sensitive material.

Even if the developed systems benefit from the Updates security, the creation of the web portal involve a new communication with the server. As this communication is used to authenticate the final users and transfer their private data, additional security is needed. In order to assure that, the user authentication is performed through Basic Authentication over HTTPS, creating a secure channel to send the credentials. After that, only a identification cookie is sent in every request. Just as the authentication, all requests are done over HTTPS. As stated before, the privacy of user personal information is a very important matter and when it comes to this project, many privacy concerns may arise. The project intents to transfer and manage private data between several points and for such behaviour there are regulating international laws which for instance, enforce systems to ask for users consent. However, this problematic is out of scope of the internship and the author does not have to deal with them.

Chapter 4

Project Management

For the sake of any project, software related or not, planning must be done in order to be successful. Otherwise, only after the roof is built, the workers find out that a building cannot be built from a top-down approach. In software development the problem is the same, therefore people can't start coding from the very moment the project begins. There must be a previous analysis of the project and the variables that affect its course so the team can have a clearer view of them. In software engineering there are processes which intend to define structured sequences of stages to develop the intended product. These process models are called Software Development Life Cycles and help software teams to develop products efficiently. So as previously said, a previous analysis must be done to understand which life cycle the team benefits the most in using. This chapter presents the analysis done, which life cycle was picked and why.

4.1 Methodology

When it comes to choosing the best methodology, firstly it must be taken into account the software development phases:

- Requirements identification and analysis
- Architecture and design
- Implementation
- Testing
- Deployment
- Maintenance

Considering this internship, these two last phases are irrelevant due to not being done. Besides these phases, other factors influence the life cycle decision such as the team members' geographic location, how well the team members understand each other's work, the project lifetime, the type of client and some more factors. After an analysis, some important characteristics of this project were found:

- Settled requirements – the defined requirements for this internship were established in the beginning and no major changes would be done in the future.
- Team proximity – team members were very close to each other which facilitated the contact between them.

- Three week sprints – It was defined that the author should do developing sprints of three weeks during the second semester.
- System integration – To check a feature as completed, all three main components must be working: Android App, Server and Web Portal.

The methodology chosen for this internship was Lean with three-week sprints. Lean is an agile methodology with seven principles that make it unique:

1. Eliminate waste
2. Amplify learning
3. Decide as late as possible
4. Deliver as fast as possible
5. Empower the team
6. Build quality in
7. See the whole

This methodology supports the idea that everything that is not adding value to a project must be eliminated. This waste applies not only to the basic time wasting activities but also to a lot other software development tasks such as useless meetings, documentation and so on. The point is to focus on what should be developed in the present because the future is uncertain and so are the requirements. For instance, a change of the requirements can lead to refactoring, over-production and defects which is all considered waste and keep the team to achieve important goals like delivering fast and with quality. Regarding a human vision of this methodology, it is supported that people must be respected and trusted. Software development is a never-ending cycle of learning, so if people have everything they need to be efficient, they must be trusted. Also, if knowledge is increasing over time, commitment should be done as late as possible. Better solutions can come up, requirements can change and they all lead to problems and rework. So, when it comes to irreversible decisions, as long as the project is not compromised, commitment should be deferred.

This methodology prove itself worthy: it gave the author freedom to take his decisions wisely even if to do so he had to defer them. This way, later on he will always have more knowledge on the problematic and his view on how to solve some key decisions may be different but better. This situation was noticeable during the first semester: once the development period started, the first stated risk started to take effect on the author as the complexity of the Updates project was so big that the definition of the data and communication between the server and the web portal was hard to define. With this problem to solve, the author with the power to decide as late as possible, decided to change the planning previously done, developing firstly the web portal and then the server, not compromising the specified deadlines and delivering in time. During that stage of web portal implementation, he kept amplifying his knowledge of the server and on how to solve such problem. Also, as the team members are very close to each other, asking questions was a much easier than scheduling a meeting and sending inefficient emails continuously.

4.2 Planning

After having the methodology, state of art and requirements well-defined, it was time to create a plan for the rest of the internship. This first plan reflected as a Gantt diagram presented in the following two sections, intents to show a realistic temporal plan of the project and to support the author during the internship, giving him an idea of being late or ahead in time compared to the initial plan.

The process to create such plan was made in two steps: firstly, the author gathered all tasks to be done and gave them a complexity grade and an estimation of how long in his opinion it would take to be completed. As soon this task was done, the second step started: the author presented the created diagram and discussed with his WIT supervisor the resulting diagram. After this discussion, the author revised the plan created, building a second version of it, a more realistic one.

4.2.1 First Semester

Looking back at the first semester, visible in figure 4.1, the month of October was spent gathering the state of art, the requirements and designing the project architecture. This month was spent studying the problem, investigating about the project, the features in sight, and questioning stakeholders.

The month of November marked the beginning of coding with a previous week spent analyzing the Updates Server. As the diagram shows, the prediction was to spend from that date to mid-January implementing the base implementation, representative of all changes needed to perform in the database, Server and creating the Web Portal. However, after finishing the database changes's cycle and spending some time analyzing the server, it was the author's opinion that the order of this base implementation was not proper, so the task Server changed with the Web Portal.

By the time the semester was over, halfway through January, compared to the diagram made in the beginning of the semester, it was concluded that the project was around two weeks behind schedule, mostly due to the planning change mentioned before. As this difference is relatively short compared to the duration of the project, it was believed that such delay could be recovered in the following months.

Figure 4.2 shows the Burn Down Chart relative to the first semester, from the first to the last week. This type of chart represents the work done on some task with the time left to perform it, being very useful to compare the work done on some point with the work that was supposed to do, predicted in a previous state. It is very useful as it gives a perception of a project being late or ahead in time, an important sign for any project. In methodologies that need that indicator very often because the development cycles are shorter, this guidance takes more relevance. For this reason Burn Down Charts are so used in the Scrum methodology to track a sprint state. Despite that, this chart displays an overview of work done in time which helped the author to realise the project progress, which supported the idea of being late but not enough to be a great concern.

4.2.2 Second Semester

As the second semester had more tasks to be performed, the Gantt diagram is divided into two figures, figure 4.3 and figure 4.4. During this semester it was expected to be developed the Android SDK and all its features alongside some improvements that might be made to the server and Web Portal, sending the required information to the server so it can be dealt with. As in the end of the first semester there was a small delay of the

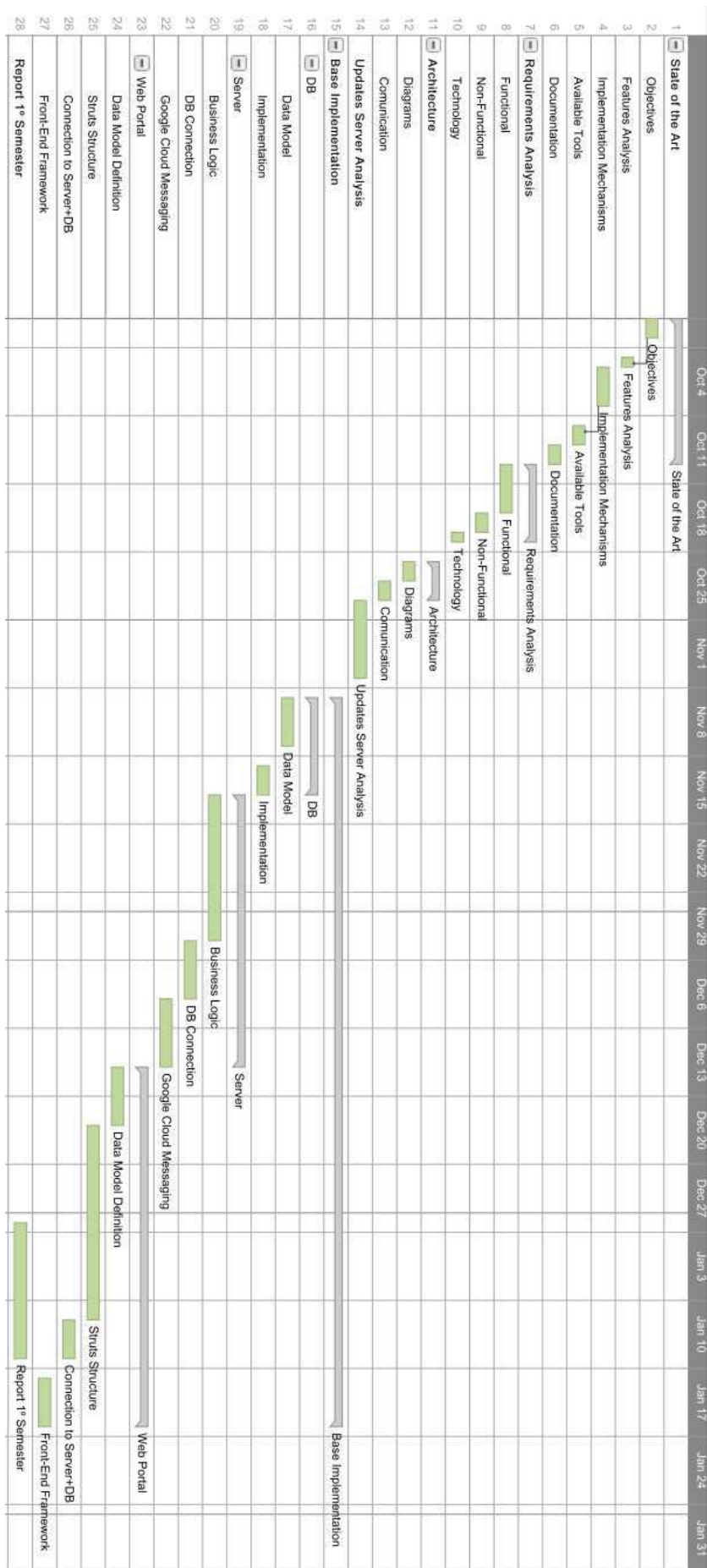


Figure 4.1: Gantt First Semester

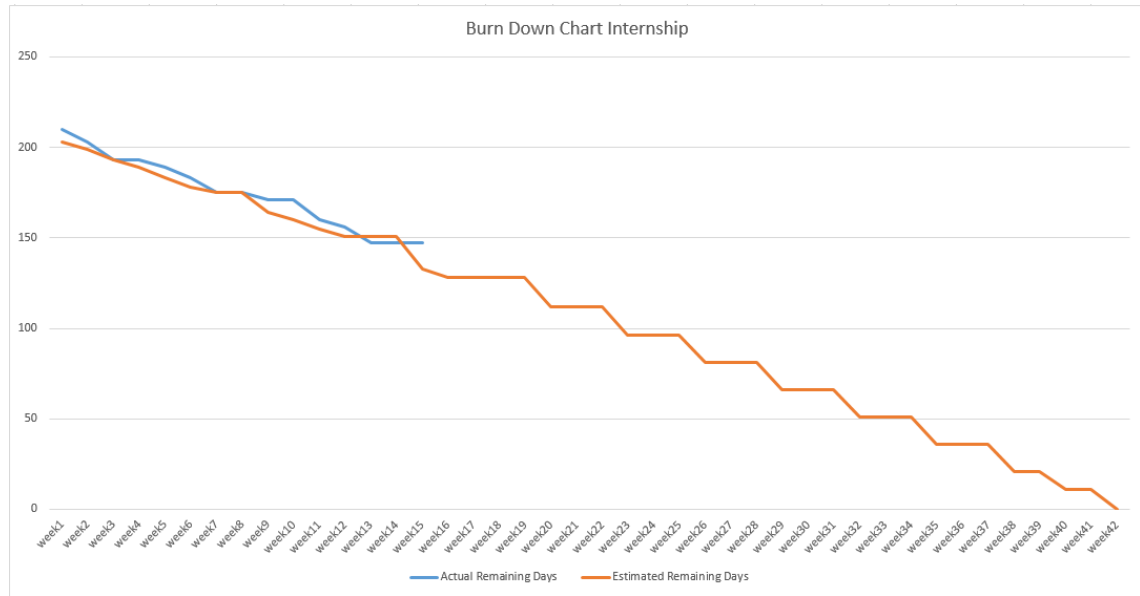


Figure 4.2: Burn Down Chart First Semester

work done, in the beginning of the second semester, some time was spent recovering such difference. In terms of tasks, this means that the task "Google Cloud Messaging" and Business Logic" were the ones finished before the android development started. For that reason, the semester was started finishing the base implementation and the first sprint still had some tasks from it.

The semester was divided into seven sprints being the first six development mixed with some testing, and the final sprint being entirely dedicated to testing. The planning for each of these sprints was decided by the stakeholders to be discussed and defined in the first day of each sprint. This way, it was easier to analyse the progress of the development so far, and a more realistic plan for the following sprint could be done as the complexity of some tasks were considerable and there were some doubts regarding if it was possible to develop them all.

As stated, the second semester started by finishing the base implementation development. This task took the rest of the month of January and up to the middle of February. Only then the Android development started by creating the base for the target app. The first sprint ended approximately at the same time as February and at this point, the app was created and already communicating with the Updates server. During the month of March took place the second and part of the third sprint. The first, represented the finishing of the base structure of the android app and already some features being developed - the capability of locate and get the available Wi-Fi networks. The second one, was already spent focusing only on the development of the designated features - Lock, Wipe, Made/Received calls and Sent/Received SMS, and ended on the beginning of April.

During the rest of the month, the author was on sprint 4, developing three more features - Retrieval of internet history, the MMS and the device gallery. After that, it was created a the structure to receive the multimedia files and save them. Once this sprint was finished and the team members met, it was realised that there would be no time to finish all features in sight. So, the next two sprints, the last ones of development were spent doing the features related to the camera and a few more related to the user sign up process, and his account management. The seventh sprint that started in the beginning

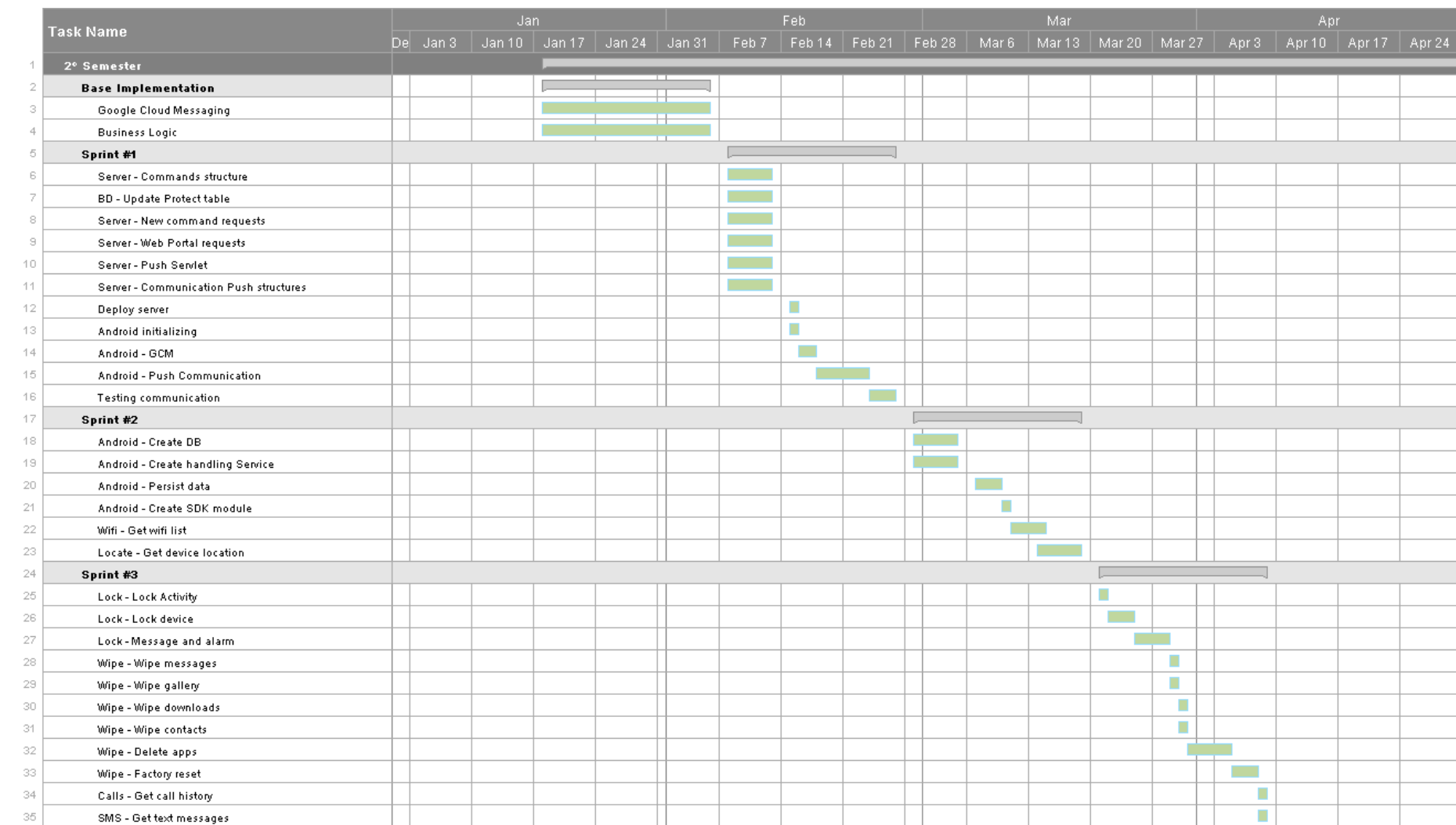


Figure 4.3: Gantt Second Semester Sprints 1-3

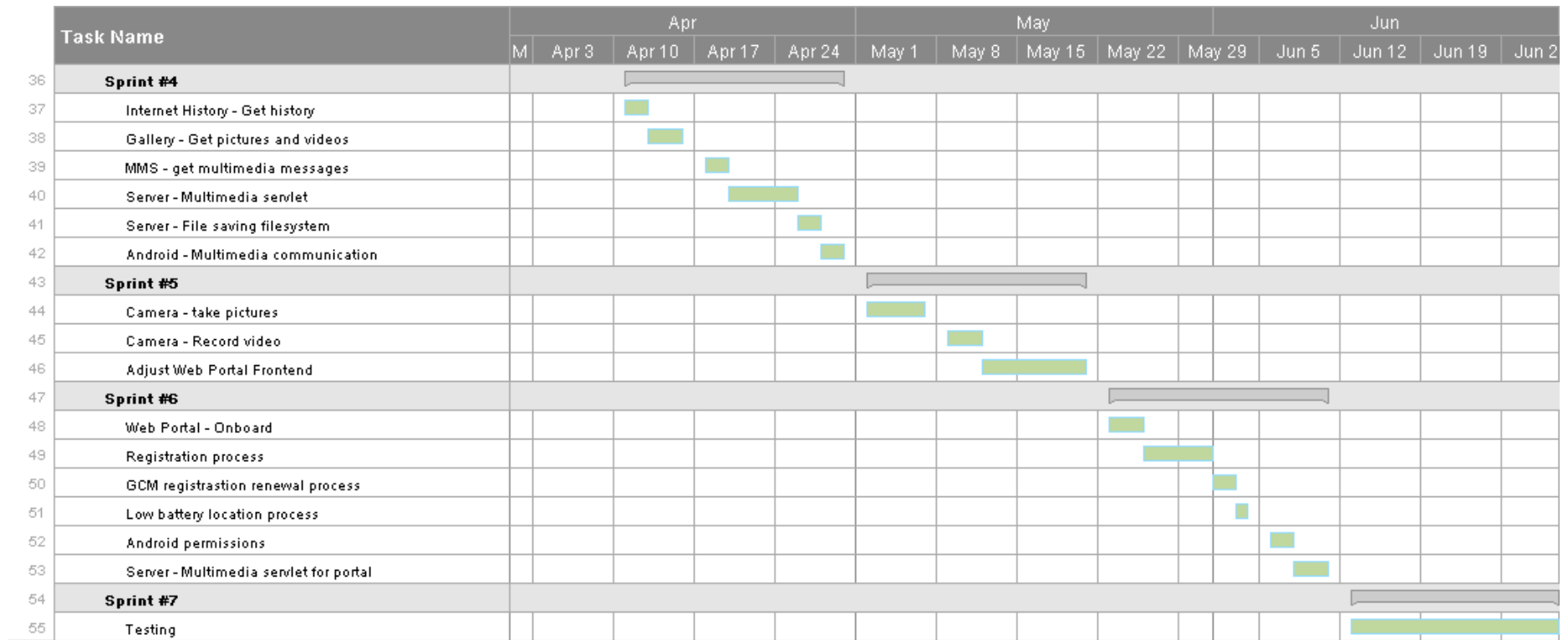


Figure 4.4: Gantt Second Semester Sprints 4-7

of June and finished in the end of the month, took focus only on testing the solution. The selected features which weren't created due to lack of time were the Used Apps retrieval and Livestream. They were the chosen ones to be left aside for several reasons: firstly, the time that was predicted to take in order to develop each of them was one of the biggest, being the Livestream the most time consuming one of all. In addition, their priorities weren't either the toppest ones as it was for example with Locate, Lock and Wipe. Finally, these were the features with the highest probability of having the worst success rate: due to the diversity of devices running many different versions of Android, after some research, it was concluded that the quality and proper execution of these features on the majority of the devices couldn't be guaranteed. For these reasons, they were the assigned the least priority.

The burn down chart for this second semester is shown in figure 4.5.

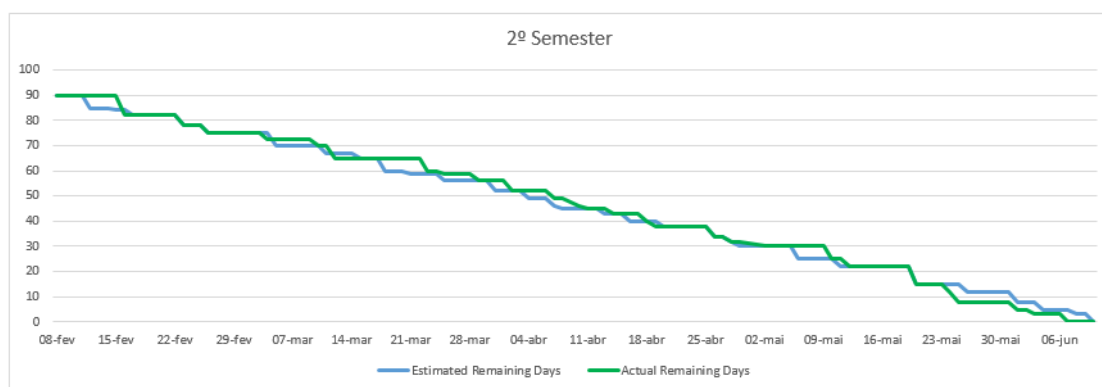


Figure 4.5: Burn Down Chart Second Semester

This time, this kind of chart is better applied since in second semester the development methodology was more closed to an agile approach. This way through the analysis of this chart, we can see that in a high-level view there was no major delays neither advances. It can be seen that during the first, third and fifth sprint the actual remaining days got a bit separated from the prevision but not enough to be a concern. In the other hand, the final development sprint, the sixth, finished a bit before the prevision which lead to the testing phase, last sprint of the year, to start earlier than predicted. More info specific to the every sprint can be found in the respective appendix B - Project Management.

4.3 Risks

As in any project, there are risks which are a constant threat to this software project. Their source may vary and may be even unrelated to software development, but a risk has always an associated probability of leading to the failure of a project and it must be analyzed and mitigation plans must be prepared in order to reduce that probability to the minimum possible. Also, the risks and their respective plans should be revised over time in order to prevent possible changes and new threats. This section describes how the risks were found, defined and which are they. Presented in appendix B - Risks, is the information on the risks evolution and their re-evaluation.

4.3.1 Risks definition

The process of managing risks is based on these stages:

- Identify – Initial stage where the author tries to understand if there are any factors that may create difficulties to achieve the existing goals. This process was performed after the requirements and the architecture were defined, as only after those two steps the author had a better understanding of the project. After this first identification phase, risk identification was done every month. Every risk must be identified and explained with a concise and clear condition that leads to some consequence.
- Analyze – After identified, a risk is analyzed and must have some attributes associated. These attributes are its level of impact on the project, its probability of happening and if they can be mitigated.
- Plan – In the previous section it was analyzed if the risk can be eliminated. If so, a plan must be defined to lead to its extinction, which happens in this third phase.
- Monitor – The last step is to monitor the risks in order to know how they are evolving: if the mitigation plan is working, if they are becoming a bigger threat and so on.

Regarding to the risks attributes, some metrics must be defined so their level can be fully understood.

- Impact – Representative of the level of threat the risk represents to the project in case of happening. Can be either:
 - Minimum – success goals may still be achieved easily.
 - Medium – success goals may be achieved but extra effort is needed.
 - Maximum – success goals won't be achieved and the project fails.
- Probability – Represents how much the risk is likely to happen. Can be either:
 - Low – probability of happening is below 20%.
 - Medium – probability of happening is between 20% and 70%.
 - High – probability of happening is higher than 70%.

After having the risks identified and evaluated, they will be exposed in a matrix which is an easier way to understand how critical they are to the project, visible in figure 4.6. Here, it can be seen that the matrix is divided into 9 different categories with different colors stating as mentioned how critical risks are to the project starting with green coloring the zone with lower danger and colored in red the opposite danger, passing through the yellow zone which is an intermediate from these two.

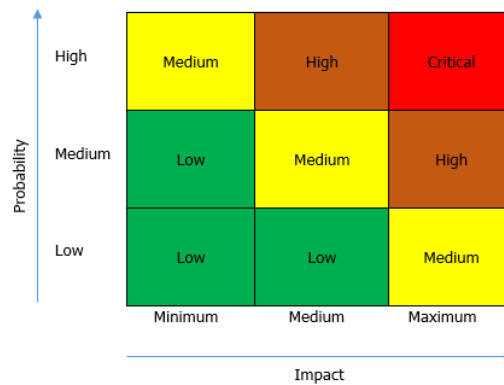


Figure 4.6: Risks Matrix - Danger Level

4.3.2 Risks description and conclusions

This section states the information regarding the risks found for this internship, showing

ID	R.01
Name	Size of project
Description	The project in which the intern will be working on has a great complexity which is the result of years of many people working together to build such a high quality solution. So, its understanding is a time consuming task that is clearly difficult
Impact	Maximum
Probability	High
Eliminable	Yes
Mitigation Plan	Keep in contact with the team that is currently working on the existing solution in order to have feedback and helpful explanations on how the system works

Table 4.1: Risk 01

ID	R.02
Name	Android learning
Description	Although the intern already has some experience in the Android development world and its programming language is Java, there are some different characteristics that alongside with the complexity of this project can become an adversity
Impact	Maximum
Probability	High
Eliminable	No
Mitigation Plan	As this is crucial to the success of the project, the intern will take classes and study concepts and Android programming techniques that improve his capability of developing better Android apps. Also, more experienced colleagues at WIT Software, S.A. are always available to help

Table 4.2: Risk 02

ID	R.03
Name	Bad estimation
Description	Due to the lack of experience in Android programming and the complexity of the project, estimating the total effort needed to complete the software is not easy, and bad estimations can be made. Estimations are a crucial task in software development in the way that the project completion can be projected to a specific date and its real completion date can be 6 months later. Such a situation would be a total failure to the project
Impact	Maximum
Probability	Low
Eliminable	No
Mitigation Plan	Create Gantt charts and ask supervisors to revise and suggest improvements in the estimation plan as they have more experience and in consequence, their opinion is more likely correct than the intern's

Table 4.3: Risk 03

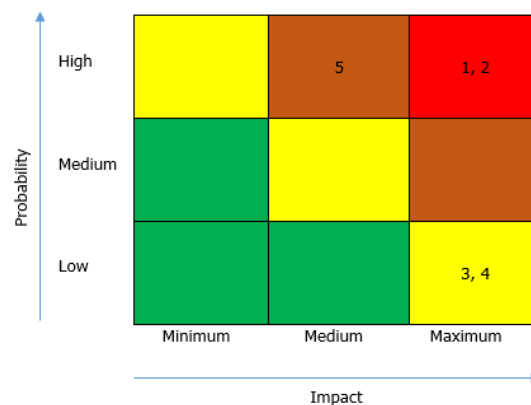
ID	R.04
Name	Android updates
Description	Most of the technologies chosen to work with in this project bring no trouble if a new version is released. However, the software the intern will developed must run in all Android versions and if a new Android update is released, problems may arise and rework to fix this issue must be done
Impact	Maximum
Probability	Low
Eliminable	No
Mitigation Plan	The latest Android version, Android Marshmallow, was released in late September of 2015 and no updates are in sight so can be assumed that although the risk's impact is huge, the probability of happening is very low

Table 4.4: Risk 04

ID	R.05
Name	Different android devices
Description	There is an enormous variety of Android powered devices and each one has its own specifications. Testing the developed software in all devices is not feasible and so possible hidden bugs may pass the testing phases
Impact	Medium
Probability	High
Eliminable	No
Mitigation Plan	Within the available resources, the tests will be run in two devices the author has access: his personal device, which runs Android 4.2.2 Jelly Bean and a device provided by WIT Software, which runs the latest Android version. Also, in an attempt of cover the maximum Android versions, the remaining time will be spent testing the features in an Android emulator.

Table 4.5: Risk 05

These identified risks are represented in the following matrix:

**Figure 4.7:** Risks Matrix

As it is easy to perceive, all risks had either the highest impact or probability and in two cases, both of them. This was a great indicator of how dangerous they were to the project and for that reason is why it was so important to create a plan to deal with all of them, in order to eliminated them or at least reduce their impact.

As this internship has come to an end, all changes made during the year to the risks were documented and can be seen in figure 4.8, which lead to conclusions. Through the analysis of the figure, the first thing that comes in mind is that there are fewer risks than the ones that were set in the beginning. The reason for it is that they were mitigated. The risk R.01 was eliminated due to the author gained enough knowledge of the project he was starting to work on and consequently, it stopped being a threat. The second one, risk R.04 was carefully observed during the year and as the year was coming to an end, the risk's probability was lowered to low and then considered eliminated.

Also, the most hazardous zone to the project, the red zone, used to have two risks in it and now has none. Part of it was explained before, regarding risk R_01, but also, risk R_02 stepped down the its impact and probability, as time went by and the author got more knowledge on the Android programming by developing and by taking courses. Overall, the risks started with high values of Impact and Probability and were certainly a great concern to the project with the one related to the project's complexity even became a problem as mentioned before, leading to some planning changes. However, such as this one, some risks disappeared naturally, others were victims of the mitigation plans. Concluding, the risk monitoring was successful as no risk ruined the project and the mitigation plans were majorly successful.

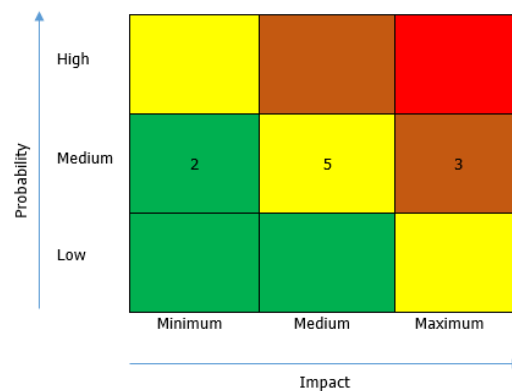


Figure 4.8: Risks Final Matrix

Chapter 5

Requirements

Contrary to the software developers most common dream, the first stage of any software project can't be start coding right away. Beforehand many stages must be complete in order to reach a state of full understanding by everyone related to the project of what has to be done and how people will do it. Unfortunately, that is one of the major causes of failure when it comes to software related projects. In this chapter will be discussed the requirements of this project, concerning which ones were defined and how they were found.

5.1 Requirements Analysis

Through the requirement analysis, the problem which in the beginning may seem complex and difficult to understand, becomes clearer to the developer and this way he is lead to have a better knowledge of what he is supposed to do. One of the best techniques to define and analyze the requirements is by creating user stories. User stories are a tool used in agile software development which focus on thinking through the user perspective of a requirement: looking at a requirement and asking the right questions in order to understand the real problem behind the requirement.



Figure 5.1: User Story

Defining a user story is the action of answering these three questions in a casual language explaining well the intention of the user but keeping it simple and short. After thinking of requirements from this perspective, grouping the user stories by the different users and points of view, it becomes easier to understand them and also helps to define and guide the development phase. To find the user stories, some time was spent thinking

from the user's perspective what would people do if their phone was stolen and there was the chance to get usage info from the device, considering its regular capabilities and what they would expect to be returned from the system. This way, the typical location and lock capabilities came up, as well as features to use the cameras. Also, considering the common usage of smartphones nowadays, text messages and internet usage was also taken into consideration. The remaining features were brought up in a mixture between these discussion and the competitors analysis. The list of all user stories found is presented below in table 5.1:

User Stories		
"As a"	"I want to"	"In order to"
User	Login in the Web Portal	Use its features
User	Get my device's GPS coordinates	Know its location
User	Lock my device	No one can use it
User	Show a message in my device	Get in touch with the person
User	Start an alarm in my device	Find its location
User	Delete all my personal data	Avoid to someone seeing it
User	Wipe my device	Delete everything that is connected to me on the device
User	Get call logs	Know which mobile numbers the thief is contacting by call
User	Get SMS logs	Know which mobile numbers the thief is contacting by message
User	Get MMS logs	Know which numbers the thief is contacting by media message
User	Take a picture from the device	Get info on the surrounding area or on the thief
User	Film a video from the device	Get info on the surrounding area or on the thief
User	Get new media from the device	Know if the thief took any pictures/film any videos
User	Get new browser history	Know what the thief is searching for
User	Start a live stream from the device	Spy through both cameras what is happening around the device
User	Get available Wi-Fi's list	Know which networks are available in the surrounding area
User	Logout	Leave the Web Portal
Web Portal	Connect to the Server	Receive info and send requests

Web Portal	Send requests to the Server	Start the User request
Web Portal	Receive info from the Server	Get the User requested info
Web Portal	Display info received from Server	Complete the User request
Server	Keep available	Be contacted and contact the other actors
Server	Save all requests in the DB	Keep data persistent
Server	Read data from DB	Take actions and send info to actors
Server	Contact the GCM	The Android App can reach me for requests
Server	Contact the Android App	Send requests to execute
Server	Contact the Web Portal	Send results info to the User
Server	Log important actions	Keep track of what happens in the system
Android SDK	Keep running	Wait for User requests
Android SDK	Execute User requests	Fulfil the User intentions
Android SDK	Send the device location	Save the last location before shutting down

Table 5.1: User Stories

5.2 Functional Requirements

After defining the User Stories, it becomes easier to settle the requirements as they are based on the first ones: we take the user out of the user story and what he intends to achieve, and discuss this topic with the stakeholders. For each user story several interventions can be found, each of them taking a different role on the story, which leads to a requirement. The final step is to analyze them and give them a priority, task performed by the author and later on, discussed and revised with his supervisor. These requirements also have dependencies: each one of them in order to be fulfilled might need that other requirement has already been accomplished. These dependencies can be found in the Appendix C - Requirements. The final list of functional requirements can be found below in table 5.2. Here, every requirement has an identifying number, a description and a priority. The process to get every requirement's priority, was a long-term discussion through the year between the stakeholders and the intern. This process, quite similar to a MoSCoW approach, started with an initial discussion on which of these requirements were the most relevant and important to this project. Through the year, after the development phase started as well as the sprints, before every sprint at the sprint meeting, these requirements were discussed again and their priorities revised, taking into consideration the previously done work. This way, some changes were done through the year related to some requirements priorities but also, some new requirements came up and others disappeared. These major changes are stated after the requirements.

Functional Requirements		
ID	Description	Priority
FR_US_01	User logs in the Web Portal	Must Have
FR_US_02	User confirms that understands his actions and its consequences	Must Have
FR_US_03	User requests new device location	Must Have
FR_US_04	User requests to lock the device	Must Have
FR_US_05	User requests to delete personal data	Must Have
FR_US_06	User requests to delete everything on the device through a factory reset	Must Have
FR_US_07	User requests to view the last incoming and outgoing calls	Must Have
FR_US_08	User requests to see the last sent and received SMS and MMS	Must Have
FR_US_09	User requests to see the last record videos or taken pictures	Must Have
FR_US_10	User requests to see the last browsed pages in the device	Must Have
FR_US_11	User clicks on image to zoom it or in video to play it	Must Have
FR_US_12	User requests to take picture from device's cameras	Must Have
FR_US_13	User requests to record video from device's cameras	Must Have
FR_US_14	User requests to start a stream from the device's cameras	Nice to Have
FR_US_15	User enters an invalid URL and is redirected to a 404 page	Must Have
FR_US_16	User logs out the Web Portal	Must Have
FR_WP_01	Display the form from which the user may login	Must Have
FR_WP_02	Display the spy mode warning screen	Should Have
FR_WP_03	Display the main menu of the application	Must Have
FR_WP_04	Display an onboard on homepage	Nice to Have
FR_WP_05	When logged in, account thumbnail is the device picture	Should Have
FR_WP_06	Display the locate menu	Must Have
FR_WP_07	Load Google maps API	Must Have

FR_WP_08	Get from server previous locations found successfully	Must Have
FR_WP_09	Load to Google maps API a previously found location	Must Have
FR_WP_10	Request server the current device's location	Must Have
FR_WP_11	Load to Google maps API the new location found	Must Have
FR_WP_12	Display the lock menu	Must Have
FR_WP_13	Get from server the previous locks set successfully	Must Have
FR_WP_14	Display the previously successful device locks	Must Have
FR_WP_15	Send request to server to lock device and to set a password, if there is none	Must Have
FR_WP_16	Send request to server to lock device and sound alarm	Must Have
FR_WP_17	Send request to server to lock device and display message on customizable screen	Must Have
FR_WP_18	Display the last set password	Must Have
FR_WP_19	Display the wipe menu	Must Have
FR_WP_20	Get from server the previously successful wipes	Must Have
FR_WP_21	Display the previously successful wipes	Must Have
FR_WP_22	Get from server the last successful request for the installed apps on the device	Must Have
FR_WP_23	Display the last successful installed apps result	Must Have
FR_WP_24	Send request to server to get the installed apps on the device	Must Have
FR_WP_25	Send request to server to wipe User's personal data	Must Have
FR_WP_26	Send request to server to delete everything on the device	Must Have
FR_WP_27	Display the calls menu	Must Have
FR_WP_28	Get from server the last successful calls requests	Must Have
FR_WP_29	Load to table the content of a calls request	Must Have

FR_WP_30	Request the last incoming and outgoing calls from the device	Must Have
FR_WP_31	Display the messages menu	Must Have
FR_WP_32	Get from server the last successful SMS and MMS requests	Must Have
FR_WP_33	Load to section the content of a messages request	Must Have
FR_WP_34	Request the last sent and received SMS and MMS from the device	Must Have
FR_WP_35	Display the gallery menu	Must Have
FR_WP_36	Get from server the last successful gallery requests	Must Have
FR_WP_37	Load to section the content of a gallery request	Must Have
FR_WP_38	When the User selects a picture, show the image zoomed	Should Have
FR_WP_39	When the User selects a video, show a player playing the video	Must Have
FR_WP_40	Request the last taken pictures and videos from the device	Must Have
FR_WP_41	Display the browser menu	Must Have
FR_WP_42	Get from server the last internet history requests	Must Have
FR_WP_43	Load to section the content of a internet history request	Must Have
FR_WP_44	Request the internet history from the device	Must Have
FR_WP_45	Display the camera menu	Must Have
FR_WP_46	Get from server the last successful camera requests	Must Have
FR_WP_47	Load to section the content of a camera request	Must Have
FR_WP_48	Request server to take picture from device	Must Have
FR_WP_49	Request server to record video from device	Must Have
FR_WP_50	Display the live stream menu	Nice to Have
FR_WP_51	Request server to start streaming from device's camera	Nice to Have
FR_WP_52	Display the device and network menu	Nice to Have
FR_WP_53	Get from server the device information	Nice to Have

FR_WP_54	Get from server the successful network requests	Nice to Have
FR_WP_55	Load to section the content of a network request	Nice to Have
FR_WP_56	Request to get the available Wi-Fis to the device	Nice to Have
FR_WP_57	Display FR_WP_01 when the user logs out	Must Have
FR_WP_58	Display 404 page when page is not found	Must Have
FR_WP_59	Display 500 page when an internal error occurs	Must Have
FR_WP_60	When an error occurs, an error notification is shown	Must Have
FR_WP_61	When the session times out or has a problem, display FR_WP_01	Must Have
FR_WP_62	Once logged in, the icon representative of the user is an image of his device	Nice to Have
FR_WP_63	Once logged in, the device image in the Lock menu is an image of his device	Nice to Have
FR_WP_64	Web Portal presents an identity validation certificate.	Nice to Have
FR_WP_65	Web Portal must be responsive, providing an optimal viewing and interaction experience, changing the webpage layout according to the size of the screen	Must Have
FR_SE_01	The server must accept new requests from the Web Portal	Must Have
FR_SE_02	When a new request arrives, the server must create and associate a ID to it	Must Have
FR_SE_03	When new user requests arrive, the server must save the info associated with it in the DB	Must Have
FR_SE_04	The server must contact the GCM to send the new request ID to the device	Must Have
FR_SE_05	Once the Android app sends the ID, the server must read the ID associated request	Must Have
FR_SE_06	When the Android app sends the ID the server must respond with the request info associated to the ID	Must Have

FR_SE_07	When the Android app returns the execution resulting data, the server must accept it	Must Have
FR_SE_08	Once the Android app finishes a request and returns data of multimedia type, the server must save it in the filesystem	Must Have
FR_SE_09	Once the Android app finishes a request and returns data of text type, the server must save it in the DB	Must Have
FR_SE_10	When the user requests previous requests, the server must send the successful requests' information	Must Have
FR_SE_11	When the user requests media files, the server should send an ID per file	Must Have
FR_SE_12	When the Web Portal requests media files with their ID, a servlet should send the files to the Web Portal	Must Have
FR_SE_13	User's passwords must be stored as hashes and never as plain text	Must Have
FR_SE_14	User's media files stored in the file system must be encrypted	Nice to Have
FR_ASDK_01	The Android app must run in the background as an Android service	Must Have
FR_ASDK_02	The Android app must be a system app in order to not get uninstalled	Must Have
FR_ASDK_03	GCM messages must be received	Must Have
FR_ASDK_04	When the ID is received, it must be sent a request to the server	Must Have
FR_ASDK_05	Android SDK must accept the new request from the server	Must Have
FR_ASDK_06	Once the request is received, it must be executed immediately	Must Have
FR_ASDK_07	When the request is executed, the success or failure status must be sent to the server	Must Have
FR_ASDK_08	When the battery percentage is below 5%, the SDK should stop working	Must Have

Table 5.2: Functional Requirements

5.3 Non-Functional Requirements

Once the functional requirements are set, defining the non-functional requirements is an easier task. Such as the functional ones, these requirements are discussed and defined with the stakeholders, with the characteristic of being quantifiable. The complete list is shown below:

Non Functional Requirements		
ID	Description	Priority
NFR_SE_01	Scalability - Server should be capable of handling media requests containing files with at least 100MB from two concurrent users without crashing the server	Must Have
NFR_ASDK_01	Efficiency - SDK must be efficient in terms of battery usage closing resources when it doesn't need them anymore	Must Have
NFR_ASDK_02	Compatibility - SDK must be compatible with all android versions above 2.3	Must Have
NFR_ASDK_03	Effectiveness - SDK must execute all functions successfully on all android versions as long as they support them	Should Have
NFR_ASDK_04	Performance/Response time - Message handling must be done as soon as possible. If the phone is working right after receiving the request to execute a feature, the app must execute it. If the device's battery is below 5%, as soon as it has over that percentage of battery, the request must be handled	Must Have
NFR_ASDK_05	Recoverability - If the device crashes, right after returning to functional state, the app must start itself. If the device runs out of battery during some feature execution, after rebooting with enough battery, the app must execute the previous request again	Nice to Have

Table 5.3: Non Functional Requirements

5.4 Requirement Changes

As previously said, some requirements were changed, others appeared and even others were discarded. This is very common in a Software Engineering project as requirements tend to change. In this case the changes were minor and represented no big issue to the project as their priorities were mostly the lowest, nice to have, and so, they were not seen as crucial to be realized.

The changes verified in the requirements were:

- **Used apps** - The set of requirements related to the ability of retrieving the used apps was discarded. After some investigation and implementation, it was realized that the probability of retrieving the objective information with the best quality was not enough to build a feature with quality. For that reason, they were deleted;
- **Onboard** - Related to contextualize the user with the Web Portal, the onboard requirements came up;
- **Account thumbnail** - As there were requirements with higher level of priority to be developed, the priority of the account thumbnail presented in the Web Portal was lowered;
- **Requirement refinements** - Mostly related to the Web Portal, some requirements came up as refinements of already existing requirements, meaning that inside some feature, several other smaller ones had to be accomplished;
- **GCM** - After some investigation and implementation of the XMPP version of the GCM, the one that allowed the Updates Server to know when the messages were received by the device, it was realised that the rate of success of these acknowledge tickets was very low and inaccurate. Due to that, the requirements related to the GCM were removed as they related to build a XMPP server and handle such acknowledgement messages. As the implemented server-side version contacts the GCM servers through HTTP, the requirements related to the GCM stopped making sense and were removed;
- **Certificate** - For the same reason as the third change, it was considered that there were requirements with higher priorities to be done than the creation of the certificate to be applied to the Web Portal. So, its priority was lowered;
- **File encryption** - The final change is related to file encryption in the server. This requirement appeared from the idea of encrypting the media files stored in the filesystem. However, due to priorities of other requirements this one was considered out-of-scope to give priority to them.

Chapter 6

Architecture

One of the major problems of software teams is communication. The 65 to 80%[7] failure rate to meet the company's objectives of IT projects is a scary and disappointing number that must be changed. There are many processes that are the cause for that failure, and if people improve those processes, the success rate will rise. One of them, is the architecture definition.

Defining an architecture is a complex process that brings a lot of responsibility as the outcome of all the diagrams and documentation shapes the team concept of what the project is and how it must be developed. If the architecture has ambiguities, does not focus on a specific and important point or uses the wrong words, the architect's concept of the project will be different from the other team members and later on problems will arise. Due to that, it is important to create an architecture that is simple but complete, presenting all the information required to understand the project and how its components interact. That way, success is an easier goal.

This chapter refers to the explanation of the architecture of this internship, presenting easy to understand views of the multiple structures, what they do, how they interact with each other the content that flows. A more in-depth view of all the content can be found in the appendix A - Architecture.

In order to achieve this ultimate goal of creating an easy to understand system architecture, some rules must be applied and different shapes, colors and naming must be explained. Some high level diagrams are easily explained with allusive figures and images rather than the typical boxes that can be everything and have hundreds of arrows looking like an airport, and most of the times these boxes tend to explain nothing and fail to present any type of information regarding what is presented, what they do and how they communicate. With this in mind to prevent these fearful boxes, this chapter follows Simon Brown's C4 architecting model [39].

The C4 model, sustain that it is impossible to show an entire system design in a single diagram and so, several views must be created. Having different views, it becomes easier to understand from multiple perspectives the same structure, as it reduces complexity, highlights different aspects of the solution and helps people to navigate through a complex codebase. Another principle of this model is that when attempting to understand a complex system, the first view to be seen should be the most abstract one so the system can be seen as a whole, with which actors, services it interacts and which are its main objectives. Incrementally, the next views should show the system into more detail giving a better understanding of it and it can go even to the lowest possible, showing which methods an important class can have.

The model name, C4 Model, comes after these categories:

- System Context
- Containers
- Components
- Classes

System Context – The most abstract view of all. Should show the system as a whole, who will use it, which services does it use and which roles every participant have.

Containers – Zooms into the System Context and presents the high-level technology decisions, responsibilities and how they communicate with each other.

Components – The next step is to continue to get detailed, so components are pieces of a container that show of what the container is composed of internally. They can be multiple structures such as services, sub-systems, layers, packages, and so on.

Classes – The last and most detailed type of view, classes are optional to include and its purpose is to show implementation details of a certain component.

Following the four C's, the diagrams representative of Updates Protect architecture with the exception of some high level ones, are divided into these different types of diagrams. Through them, all the participants can have one of two shapes: either a person or a box. While the first one represents obviously a person that interacts with something, the box can be any other kind of participant and has the following structure:

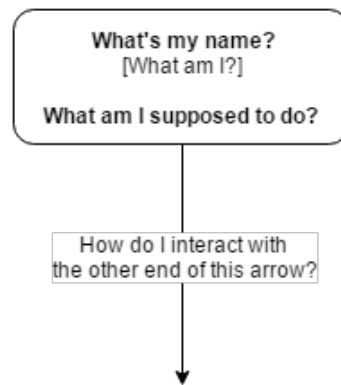


Figure 6.1: Box structure

Boxes also can be different according to its color, although the color differences are only present to make it easier to understand the different levels between them. The outer boxes are always grey and inner boxes are either white or a whiter shade of grey than its parent's grey color. The flow of actions also has a pattern: always start from top and ends at the bottom.

6.1 Context

As stated previously, the first diagram is the context diagram giving the most abstract perspective of the system and who interacts with it. The diagram represents the basic and

simple flow of actions: the user has the intent of gathering info of the device and makes a request to the system, which has the responsibility of taking action on the device and communicate back to the user. To do so, it sends a wake up message to the Google Cloud Messaging servers, which by their turn send the message to the device. The Android app receives it, and requests the server to send the commands that it must execute. Once finished executing them, the app will send the result and a request for more commands that might be waiting to be executed.

This first diagram represents the basic and simple flow of actions: the user has the intent of gathering info of the device and makes a request to the system, which has the responsibility of taking action on the device and communicate back to the user.

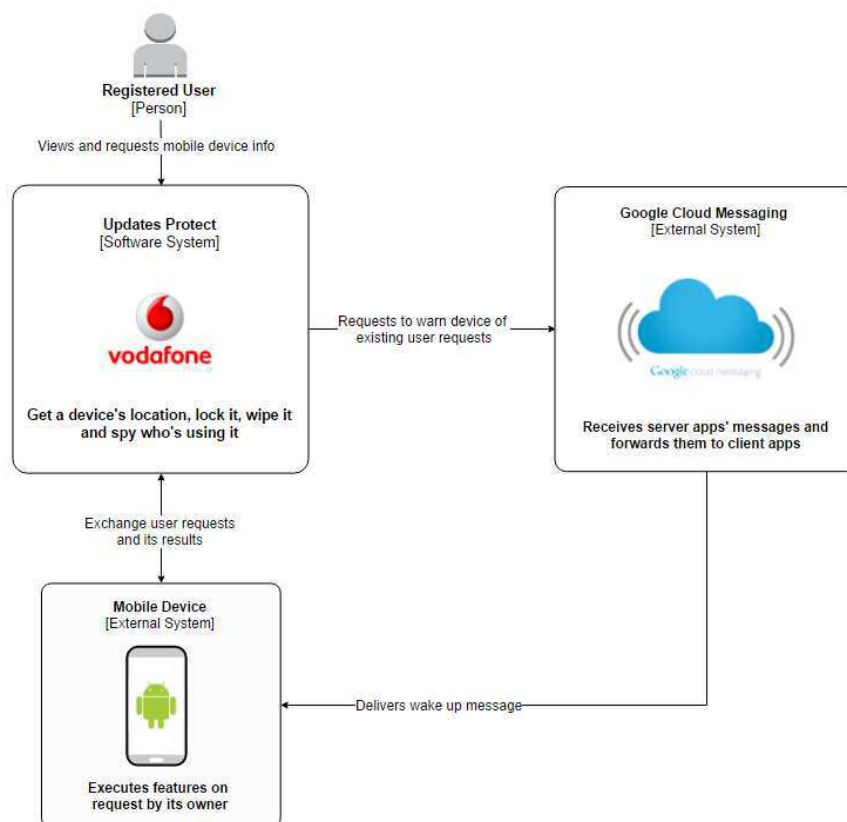


Figure 6.2: Context

To do so, the Updates Protect system, name given to the internship solution, upon receiving the intent of the user contacts the Google Cloud Messaging system that sends a message to the device, which is waken up and then contacts the Updates Protect system to know is he supposed to do.

6.1.1 Physical View

To completely better understand how these participants interact with each other and who they really are, it is presented a physical view, a view which intents to demonstrate the previously mentioned big components of the whole solution, who they are and how they communicate, shown in figure 6.3.

The user, legitimate owner of the device, only interacts with the Web Portal through his browser. In it, he can request the multiple features available. The Web Portal is running in a Apache Tomcat web server, and has the purpose of getting the user actions and send to the server its requests, presenting the information gathered afterwards in the browser. The server, running as well on a Apache web server, once receive the requests, persists all the information in a object called Command in the DB, and then requests the GCM server to wake up the device with a message that contains an ID of the created command. The Updates Protect app upon being waken up, contacts the server to get the command information, executes it and returns the info, which will be presented to the user.

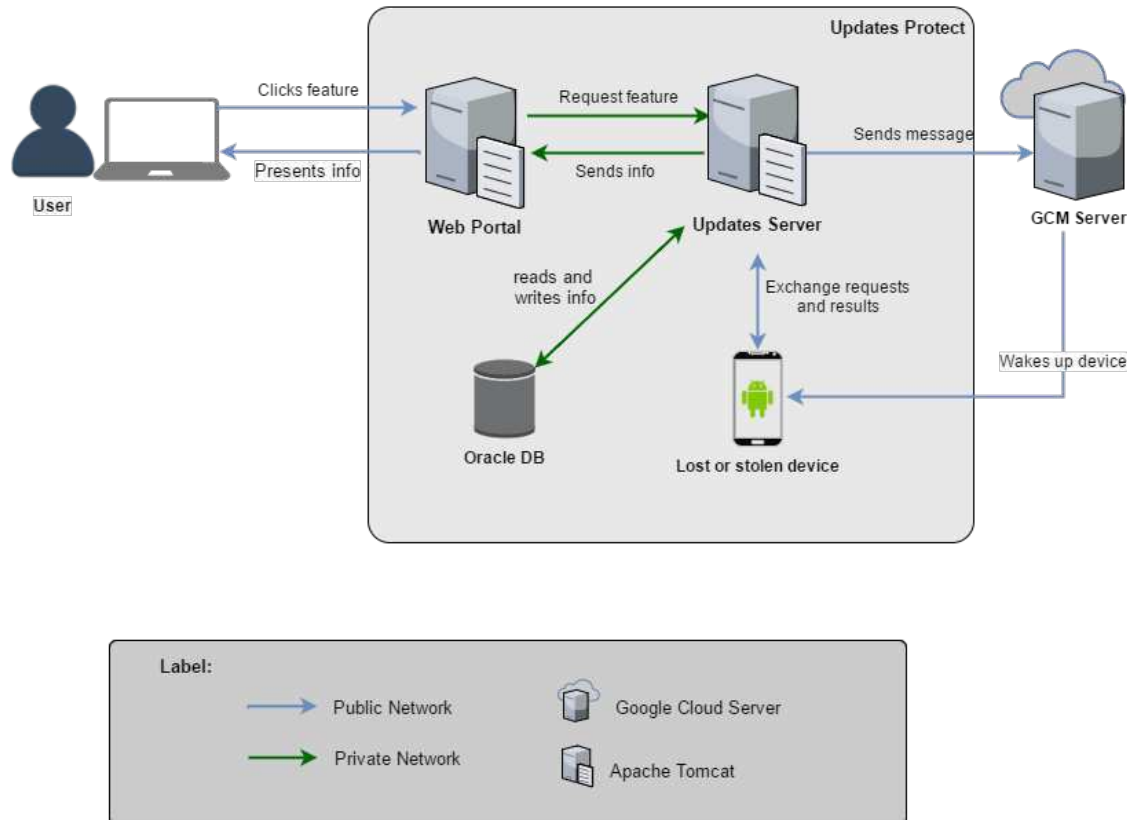


Figure 6.3: Physical View

So, the complete flow of actions is:

1. User clicks the feature on his browser;
2. Web Portal requests it to the Server;
3. Server saves the info as a Command in DB;
4. Server sends the command's ID to the GCM server;
5. GCM Server forwards the ID to the app;
6. Device's app wakes up and contacts the server requesting the command's info of the command with that ID;
7. Server sends command;
8. App executes the command;

9. App returns result to Server;
10. Server saves the result;
11. Server sends the result to the Web Portal;
12. Web Portal presents the info to the user.

Such actions will be explained next, in the containers section.

6.2 Containers

The Updates Protect system is divided into 3 containers, the Web Portal, the Server and the Android App. All have different structures and particularities which make it possible to work and communicate. As one of the architecture of a system main purpose is to explain how the architectural drivers or quality attributes are guaranteed, this section details the their structures and way of working.

6.2.1 Web Portal

In this section is presented the architecture diagram for the Web Portal container, describing its internal components and their relationships. In order to build a reliable, scalable and maintainable web application, a suitable web framework is needed. The decision of choosing a web framework is not trivial neither always consensual, so to make a decision on which framework to use, a previous investigation must be done. First of all, analyzing requirements and constraints is an important step of choosing the best framework because it can reveille some design or technical details that can narrow down the possible frameworks which can be used. The next step is to do a comparison between the remaining frameworks so their advantages and disadvantages can come up and show which we benefit the most in using. After that, a well-founded decision can be made. With this process in mind, a first analysis of the requirements and constraints revealed only one constraint to this matter: the programming language needed to be Java. The final decision was to use the Struts 2 framework and more about this decision can be found on the respective architecture appendix. The Struts 2 encourages developers to adopt a MVC architecture. MVC is characterized for dividing a software application into three parts connected to each other: Model, View and Controller.

Through this clear separation of software parts by their roles, it becomes much easier to structure code which makes it easier to read, understand, maintain and if necessary in the future to improve or to add new features. So, following these principles the structure of the Web Portal container is divided into six components:

- Filters – component responsible of filtering the requests coming from the user, preparing and executing phases of the Struts dispatching process.
- Struts – this component is the core configuration file for the framework, mapping the existing actions to the respective results.
- Model Layer – gathers all the models of data needed.
- Logic Layer – After receiving a request from the user, Struts populates input data into this layer which performs all the business logic, gets data from the server and when all the actions are done, send the result to the View Layer.

- View Layer – Responsible for receiving forward requests from Struts component and data from Logic Layer, rendering the result in form of a JSP to the Front End.
- Front End – Despite the fact that this component is a container of the View Layer because it is just the presentation of the created JSP's in that layer, the responsibility of Front End is to present info to the user and to receive his actions. For this reason, it must be separated from the View Layer.

The next diagram shows exactly those components and presents the simple flow of actions in the system:

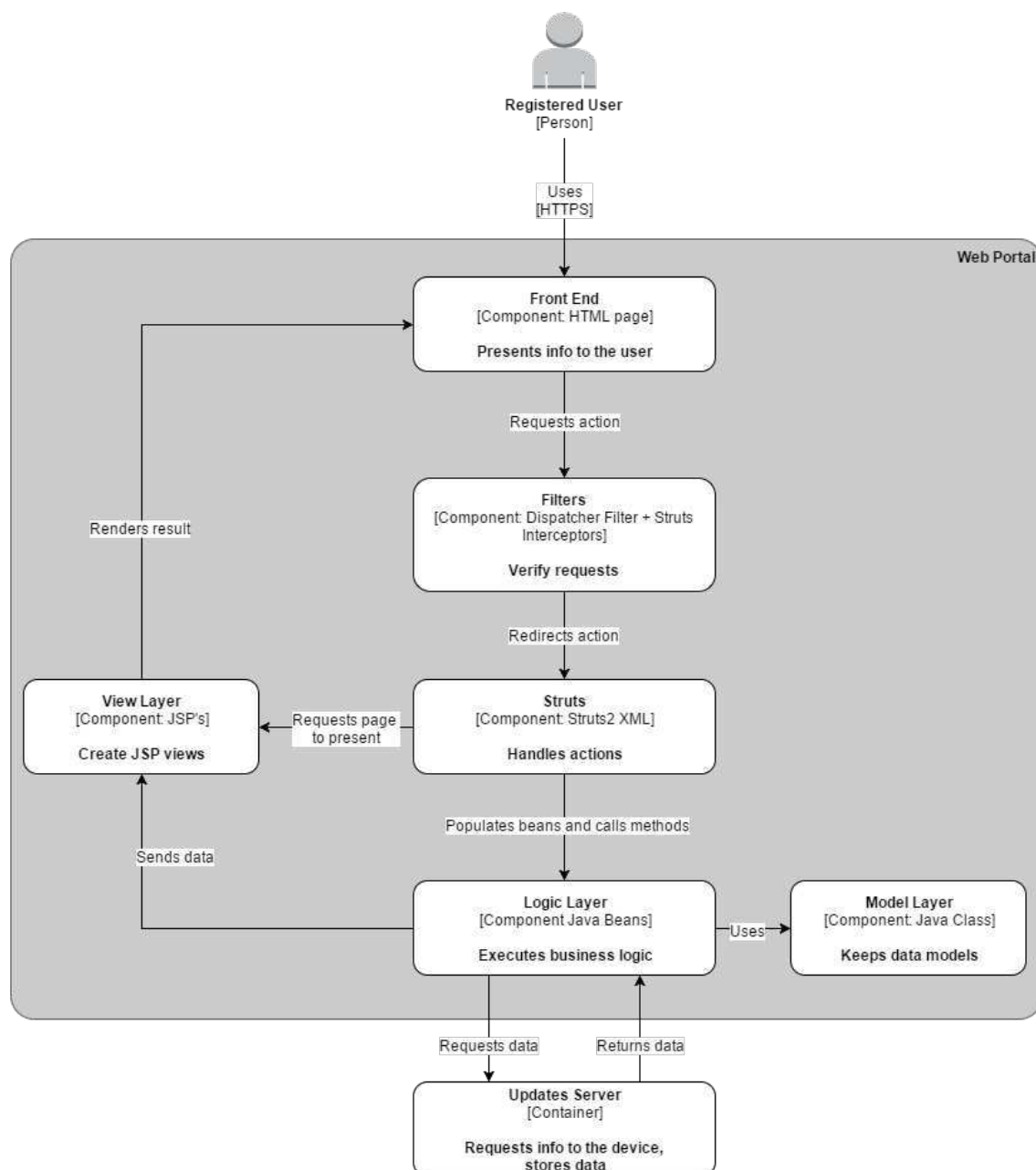


Figure 6.4: Web Portal Container

6.3 Server

The Server container is responsible for receiving, handling and forwarding the requests coming from the Web Portal to the Android SDK. Its components are presented in figure 6.5.

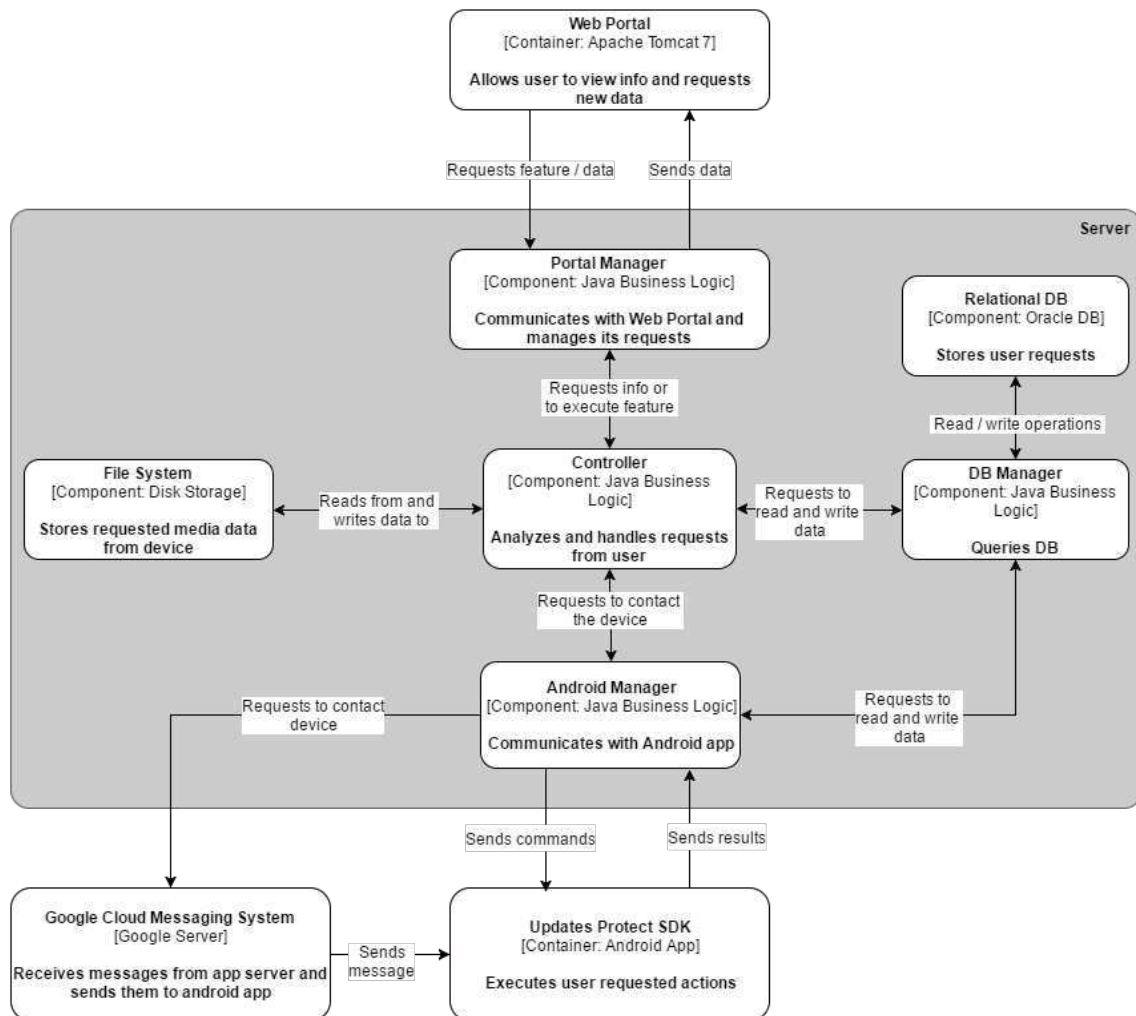


Figure 6.5: Server Container

6.3.1 Components

The Server components and their responsibilities are:

Portal Manager

This component is composed of a REST interface and a servlet. The first has the purpose of communicate with the web portal, redirect the request to the Controller and return the response back to the web portal. While the REST interface deals with all requests concerning text content, the servlet included in the Portal Manager, was built to send the multimedia files, reading them directly from the file to the outputstream and through it, alongside another servlet that handles the arrival of the media files to the server, the quality attribute of scalability is guaranteed.

Android Manager

A Java Business logic that sends the commands to the GCM server and then when the device reaches the server requesting the commands to execute, it interacts with the device through servlets, one for text content, and one for multimedia files. The second one, is also the second servlet mentioned in the previous Portal Manager paragraph, which states that these servlets guarantee the scalability of the server.

File System

Server File System that has the responsibility of saving media files that are requested by the user.

Relational DB

Pre-existing Oracle DB which goal is to save the history of user requests and if those requests were well-succeeded, it saves also the info requested as long as they are non-media files.

DB Manager

This component is a set of modules which handles requests coming from multiple components, requesting to read and write data from the DB.

Controller

Is the core of the server and has multiple responsibilities such as:

- Contains the logger module, a module that logs everything that happens in the server to multiple files;
- Has the XML module, that parses to Java objects the content received from the device as XML, and vice-versa parsing XML received to Java objects;
- Constantly checks the DB for commands that were never sent to the respective devices and triggers their sending process, or even checks for commands that were sent but the device response never arrived, so they will be sent again;
- Has the filters and encryption modules required to encrypt all communication with the devices, securing it;
- Includes the login module to authenticate the users credentials coming from the Web Portal;
- Contains the Protect module where is located all business logic regarding the requests coming from the Web Portal, either being to get new information or already saved information;

6.3.2 Data Storage

With the objective of storing all user's information, different approaches were used to save different kinds of information:

Text Data Storage

In order to record all data gathered from the user's devices, some modifications were needed to be done in the database. As there were already a table to record the user's subscriptions and request commands, a new table specific for this project was created to store the information retrieved. So, the Protect_Commands table was designed having the following columns:

- Request_Id - Identifier of the command;
- Command_Type - Type of request such as a lock or a wipe;
- Msisdn - Identifier of the user;
- Create_date - Date from when the command was created;
- Last_Update - Date from when the command was updated for the last time;
- Content text - When the command is executed successfully, the result to be sent to the user is stored here;
- Additional_Info - Additional info that may be relevant;
- Command_Status - Field which says if the command is in a queued, processing or final state.

This settled information resulted in the creation of the table:

Protect Commands		
PK, U	Request_Id	VARCHAR2(255 CHAR)
	Command_Type	VARCHAR2(30 CHAR)
PK	Msisdn	VARCHAR2(65 CHAR)
	Create_Date	TIMESTAMP
	Last_Update	TIMESTAMP
	Content_Text	VARCHAR2(200 CHAR)
	Additional_Info	CLOB
	Command_Status	VARCHAR2(20 CHAR)

Figure 6.6: Protect Commands Table

Media Data Storage

As the user request might also return media files, this brought up a architectural decision: where to save binary files as they needed to be saved in order to the user can see them anytime he wants. There were two options: either the files were saved as a BLOB in the database or stored as files in the server's filesystem. Both have multiple advantages and disadvantages relevant to this matter as we can see next.

Database main advantages:

- Queries are fast and easy to query;

- Databases are ACID;
- Databases are fault-tolerant;
- Backups automatically include the file binaries.

Database main disadvantages:

- Size of binary file differs between DBs;
- Size of database increases a lot;
- Complexity of returning files to Web Portal is bigger;
- Performance of retrieving files is weak when files are bigger;
- Hard and expensive to scale.

Filesystem main advantages:

- Better for streaming with a good performance;
- Easy and cheap to scale;
- Good reading performance;
- Specific features in the NTFS file system such as Remote Storage are available;
- Better performance over time.

Filesystem main disadvantages:

- Harder to access a specific file;
- Fragmentation on the media storage may become a issue;
- No easy backups.

As there is already a lot of discussion on this topic and not always a right answer, the best answer depends on the case. In this case there are a few things which lead to the right answer. First, the retrieved files may be a list of images and videos which can have a few Kb's, but can also have many MB's or even GB's. It is easy to imagine if the thief uses the device to record a 5 minute video with resolution of 1080p the resulting file could be from hundreds of MBs to even some GBs.

Secondly, if it may be hard to store such huge files, to get these files to the user may be another issue and here, performance is a must. Yet on this matter, if the system scales, a lot of storage will be needed, so its cost may be a factor. Also, the users that will access the Updates Protect won't make a hard usage of it, meaning that only when they loose the device or get robbed they will access the Web Portal, and so an intensive backup plan shouldn't be needed as probably files older than 1 month won't matter to the user. Besides, Updates Protect has no objective to serve as a cloud storage which the user probably has somewhere, being this sub-feature part of the spy capabilities to provide some safety to the user.

These were the major factors taken in consideration to come up with a final decision which was the filesystem. With it, the performance of storing and getting files would be better, the cost of scaling wouldn't be as large as with the DB solution and the maintenance of old files and backups isn't really the major issue here, so the best answer in this case was the filesystem.

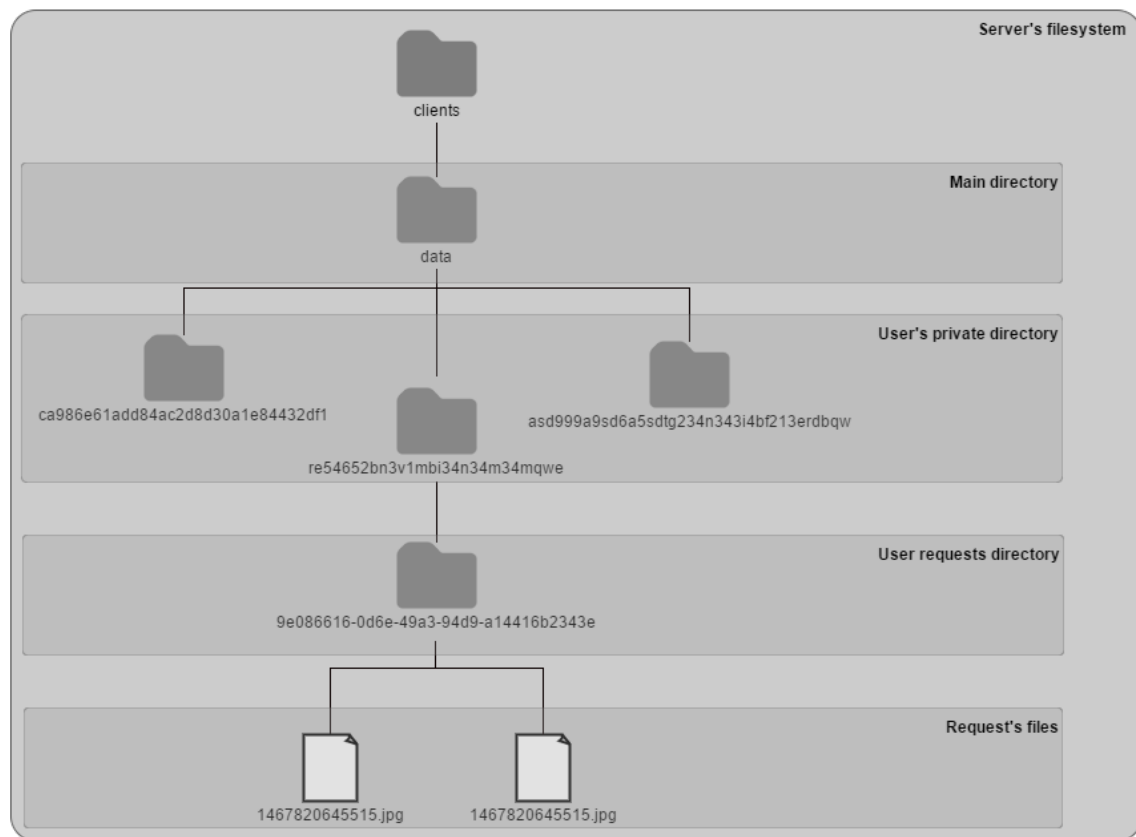


Figure 6.7: File system hierarchy

For each registered user, a different directory is created with the name of the user's encrypted MSISDN, to keep his privacy. Inside the user's private directory, the server creates directories regarding different requests made by the user with the name equal to the request's id, and it is inside those directories that the resulting media files of that request are stored, this way this division facilitated the organization of files per requests. File names are always related to the date when that file was created in the mobile device and the file extension remains the same.

6.4 Android SDK

The last container of the system is the Android SDK. This app which is the final component of the internship is an Android prototype app with few information or screens to be presented. Its main purpose is not to display information to the user but to run in background as an Android service, ready to be awoken by a message informing that there are new requests to be executed. Therefore, this container only has one component which is the SDK, an Android Module with the responsibility of receiving and executing the requested commands returning their result.

6.4.1 SDK Components

The SDK is composed of several inner-components which interact with each other and are explained in this section.

As mentioned, the messages arrive to the app by two entities - the Push System and the Google Cloud Messaging, though only the second one was used. Both have the

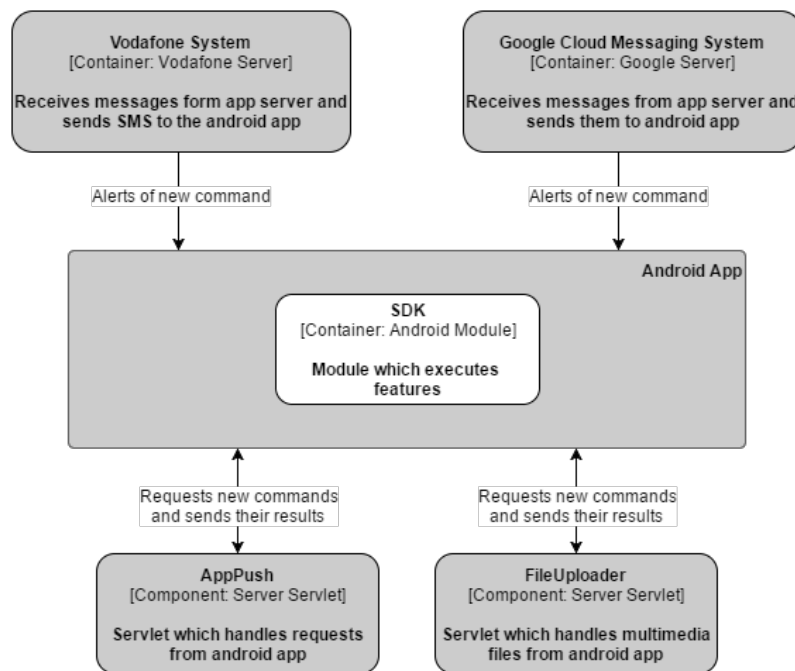


Figure 6.8: Android Container

objective of alerting the app of the existence of commands to execute. Upon arrival of these warning messages, the app then contacts the AppPush servlet requesting commands. After executing them, it sends the results either to the ProtectMultimedia servlet if the result are photos or videos, otherwise it sends the results to the AppPush.

Service Manager

The center of operations. This component is an Android IntentService, a type of service that can perform long-running operations in the background without presenting an interface to the user which means that is always running and waiting for orders without being noticed, handling asynchronous requests on demand and for each request that arrives, a new thread is created to handle it and put in a thread queue. Once the job is done, the thread automatically stops itself[40]. This IntentService is what assures the quality attributes of Efficiency and Performance/Response time, seen in table 5.3, as the app isn't consuming significant battery until a message arrives, time when the Service Manager creates a thread to handle the it right away and once everything is done, the thread is killed and the app goes to idle again.

In order to contact the Service Manager requesting some action, classes must send an intent which is a type of call for an operation to be performed. When an intent is sent, the IntentService is automatically started and in the interest of knowing what kind of action was requested, inside the intent may exist an action, a String representative of something to be done. The Service Manager handles seven types of actions:

- Nudge - The app received a message through the GCM listener so it must contact the server for commands to be executed;
- New command - The app reached the server and the new command arrived so it must be executed;

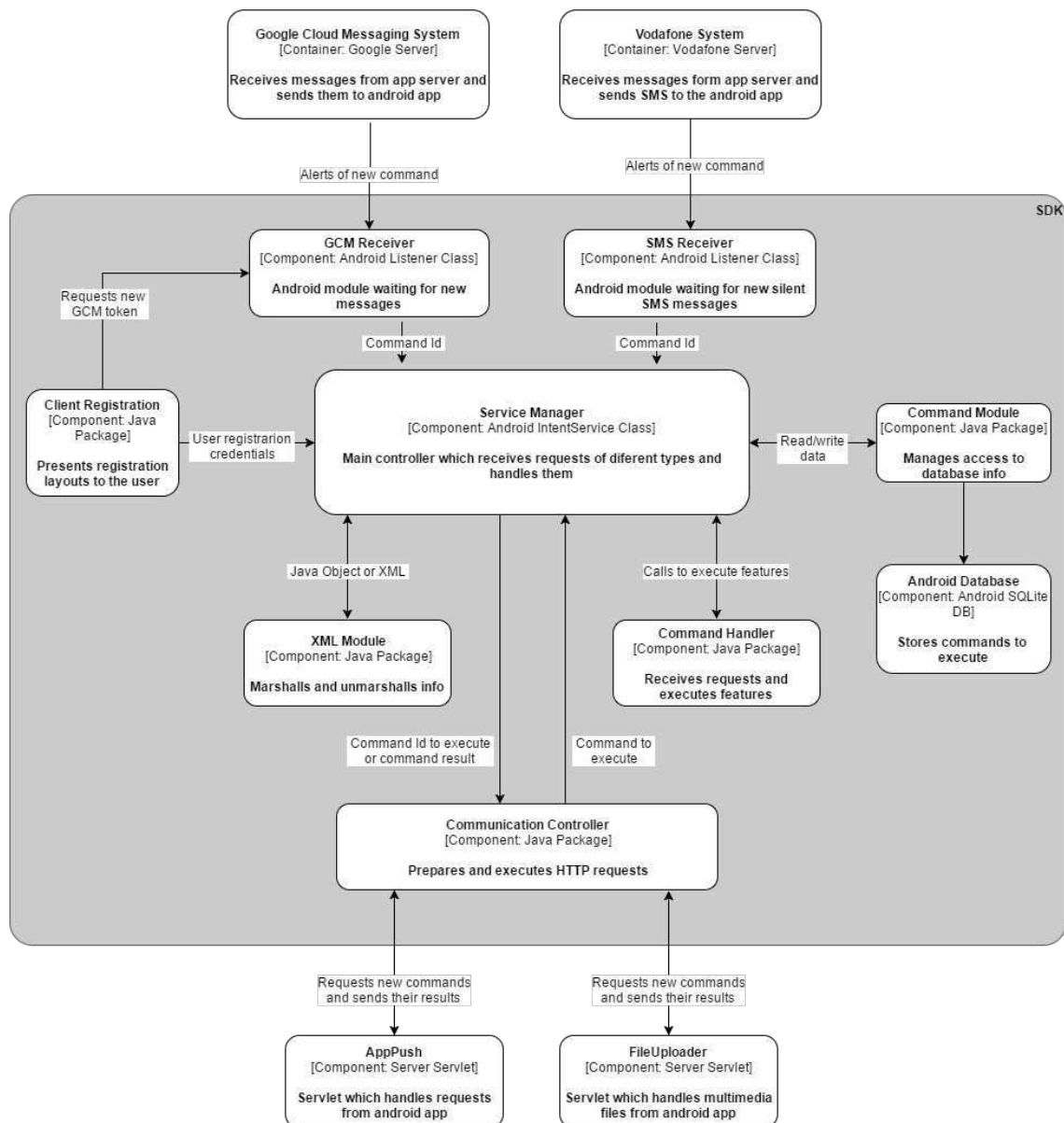


Figure 6.9: Android SDK container

- New result - The app finished executing a command and it must send to the server its result;
- Queued commands - The device just restarted and the app must check the database for commands that were not executed or finished;
- Register GCM - The app renewed the GCM registration id and it must be sent to the server;
- Register client - The device was run for the first time, the user entered his credentials and the app must send them to the server;
- Low battery - The device has low battery and so the app must get the device's location and then send them to the server.

As this component is the core of the SDK, it also checks if the device has a minimum level of battery, stopping its actions when the level is very low.

GCM Receiver

The GCM Receiver includes the capability of handling the GCM actions. This does not mean only the receipt of downstream messages from their servers but also the capability of registration and update of the device id on their account. So, its communications are getting the messages from the GCM server and from the Client Registration module the requests to register, and then, forwards the info to the Service Manager.

SMS Receiver

Sharing one of the tasks of the previous module, the SMS Receiver handles the silent SMS received from the Vodafone servers, decrypts its content and forwards them to the Service Manager as well.

Client Registration

When the app realises that the user isn't registered yet, that process starts. This means that there will be presented to the User 3 or 4 different screens depending on the need of requesting Android permissions. After the User enters the password, an asynchronous task starts getting a new GCM registration id and then the Service Manager will deal with the remaining actions of contacting the server, saving the new registration.

Command Module

Just as the similar version of the Server, Command Module is an API to perform the read/write info on the DB, having the necessary interface and methods to do such actions.

XML Module

With the same structure of the Command Module, the purpose of this module is to convert Java objects in XML strings and vice-versa.

Communication Controller

Everytime the app needs to contact the server, this module is used. All the data starts flowing through the ProtectCommunication class, which divides the type of request and forwards it to one of the Requests classes, which build the content to be sent. Then, the HTTP requests are executed and the responses received.

Command Handler

This module is the one which executes the features on the device, and is divided into a main class that receives the requests and forwards them to the respective module. Once the execution finishes, the Service Manager is contacted so the response can be sent to the server. The module that guarantees the quality attribute of Effectiveness, executing the

features in all possible Android versions.

Database

In order to execute all functionalities with a success rate of 100%, the app must have the ability to resist failure that may come through device problems such as low battery, random OS crashes and other causes created by third-party apps or services. Due to that, the app must continue to execute previously interrupted commands, and to know when they happen, every step along the way must be recorded. So, these steps are persisted in an Android SQLite database and guarantee the last quality attribute of Recoverability. This database only needs to record command's info and so, a unique table is required.

Protect Commands		
PK	_id	INTEGER
U	Request_Id	TEXT
	Command_Type	TEXT NOT NULL
	Create_Date	DATE NOT NULL
	Update_Date	DATE
	Response_Info	TEXT
	Request_Info	TEXT
	Command_Status	TEXT NOT NULL
	Failure_Status	TEXT

Figure 6.10: Android Database

- Request Id - The id of the command received to execute;
- Session Id - The id of the client's session in the server. When several request are made before the device starts the handle them, they all have the same session id;
- Command Type - Type of command such as locate, lock or wipe;
- Create Date - Date when the command arrived to the app;
- Update Date - Date of the last time the command was updated;
- Request Info - Additional info of the request existing for example on lock requests, as they have an optional message or alarm;
- Response Info - When the command finishes its execution, the result is saved to be sent to the server;
- Command Status - Status of the commands' execution. May be the following types:
 - Incomplete - When the nudge message arrives but there is still no info on the command;
 - Queued - The command's info arrived but is still waiting to be executed;
 - Processing - The app is executing the command;

- Success - Final state of the command when the execution went as expected;
 - Error - Final state of the command when the execution went wrong.
- Failure Status - When the execution of the command fails, the reason is saved to be sent to the server.

Chapter 7

Development and Final Product

After having finished most of the path settling the requirements and planning how would the implementation go along the year, the development took the first step and the final product started to be built. This phase took most of the year and once done, only the testing phase was left. However, as it is usual every project, many challenges appear along the way, creating more or less complex problems to be solved and this project was no exception. David May, Professor at University of Bristol once said - "*With good program architecture debugging is a breeze, because bugs will be where they should be*". Well, many times this applies not only to bugs, found in the final step of the project during the testing phase, but also applies to the implementation, during development phase. In this particular project, even though the architecture defined was well built, some weak spots were missed and some challenges came up. This chapter intends to show some challenges found during the development phase and the final product of this internship. More on this topic, with in-depth explanations can be found in Appendix D - Development.

It is very hard to design a proper architecture that leads to confusion or roadblock ahead in development. Fortunately, this project's architecture was well defined but even though some minor problems appeared such as the inefficiency of the GCM XMPP implementation. Due to its slowness and bad execution, this implementation was discarded but a new one HTTP based was fastly put to work presenting no bigger challenge. However still one major problem came up during the development, related to the user's media files communication.

The first approach cared only with saving the files in disk and forwarding them to the user, with a similar process to the text based requests. However this approach was basically loading to memory every single file that arrived to the server. As soon as some tests were made sending big enough files, the server started to crash, lacking available memory. Once realised, the author started to question how such process was implemented and started searching for an alternative. This problem was solved with the implementation of dedicated servlets. Instead of loading everything to memory as a whole, files started to be sent and read in small chunks, keeping the used memory in low values. This way, tests were re-run and no more problems were found related to this matter. With the exception of these two issues, no other implementation challenge cause major trouble and the implementation finished successfully. The final product can be seen in the following section.

7.1 Web Portal

As can be seen in the Project Management chapter, the development phase started by making the changes in the DB and by building the Web Portal. The website was built using the Struts 2 framework and it was carefully taken in consideration the objective of it. Considering that people who access it are most likely in a situation of panic, looking at all chances to recover their device, simplicity is even more important than usual. People must spend the lowest time possible learning to use it so they can start taking advantage of it the soonest possible as time is essential here.

Once the user logs in, he arrives at the home page, seen in figure 7.1.

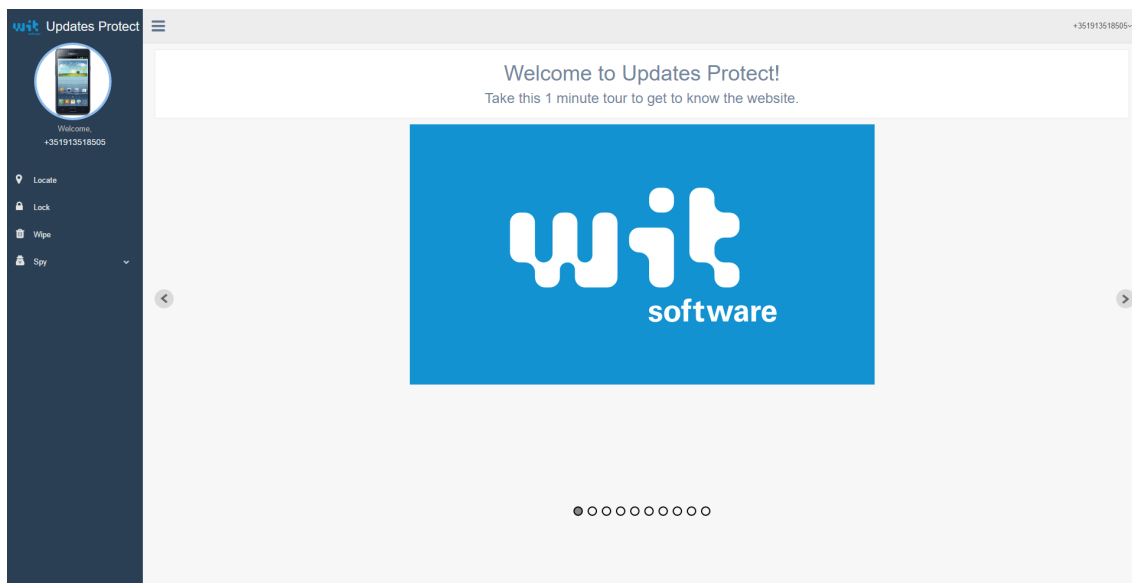


Figure 7.1: Web Portal Homepage

The main structure of the website is composed of a left side menu, which contains the Updates Protect logo, link to the homepage, the buttons to the feature menus and an icon of the associated account's device. On the top of the page, there is a bar with the option to logout.

The content of the homepage is an onboard screen, a set of images that show the essential parts of the website to get the user familiar with it. Again, as time is important, the onboard takes 1 minute to read the captions and retain the main components of the website, which is also stated in the first caption. It has 9 images and their captions which relate to:

1. Locate menu - The menu from which the user can request his device's location;
2. Lock menu - Menu where the screen of the device can be locked with an additional message and alarm;
3. Wipe menu - Menu where the ability of wiping data, apps or doing a factory reset can be done;
4. Calls menu - The menu from which the device's calls can be retrieved and presented;
5. Message calls - Same of above but referring to messages on the device. Displays SMS and MMS;

6. Browser History - Menu where the browsed webpages from the device can be requested;
7. Gallery and Camera - The first is where the device's gallery can be retrieved and the second where the user can use the cameras from the device;
8. Device and Network - Final screen where the device's info is displayed as well as the wifi's around it and its current network connection;
9. Previous requests - In all menus, previous successful requests are clickable and most of them can be expanded.

Starting with the first menu, the Locate one can be seen in figure 7.2:

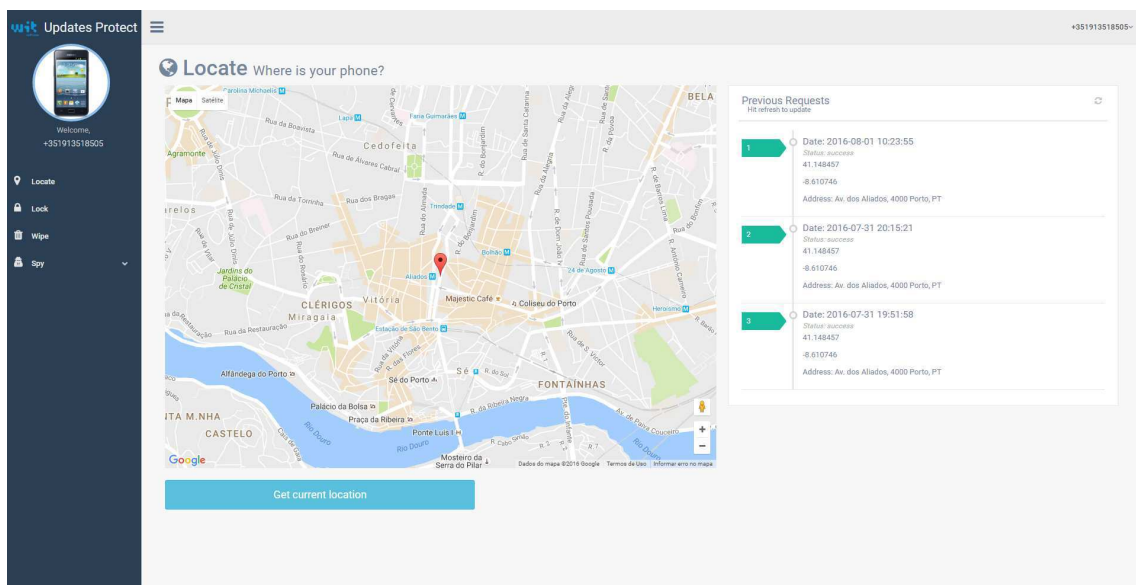


Figure 7.2: Web Portal Locate menu

As stated in the onboard, all menus with no exception contain a section, always on the right side to keep the menus very equal, which relates to previous successful requests. All of them are structurally equal, refreshable and their content expandable, with the exception of the Lock and Wipe menus, which are already expanded. Also, all buttons and boxes have allusive names and sometimes short explanations of their functions.

So, this menu has the obvious map to see the request's associated location, a button to request the current location and on the right side the previous requests, containing the date of request, its coordinates and address as well as its clickable number which loads to the map the location with a pin point to the spot.

The next menu is the Lock one, where the user can request to lock its device, seen in figure 7.3.

On the right side, can be found the previous requests, now with the requests' content already expanded stating if the lock defined any password to lock the screen, displayed any message and sounded the alarm. The middle box gives the chance for the user to set the message and trigger alarm on the device. On the left side, there is the image of the device. The purpose of it is to display the last password set: when the device has no password set, the app will set one and obviously it must be shown to the user so it can know it and unlock the device when he recovers it. So, clicking the device, it will show

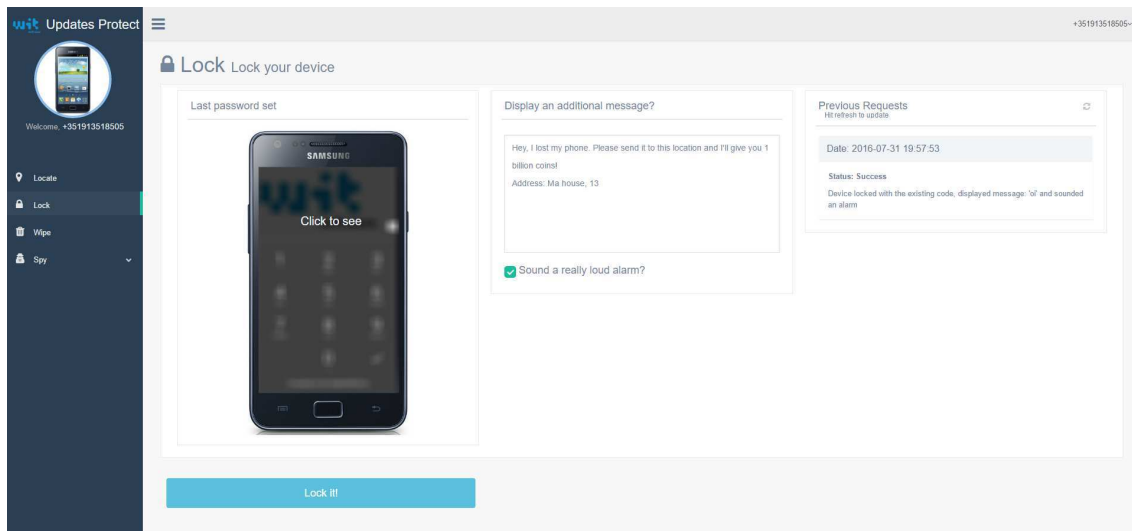


Figure 7.3: Web Portal Lock menu

the last password set and hide it once clicked again.

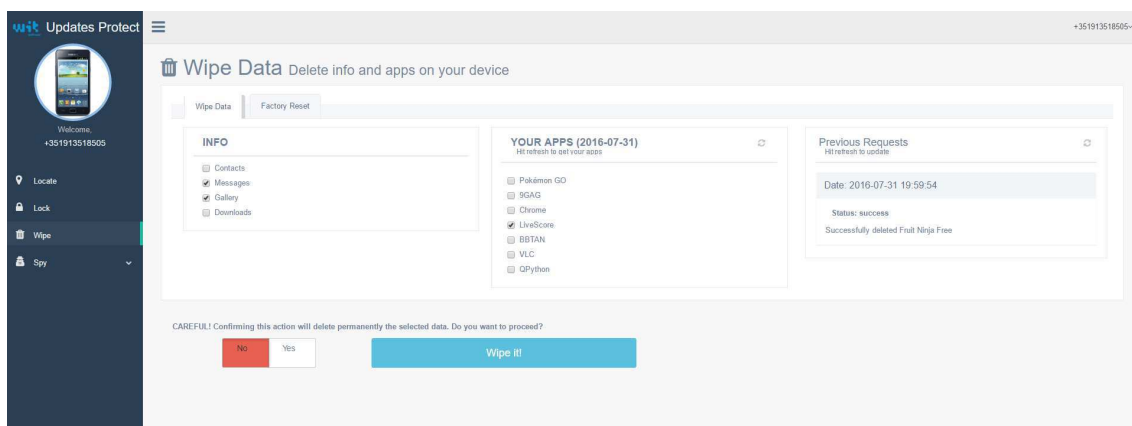


Figure 7.4: Web Portal Wipe Data menu

On figure 7.4 can be seen the Wipe menu. This menu following the previous one, has the previous requests section already expanded and showing what has been deleted previously. The first two sections show what can be deleted starting with the personal data such as the contacts, messages, gallery or downloads, and also the device installed apps. These apps must be requested when the user accesses the website for the first time, having a short explanation of this procedure on that section instead of the apps seen in the figure.

All these info is related to the wiping of data. If the user wants to do a factory reset, he can switch tab which leads to the disappearance of the two sections to be shown a warning screen, as there is no input needed to perform it. Either way, wiping data or performing a factory reset, the user must confirm that he understands what he's doing, and only after having confirmed the checkbox, he can request the wipe through the button.

The next menu is also the first from the spy features. Its the Calls menu, where the user can get all calls from the device, presented in figure 7.6.

This menu as usual has the previous requests section, now with the need to be clicked to see its content. The figure shows the post request screen, where the request result was

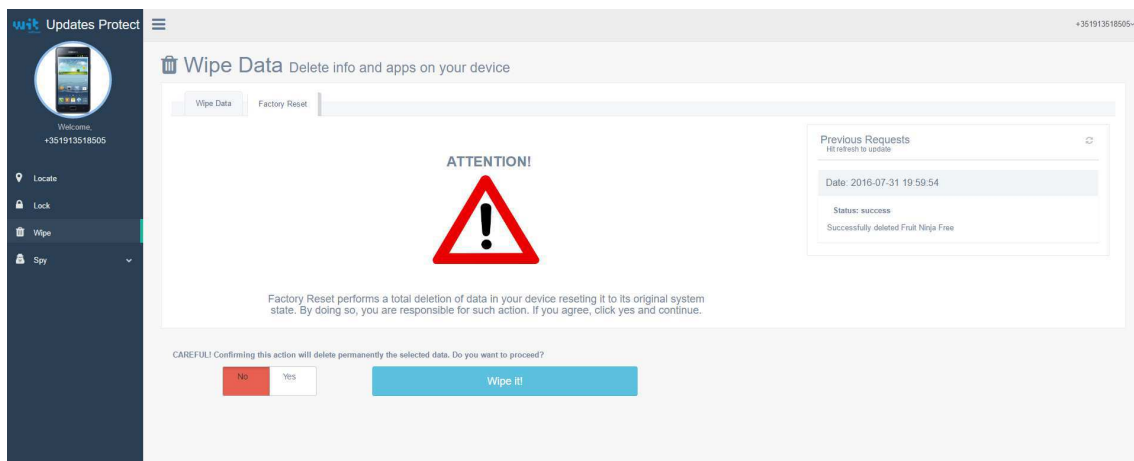


Figure 7.5: Web Portal Wipe Factory menu

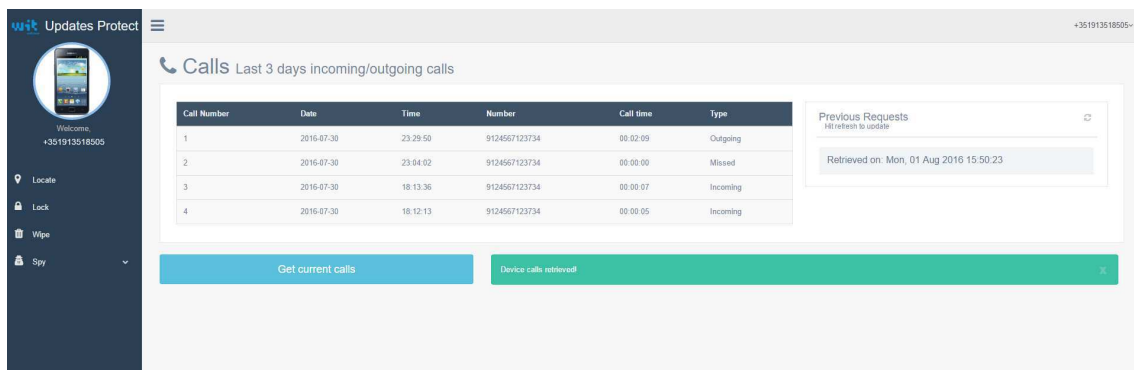


Figure 7.6: Web Portal Calls menu

loaded automatically to the table, section on the left and the green successful notification shown below as a request was well succeeded. All spy menus show the content of the device from the last 3 days.

The table on the left shows the calls details of the user's interest: first column shows the number of the call, the second column, the date of call and after it the time of the day, the number which the call was related to, next to it the duration and finally, the type which can be outgoing, incoming and missed.

The second spy menu is related to the SMS and MMS retrieval.

This menu has a different content layout as it can also have images. So, besides the previous requests section, the left box with the content shows with a green color the type of message which can be of type sent or inbox, and then shows the number which sent or received the message, the message's date and its content. When the message is a MMS, it can be zoomed when clicked.

The following menu is regarding the internet history.

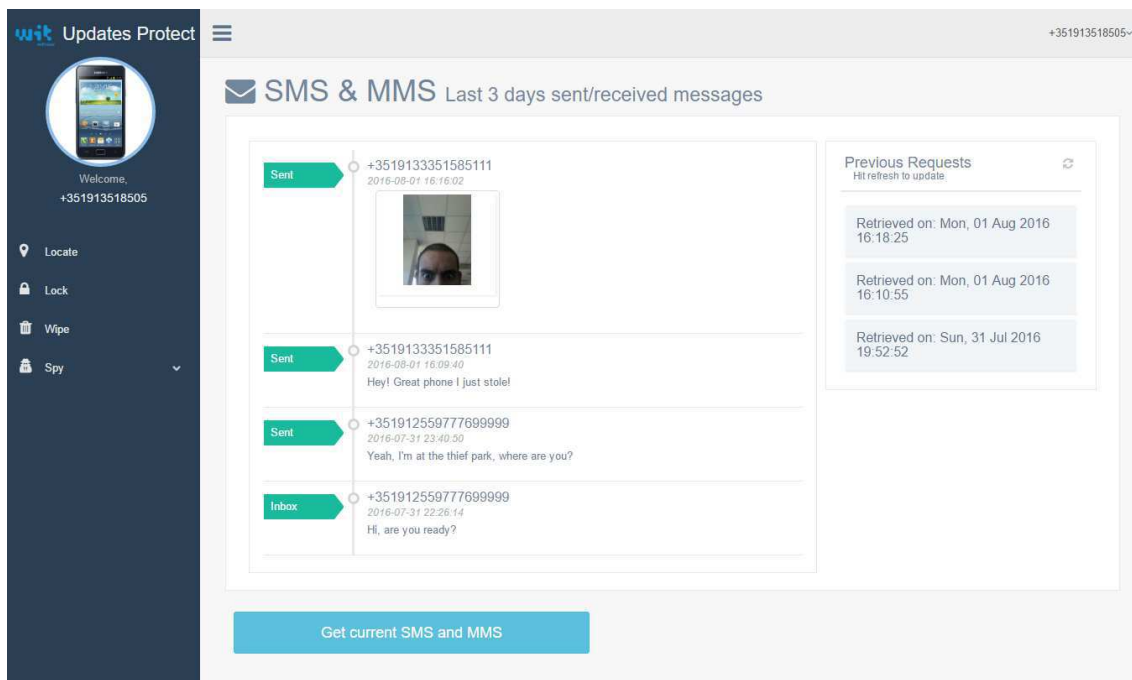


Figure 7.7: Web Portal Messages menu

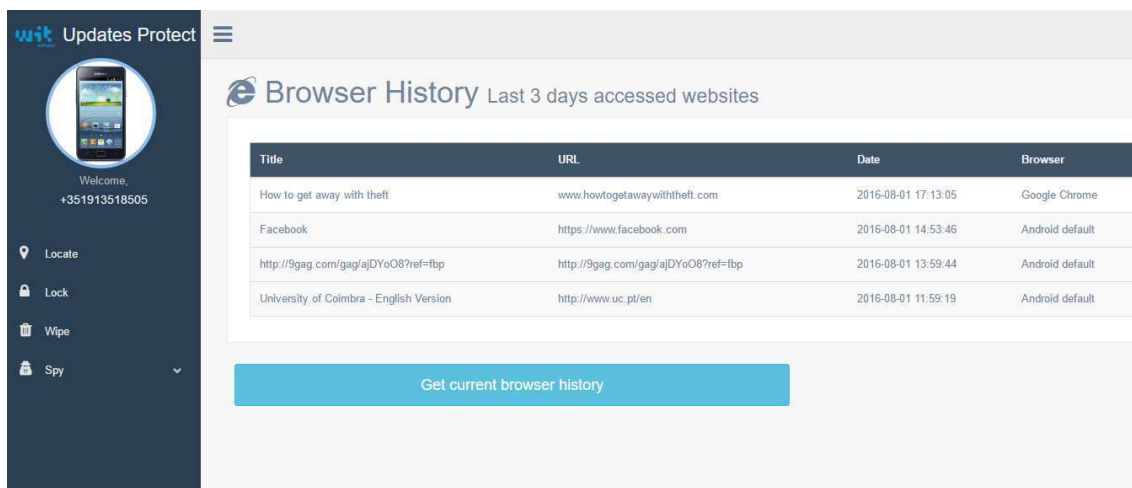


Figure 7.8: Web Portal Browser menu

Very similar to the Calls menu, this one changes its table content. This version shows the title of the webpage visited, the URL of it, the date when it was accessed and also the browser which was used to visit it.

Figure 7.9 shows the Gallery menu, the first related to the camera usage.

This menu presents the list of taken pictures and recorded videos on the device. These are presented in the left section box, presenting a div for each image and video, all clickable, images to be zoom, video to start playing and possibly to go fullscreen.

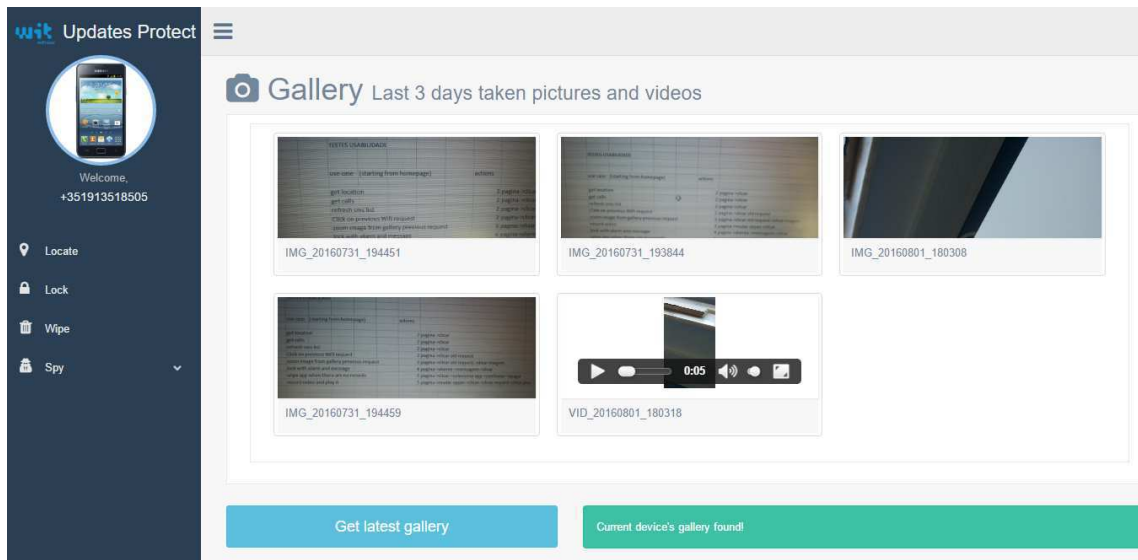


Figure 7.9: Web Portal Gallery menu

Very similar to it is the Camera menu, seen in figure 7.10.

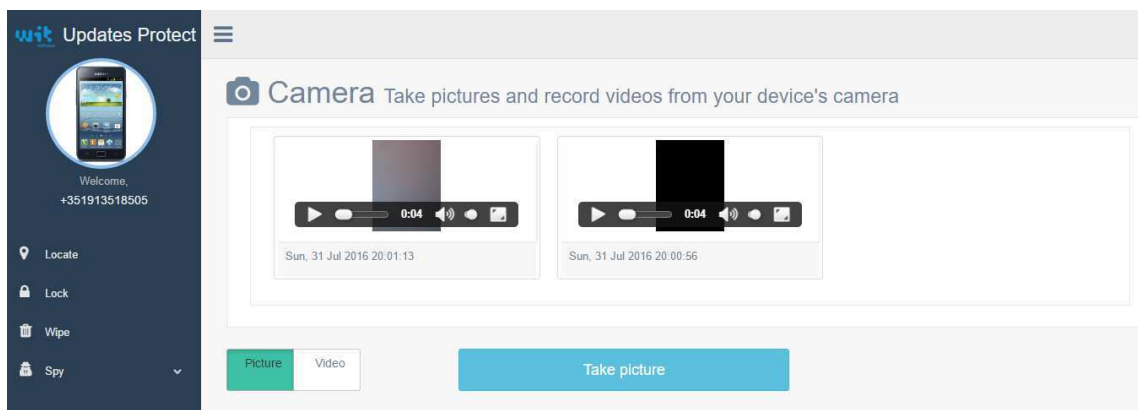


Figure 7.10: Web Portal Camera menu

The unique difference between the two menus is the existence of a switch button to explicit if the user wants to take a picture or record videos. Once the request is finished, the result is limited to a file per camera in the device contrary to the gallery which has no limit.

Finally, the last menu available to the user is the Device and Network.

This final menu shows the device info regarding the user's account: the device brand and model, its IMEI and MSISDN, his phone number. Also is where the user can request to see the available network connections to the device. Upon request by the user, it is shown in the middle section of the page the current connection of the device which may be Wi-Fi, 3G and other types of data connection, and below that, there is the available Wi-Fis for the device, sorted by their strength and displaying also its name or SSID.

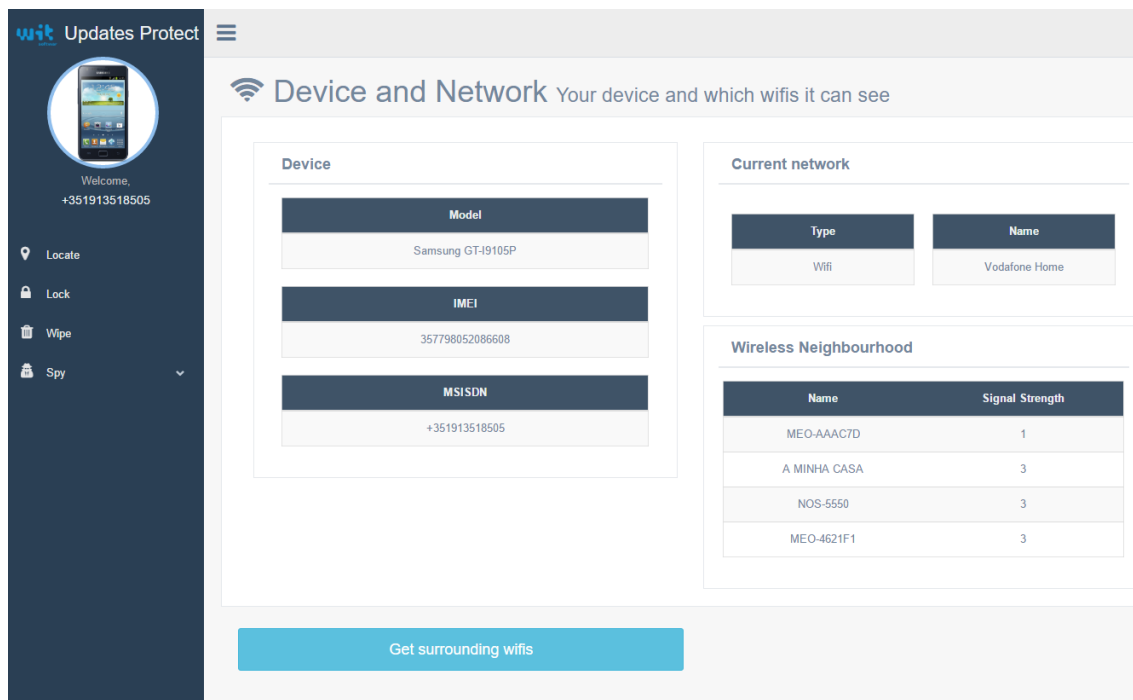


Figure 7.11: Web Portal Network menu

Mobile version

As it was a requirement, the Web Portal is completely responsive, adjusting its contents to the size of the screen, maintaining everything understandable and readable without losing features. Some issues came up due to that as the responsiveness of a website sometimes isn't easy to achieve. For instance, the tables created to show the calls and browser menu aren't suited to mobile versions, which lead to the creation of different styling layouts for smaller screens, visible in figure 7.12. When the screen is low enough while browsing on a desktop version, the left side-bar hides itself to give its space for the content div. When in mobile version, it hides automatically, being available to be used by clicking its button on the top right. The rest of the content, adjusts itself, moving in the screen properly, as seen in figure 7.14.

What the figure 7.12 also shows is the creation of two buttons to scroll the page. Easily some of these menus can get enormous, containing a lot of information to show. In order to get the best user experience, such pages need to have at least one of these buttons so the user can fast scroll up or down the page dodging the trouble of doing it himself. Usually, only the fast scroll up is presented, which typically leads to the menus, but this time, as the user to request some action on the device had to click the button on the bottom of the page, the opposite of that button was also created. Finally, the figure 7.13 shows how the zoomed picture adjusts its size to the screen resolution.

With all these differences, the Web Portal ended up with a friendly user interface in both web and mobile platforms, having some details to provide confort in user experience.

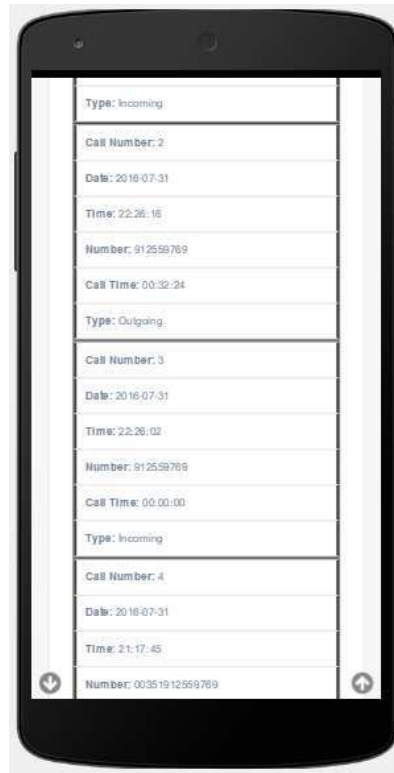


Figure 7.12: Calls menu content mobile version

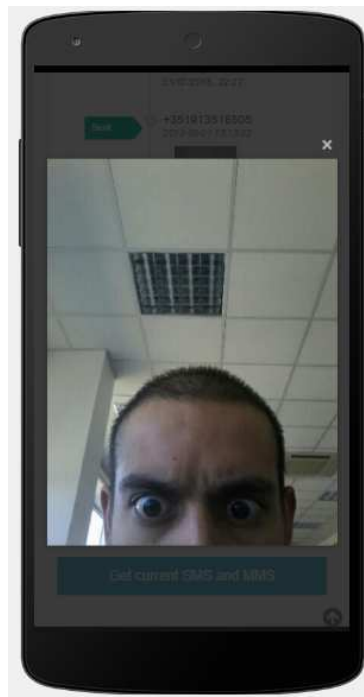


Figure 7.13: Mobile version image zoom

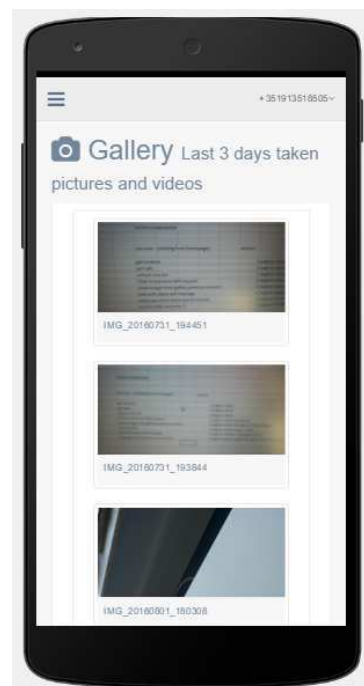


Figure 7.14: Mobile version responsiveness

7.2 Android App

In the other end of the solution lies the Android app of the Updates Protect. This app has only two types of interactions: the first one when the user turns on the device for the first time and the registration process starts. The second type is when the user requests the device to be locked which leads to the presentation of a lock screen to the person who has the device at the time.

The registration process begins when the device is turned on and the app initiates and checks for a registration and finds none. Once this is realised, the first android layout is presented to the user, which can be seen in figure 7.15. This screen introduces the app to the user, stating its purpose and what the user can benefit using it. It only has one path, which is achievable pressing the 'Next' button.

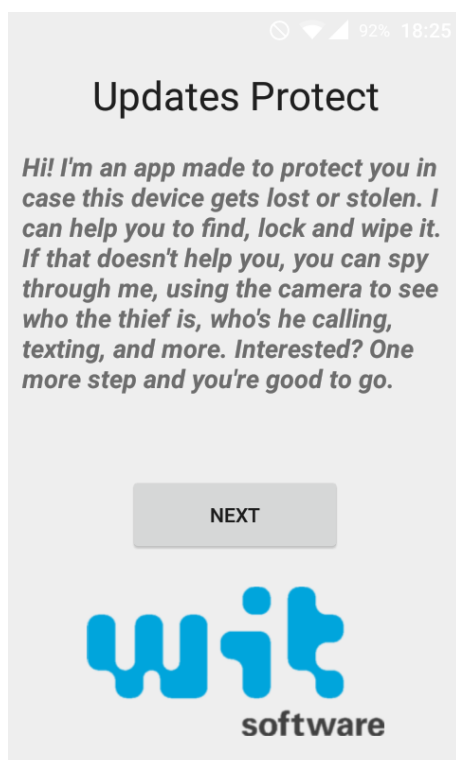


Figure 7.15: Welcome screen

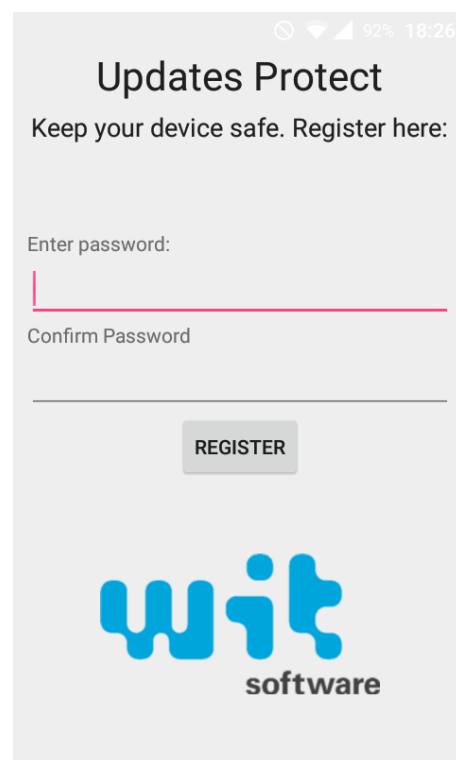


Figure 7.16: Registration screen

After the user skipped that screen, the registration screen comes up, seen in figure 7.16. Here the user is prompted to input a password and a confirmation of it.

No additional information is required because all the remaining info is gathered automatically by the app such as the IMEI, MSISDN, GCM registration id or IMSI.

After the password is introduced the user can only press the 'Register' button. Once done, the app will communicate to the server the new account data to be registered. Meanwhile, the user sees the confirmation screen, figure 7.17, informing him of the success of the operation, and reminding him that his username to login in the Web Portal is his mobile number or MSISDN.

After these customizable screens, come the permission requests. First, regarding the device administrator permission, seen in figure 7.18, which allows the app to wipe data, lock the screen and set passwords in the device.

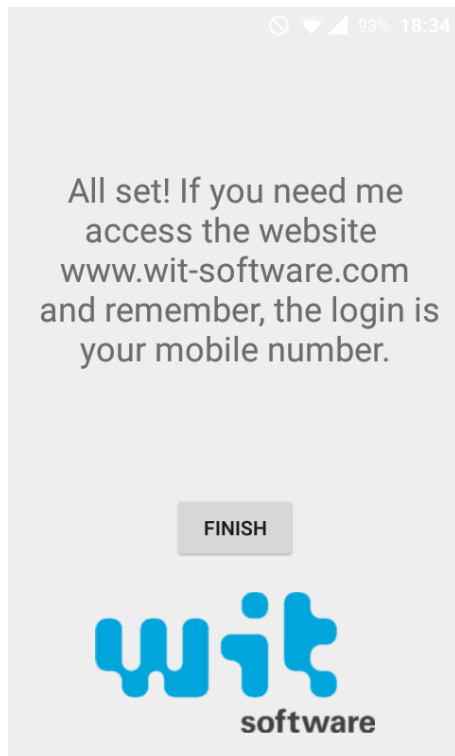


Figure 7.17: Confirmation screen

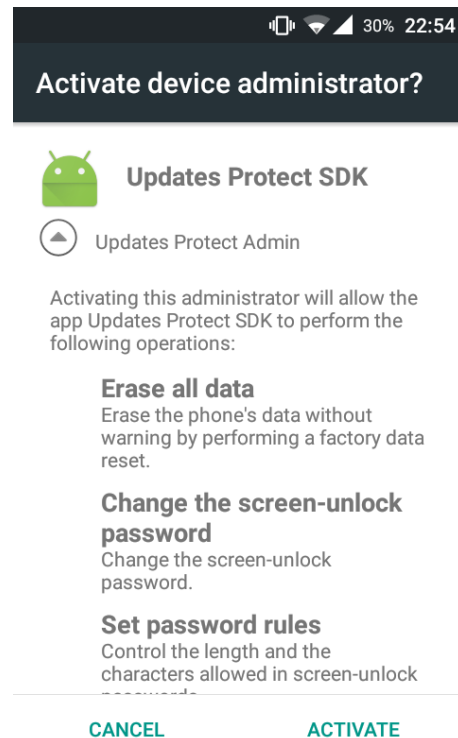


Figure 7.18: Admin Permission requesting

Chapter 8

Evaluation

Nowadays the number of Android app and solutions to solve daily problems is so big, that if a specific solution fails to meet the user expectations, ten seconds may be enough for him to uninstall and get rid of it, as he also in a very short amount of time can get another solution that does the exact same thing, available on Play Store. People expect quality, people demand solutions that work at the speed of light and nobody tolerates failure or random crashes. Fortunately, this area is improving itself, and the concern companies take with software quality is continuously growing. This project is no exception and proof of that is the amount of time and effort spent during this year, testing the solution to guarantee its excellence.

As this project is composed by several components, all these must be taken into account as they can be weak spots which make the system fail. These components, which are the Web Portal, Updates Server, Google Cloud Messaging server and the Android app, must all be tested but in different ways as they take different roles in the solution, communicate through specific ways and have separate purposes.

This chapter reflects the work done to test the developed solution how it was done and the outcoming result. The tests that were done can be divided into functional tests, described in the first section, and non-functional tests, described in the second section.

8.1 Functional Testing

Software projects start with a set of objectives which later become requirements, as it also happened with the present project. These requirements define what the solution should become and are the starting guide for the development. Functional Testing is a quality assurance technique for testing software solutions in various levels to test its functionalities in order to avoid failures. Its type of testing is of black box type, which means that all test cases are created based on the project specifications and with no knowledge of the implemented code. This way, these tests focus on what the the solution does, caring with the inputs it takes and the outputs it gives. Tests have also different levels, being the major ones:

- Unit Tests - Lowest kind of tests, verifies the proper execution of a functionality or code;
- Integration Tests - On a slightly higher level, they test how different units or components integrate with each other;
- System Tests - After the the development is finished, system tests relate to a comparison with the requirements, checking if the system meets meets them;

- Acceptance Tests - As they are the last kind of tests, they validate if the product is what the client expected.

During the first semester unit testing, integration testing and system testing were performed in an undocumented way after the implementation of certain modules. Once the second semester and the development of the android app started, features started to be completed and acceptance tests were done too.

As the month of June was almost fully spent testing the system, a plan was created to document and perform them properly. The tests were requirement-driven or specification-based and for each component, different tests were designed trying to verify:

1. Web Portal

- How the information was displayed - if it is well-formatted and understandable;
- If the displayed information was correct - if it is the same as the results gathered by the device;
- If the state of the requests were always returned - returning info or an error user-understandable, in case of crash in any component, if the user get always a message;
- The actions were intuitive - naming of content is explicit and buttons easy to understand or have a caption;
- The website responsiveness - if it work on different browsers, dimensions and platforms.

2. Server

- The proper work of the REST interface - receiving user requests and sending their results;
- Correct login and action requesting from user - user actions are dealt properly;
- Verify the info requests results - Does it return the correct info;
- Communication with the Web Portal, GCM server and Android app - does the information flow properly;
- Process changes were properly saved in the DB - does every change persist;

3. Android app

- Communication with the Server;
- Communication with the GCM servers,
- Proper execution of all features;
- Behaviour of the app was the expected regarding the battery and resources usage;
- Every change on the requests handling was always persisted, in case of device shutdown.

The GCM server had the only purpose of forwarding the messages to the specific devices and the only possible tests made to it were some benchmarking tests, done to the whole system and described in the next section. As the server was already built and the author expanded it, the tests were focused on the parts he built, how they communicated

with each other and the external components and how they persisted information. Regarding the Web Portal and the android app, things were more complex.

The Web Portal due to its interaction with the user had to be tested in different platforms to see its outcome. For that matter, it was chosen that the tests should be executed in 3 different browsers: Google Chrome, Mozilla Firefox and Microsoft Edge. This way, it was believed that the most different behaviours would be covered and most possible bugs found.

Regarding the android app and its need to execute properly on all devices, it was top priority to test all features in the maximum number of different devices. As this app will be released as a system app, it was considered that would be the best to test it in the latest versions of Android, as the probability of being the ones running on new devices is higher. So the devices on which the tests were executed were:

- LG Nexus 5 running Android Marshmallow 6.0;
- Samsung Galaxy S2 running Android Lollipop 5.1;
- Samsung Tab S running Android Lollipop 5.0;
- Sony Xperia E1 running Android Kitkat 4.4.2;
- LG L40 running Android Kitkat 4.4.2;
- Sony Xperia Z running Android JellyBean 4.1.2.

As the priority was to guarantee the proper execution with more focus on the most recent Android versions, more tests were performed on the first two devices followed by the Sony Xperia E1. When it comes to the browsers used to test, the following platforms used were:

- Google Chrome 51
- Mozilla Firefox 46.0.1
- Microsoft Edge 13
- Blisk 0.60

With everything set up, the tests were made and the results visible in table 8.1 were recorded:

	First run		Final run	
Feature	Failed	Passed	Failed	Passed
Manage Account	1	4	0	5
Locate	0	3	0	3
Lock	2	3	0	5
Wipe	1	4	0	5
Calls	0	3	0	3
Messages	0	4	0	4
Internet	0	3	0	3
Gallery	0	5	0	5
Camera	1	5	0	6
Network	0	3	0	3

Table 8.1: Functional Testing Results

All tests were manual and executed by the author. These test cases which resulted in hundreds of executions started with some failed tests being the majority of them related to the Web Portal requirements and the rest related to the execution of the features by the android app. The testing phase went well and all the bugs were fixed which can be seen in the last run statistics, which resulted in a 100% success rate.

For more in-depth information, Appendix E - Evaluation shows all the test cases executed as well as their results and comments.

8.2 Non-Functional Testing

In opposition to first section which is based on the business requirements, the Non-Functional Testing focus on the client expectations and its view of the final product. As mentioned before, its expectations aren't if the requirement A or B are fulfilled but for example if the website is fast doing its job and gives feedback, doesn't crash on any user bad input and is pretty. So the tests made in this section are a bit distinct. First of all, some tests were considered to be done but the structure of the project made them irrelevant. That is the case of security and load tests. Security tests were considered for the obvious reason of the need of making the system secure avoiding intrusions and malicious actions, but as the whole system is intended to be integrated inside a Vodafone network, already secure and validated by a certified inspector company. Also, load tests were considered and discarded. These tests are important to realise how the system handles huge amounts of requests, helping realise if there are bottlenecks and if so, which are they. As they are important, though not crucial to this project, they were discarded as the Updates server has already been working for a few years and this kind of tests were already performed, so they would bring no relevant information. So, the performed non-functional tests made to this project were performance relative to the new flow of media files and usability tests.

8.2.1 Usability testing

Usability tests are very common in software projects and consist of getting a set of users with varying background knowledge of the project concept and asking them to execute some pre-defined tasks in order to evaluate how good their interaction with the product is. This can be used in multiple ways but in this case, as the major interaction the user has with the system is through the website, it was the kind of interaction tested. These kind of tests are crucial to fulfill 2 major concerns. Firstly, they help to verify if some requirements are being achieved, more related to the interaction with the end user. Secondly, they reveal a lot of information about the interaction people will have with the developed product. The reason for this being so important is easy to perceive, as many products fail to take position in their markets because the users felt confused about what they could do with the product and how they could achieve some goal, and this way the best available product to the user might be discarded by him for not being easily usable and understandable.

So the task here was to present the Web Portal to someone and test its usability. The process is not so straight-forward as some preparation was needed with represent to creation of which tasks were to be evaluated and which metrics would evaluate them. As it would be unreasonable to force some user to test every existing requirement, the defined tasks had to be each one a flow of actions needed to fulfill a requirement which process is very similar to other requirement's processes. This way, when the user executes a task representative of a requirement, he is learning a process quite similar to processes of other

requirements and avoiding repetitive tasks. Once these tasks were created, this set of tasks covered the major paths the user could take in the Web Portal. So, the created use cases were:

1. UC_1 - Get the current device location
2. UC_2 - Refresh the previous requests of the device's messages
3. UC_3 - Get the network to which the device was connected in the last request made
4. UC_4 - Zoom an image from the device gallery of the last request made
5. UC_5 - Lock the device with the message "Hi" and sounding an alarm
6. UC_6 - Wipe any existing app on the device without knowing which apps are installed
7. UC_7 - Request the device to record a video and play the result

These created use cases had some pre-conditions that had to be accomplished before the user could start them. Relating to every test, a testing device with available cameras had to be registered in the system, and the Web Portal had to be logged in that account beforehand so that the requests made during the testing session were intended to that device which had to be connected to the internet and had to have given all permissions to the app. Then, relating to singular tests there were other pre-conditions. To perform the task 2, 3 and 4, previous requests had to be made so the target info was available. To perform the task 6, some apps had to be installed so one of them could be deleted.

These tasks would mean nothing if there were no way to evaluate them. Firstly, it was verified if the user achieved the proposed goal of the task. Then, how hard it was to complete the process, successfully or not, so the number of actions he had to take to get to the final result were counted and compared to an optimal number. Finally, the amount of time he took during the whole process. Unfortunately, no system is perfect and the GCM is no exception. As most of the times the biggest amount of waiting time in the website is dedicated to waiting for the GCM message to arrive to the device, these intervals when the user was waiting and the system "idle" waiting for that message, were not counted. So, the final task times aren't absolutely synchronized and precise, although they are very approximate to the real times.

The users selected to test these requirements were from Wit Software to respect the company's privacy policy. They were given some background high-level information regarding the project and no access to the onboard available in the homepage of the Web Portal. This way, it was tested the most realistic worst case scenario, when a user knows the minimum about this solution and gets to the website in a hurry to get the device back and most likely, won't pay attention to the learning process of the onboard.

So, finally these use cases were given to the users and their interaction recorded. The results are presented in the following charts. All tests were passed successfully, but for each of them, the actions needed and time taken to achieve success were different. So, each chart shows for the different users how many actions they took in the vertical bars, and shown by a green line, the time they took to accomplish it. The last bar, shows the average for both indicators.

Use Case 1

The first use case is representative of requesting the current device's location. To achieve this, two steps had to be taken:

- Access the Location menu
- Click the request location button

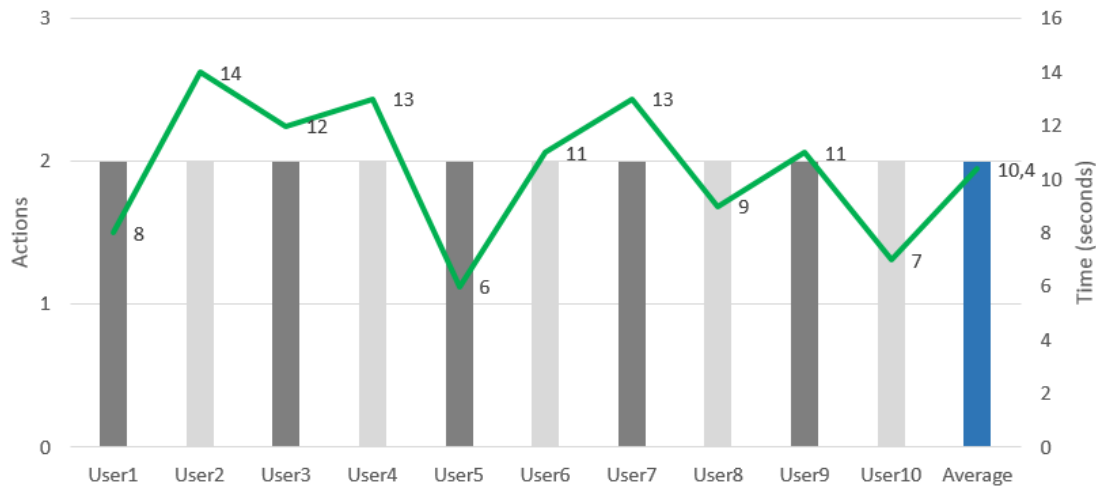


Figure 8.1: Actions and time taken for Use Case 1

This was the first and simplest use case and the users had no problem to achieve his success. All of them needed only to take the minimum possible amount of actions and the time to do it was short with an average of 10.4 seconds, so can be concluded that the results were quite satisfactory.

Use Case 2

The second use case represents the feature of refreshing the section of previous requests regarding the device's messages. This is meant to test how easily users could associate the refresh button and its symbol with its action. The needed steps to do this were:

- Access the messages menu
- Click the refresh previous messages button

The results of this second use case were not so optimal compared to the first one. Here, the time taken to visualize and understand the page content was higher with an average of 14.4, 4 seconds higher comparing with the first use case. This is easily explained because the objective button in UC_1 was the most important, bigger and visible one. In this case, the refresh button couldn't take the page attention being smaller and harder to find. During this testing session, it was noticeable that some users many times didn't read most of the text content of the page. Here as the button was harder to be located and understood, users were forced to stop and read. Even so, some of them didn't read and the whole content and were tempted to press the most appealing button, to request the current messages in the device. This was done by the 3 users with 3 actions. Only after doing such mistake they realised it, and then found the objective button. Even though, this use case was considered successful as everyone achieved the objective with small taken times.

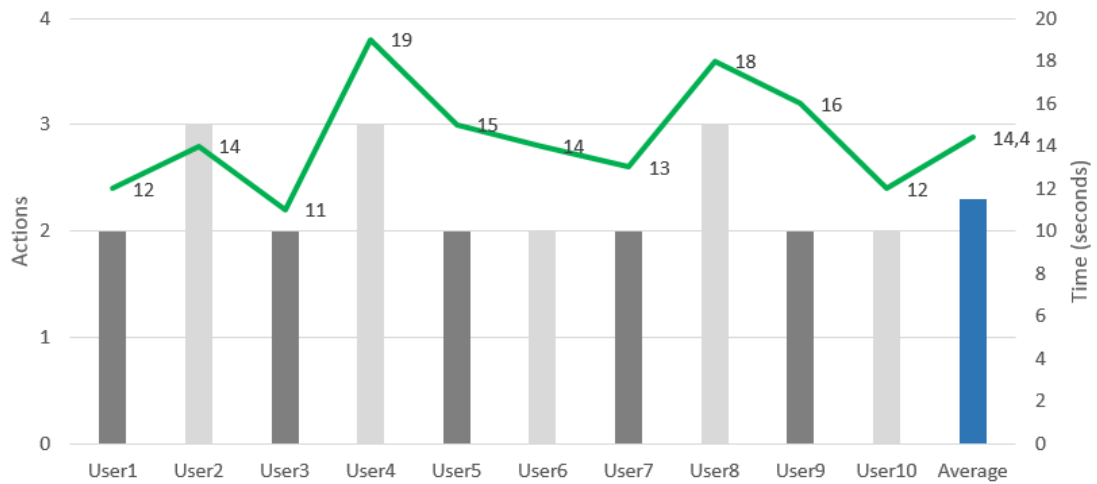


Figure 8.2: Actions and time taken for Use Case 2

Use Case 3

The third use case's objective was to get the network to which the device was connected in the last request made by the user. This use case needed the following actions:

- Access the Device and Network menu
- Click on the last request
- State the connected network's name

Its results are presented in figure 8.3:

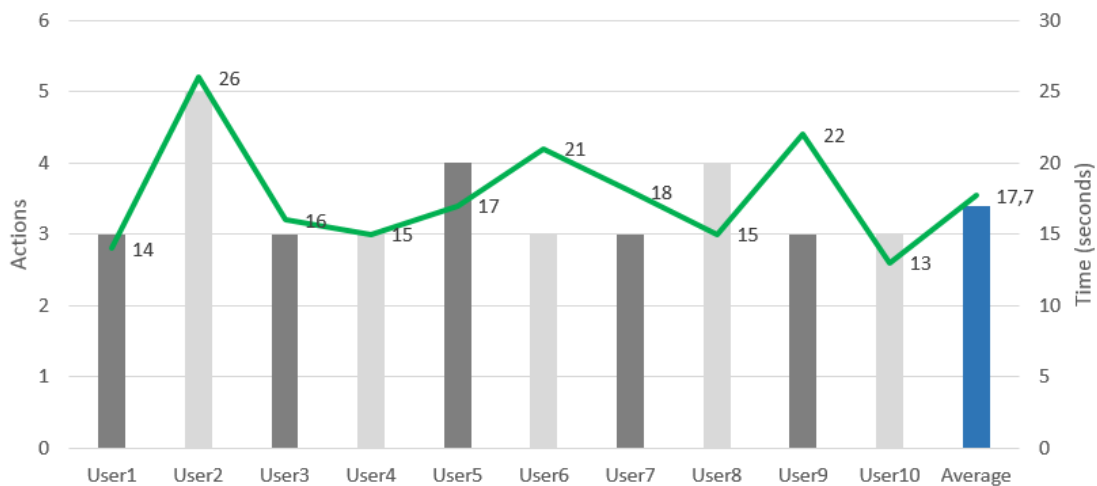


Figure 8.3: Actions and time taken for Use Case 3

The successful results here were a bit harder to achieve as some users got a little confused. Firstly, not a single user had trouble to get to the proper menu, as the name was the most suggesting and related to Wi-Fis. Unfortunately, upon facing the menu, one user was confused by seeing part of the content relating to the device information such as the IMEI or device model and thought he was on the wrong place and after checking the menus' names his next action was to click the dropdown button on the top right corner of the screen. As the only option from it is to logout he then looked again Device and

Network menu, from which he never left. Then, he clicked in the right previous request but stated as the objective network's name not the right one which was in a separate box, but the first Wi-Fi network in the available Wi-Fis list. After analyzing better, he noticed the mistake and reached the proper one, stating it as the right answer. This mistake was also reproduced by 2 other users which brought into discussion if the information was well displayed. Even though the implementation was believed to be easily understood and the best, it was suggested that these two sections could be joined avoiding the obligation of the user to look into two boxes, and if the device was connected to one Wi-Fi, its name could be from a different color. However, this solution would have to take into consideration that the device could be connected to the internet via 3G for example.

To conclude, the number of actions taken in this use case was not so good as 3 users needed more steps than the minimum, and the time taken was higher with an average of 17.7 seconds.

Use Case 4

In use case 4, the user had to zoom an image from the device gallery of the last request made. Has two menus had multimedia content, pictures and videos, the interaction between them and the users had to be tested to see if they were easy to handle. So, the needed steps to pass this test were:

- Access the Gallery menu
- Click on the last request
- Click an image to zoom it

The actions were simple and bring no major innovation, being the new part related to opening the image which was expected to be easily understood and executed by the users as most of this kind of interaction online has the same behaviour. So, the results for these tests were:

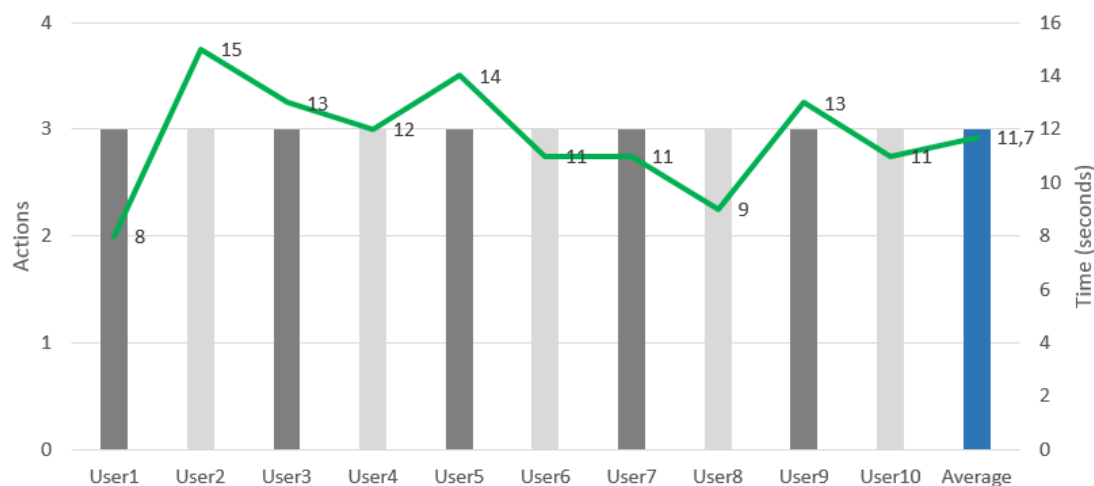


Figure 8.4: Actions and time taken for Use Case 4

As can be seen in figure 8.4, the users had no trouble figuring out how to achieve success here. The steps were previously taken on earlier tasks and everyone realised that to zoom the image, it had to be clicked. So, all users needed the minimum amount of actions to execute this test and the time taken was low, with an average of 11.7 seconds.

Use Case 5

The fifth use case was related to request to lock the device with the an additional message and sounding an alarm. To reproduce it, 4 steps were needed:

- Access the Lock menu
- Insert the message to be displayed
- Check the alarm checkbox
- Click the button to lock the device

This was one of the most important requirements of this project and its menu interaction with the audience quite unknown because it had many sections claiming for the user's attention. So, here it was more important the user's behaviour and focus than his performance as statistics. The results are presented in figure 8.5:

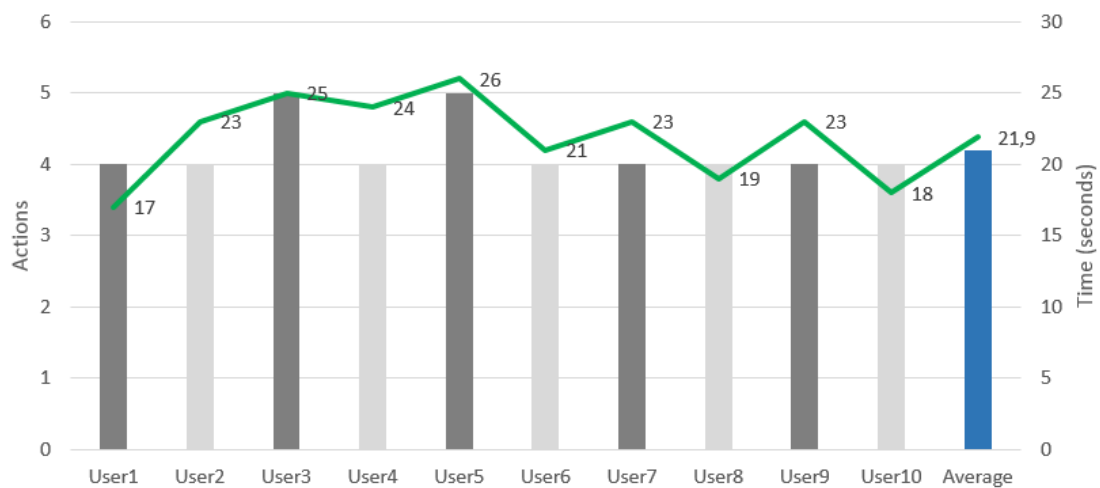


Figure 8.5: Actions and time taken for Use Case 5

After finishing the tests for this use case the results brought mixed feelings. In one hand, there were the statistics of actions taken showing good results. By the actions needed in this task, it can be seen that the users had no trouble achieving success with the minimum actions needed, with the exception of user 3 and 5. They committed the same mistake, being misled by the phone image which hides the last password set in the device, clicking it instead of the lock button. In the other hand, and as it was predicted, the time taken to execute the actions was the highest so far. This was the best indicator alongside the two mistakes made by the user 3 and 5 for the overload of information in this screen. This overload, later confirmed by some users, was prejudicial to the best flow of actions and typically lead people to try to understand all the available kinds of interactions, mainly the device's image.

So, despite being a functional menu and its interaction with the users worked out well, the average time of 21.9 seconds taken to execute the task indicates well that this menu could change the focus of each step to keep the user attention on what matters at a specific time.

Use Case 6

Despite not being the last use case, this was the most complex one, as it represented the one with more varying steps. This use case related to the wipe of any existing app on the

device without the system knowing which apps were installed on the device previously. The steps to execute it were complex, as follows:

- Access the Wipe menu
- Request to get the installed apps
- Select any app
- Confirm wipe clicking the checkbox
- Request wipe clicking the wipe button

Such as the previous one, this use case of one of the most important ones and also the most complex because it forced the user to request the system to get the installed apps on the device before requesting the wipe, if that was the user's objective and so, once the user accessed the right menu the section related to the installed apps would be empty containing a message suggesting the user to click the button to request the apps. Also, as the user could perform a factory reset as an alternative to the regular wipe, another misleading option existed. Finally, the necessity of confirming the wipe action clicking the checkbox button was likely to be missed by some users.

So, the results for this test were:

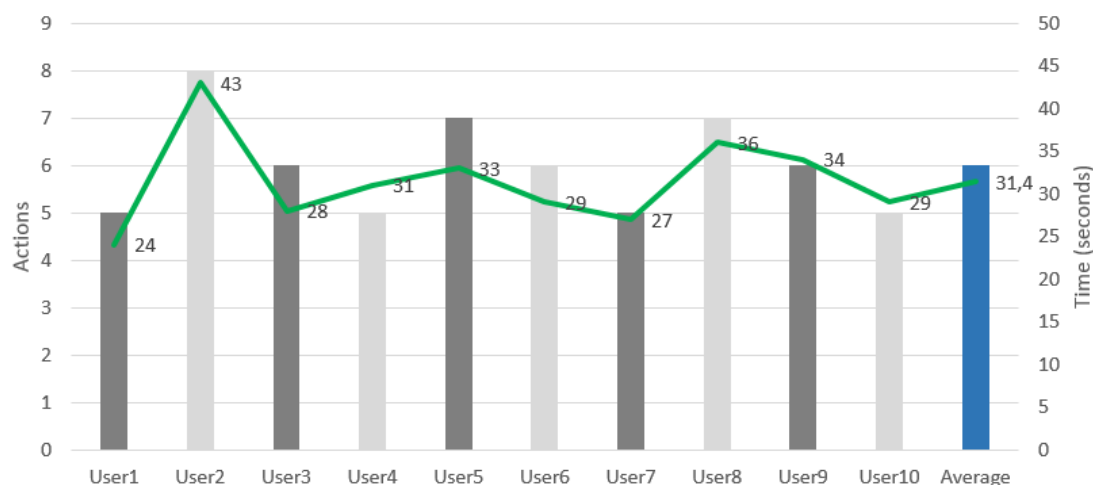


Figure 8.6: Actions and time taken for Use Case 6

The statistics presented for this use case were the most concerning. Starting with the actions needed to complete the task, only 3 out of 10 users needed the minimum amount of actions which were 5. Users 3 6 and 9 needed 6 actions, users 7 and 8 needed 7 actions and user 2 needed 8 actions. All these confusion can be divided into two different parts of the request: the installed apps and the confirmation of wiping. Relating to the installed apps, some users got confused about the missing information and proceed to click the wipe button right away, only after realising that they didn't confirm the wipe and also, that they were doing the wrong step to get the apps. Others tried to refresh the previous requests section. In addition to that, user 2 took the most actions as he was also tempted to go to the factory reset tab. Alongside the average time taken of 31.4 seconds and although everyone achieved the success result, it was clear that the amount of information and appealing areas confused the users so probably a menu more focused in certain information would benefit the usability of this use case.

Use Case 7

The final use case relates to the user to request the device to record a video, and the user to play the result. The main focus here was not the requesting process nor the playing the video as target objective. Here, the record time of each video was increased with the purpose to make the system take more than one minute between the time when the user requested the video and the time the video was presented to the user. This way, the users were forced to see the warning message displaying the info that the request took too long. This message also states that the system is still trying to record the video and the user should keep refreshing the previous requests section to check for its completion. So, this process was the major focus of this test, grouping a set of requirements and the minimum steps to achieve it were:

- Access the Camera menu
- Change the checkbox from Picture to Video
- Click the button to request the video
- Refresh the previous requests section
- Click the previous request made to see its result
- Click on the play button shown in the video to play it

Obviously, the fourth action of refreshing the previous requests section only counted when the request was finished. If the user decided to do other action besides that one, it would be counted as a mistake action. The results for this test are the following:

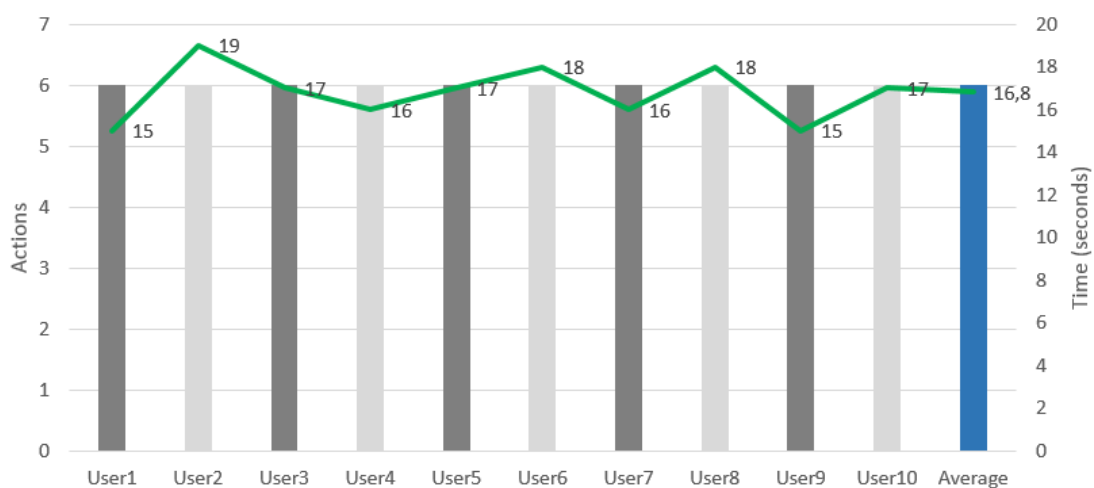


Figure 8.7: Actions and time taken for Use Case 7

As can be seen in figure 8.7, the results were good. All the users found no problem in requesting the video instead of picture and once the request took long enough, they read the warning message and started clicking furiously to refresh the previous requests. Here, once it was finished, they clicked the objective one and then as a natural process they clicked the play button and some even maximized it to full screen, so this test was a total success and the time needed to perform these actions had a small amplitude between the fastest and longest times, with an average of 16.8 seconds.

Conclusion

After finishing all usability tests an overview was made to come up with conclusions. All users during all tests reached the objective state of success which has to be highlighted. For that reason, can be said that these tests were a success and the Web Portal built in a decent way. However, it also as to be pointed out some difficulties which came up to some users on some tasks which lead to their need of more actions and time to get to the objective. These difficulties were not raised due to bad explanations, or misplaced features but instead due to the amount of information in some menus. This could be easily improved with some small changes in the design of the Web Portal, highlighting the places which would interest the user the most so he couldn't be distracted and confused by other information.

8.2.2 Performance Testing

By the middle of the second semester, some issues came up relating to the performance of the server, as many memory problems started to appear. Due to that, an additional servlet was created to send from the server to the Web Portal the multimedia files, the cause of such problem. Even with this improvement it was no cause for celebration and for reasons already stated, users expect this service to be fast and work perfectly, so, performance tests were design to test specifically the responsiveness of this system, measuring the time taken to fulfill some task.

These tests main focus was, as said, the multimedia files sending, which relates to majorly the Gallery and Camera menus, being the first menu easier to test. So, requests for the current state of the device's gallery were made and the time taken during each phase of the communication was recorded. The success criteria here was that these requests finished their execution and presented the images/videos within 1 minute carrying a different number of files, which made them different from each other. The test cases were to request the device's gallery with the:

1. Device's gallery containing 1 image
2. Device's gallery containing 5 images
3. Device's gallery containing 5 images and 1 video of 10 seconds of length
4. Device's gallery containing 2 videos of 1 minutes of length with black content

In order to maintain the best environment for these tests, no previous requests could exist, so that they wouldn't be loaded to the Web Portal, and also, the attributes of images and videos should be the same between requests, keeping variables equal between them as well as the files size. The images and had an average size of 1,70 MB. The 10 second videos had an average size of 20 MB and the 1 minutes videos an average size of 60 MB, recorded with black content. The results for these tests are available in table 8.2.

Performance Tests Results		
Test #	Test Result	Total Time (seconds)
Test Case 1	Passed	11,2
Test Case 2	Passed	45,2
Test Case 3	Passed	40,9
Test Case 4	Failed	70,7

Table 8.2: Non Functional Requirements

As can be seen in the performance tests results, even though the majority went fine and passed the test successfully, the last test failed, which was a bad indicator. As it also can be seen into more detail in the appropriate appendix, looking at the times taken in each section of the system, helps to take better conclusions out of this total time. First, the system has several sections/tasks that can be identified:

1. Web Portal requests the gallery to the Updates server
2. Server sends the request to the GCM system
3. GCM system communicates to the Android app
4. App executes the request
5. App sends the files to the Updates server
6. Updates server saves the files in its filesystem
7. Web Portal polling requests the files after successful polling
8. Server sends the files to the Web Portal
9. Web Portal displays the files

Table 8.3 shows the performance results for the 4 test cases, where it can be seen how long in seconds (approximate) each test took during action and the percentage of time it represented.

Performance Tests Results								
	Test1		Test2		Test3		Test4	
Action #	Time	%	Time	%	Time	%	Time	%
1	0	0	0	0	0	0	0	0
2	5	45,5	20	44,4	11	26,8	4	5,7
3	1	9,1	1	2,2	1	2,4	1	1,4
4	0	0	1	2,2	1	2,4	0	0
5	1	9,1	4	8,9	7	17,1	17	24,3
6	1	9,1	11	24,4	11	26,8	19	27,1
7	1	9,1	4	8,9	1	2,4	5	7,1
8	1	9,1	2	4,4	4	9,8	15	21,4
9	1	9,1	2	4,4	5	12,2	9	12,9

Table 8.3: Performance Times per action

These sections took different amounts of time and so conclusions can be taken. Firstly, the first and fourth sections always took between 0% to 2% of the time spent waiting as the Web Portal was fast to request and the app fast to execute the feature. Secondly, some time was wasted in the between of arriving the request to the server and its forwarding to the GCM system. Unfortunately, as the Updates server process of getting new requests from the DB can be slow sometimes using the default values, this was a negative factor in the waiting time although it could and was afterwards changed to be executed in smaller intervals. Also, there was an unpredictable section, related to the GCM system, number 3 on the list. Here, the percentage of time taken of the whole was very variable, taking percentages of time between 1 and 10% which were good values comparing to random tests results made during this year. This is a factor that the Updates system couldn't control and so, a lot of time wasted waiting could be avoided, using an alternative to this system, when its condition wasn't the best. Upon the execution of the feature, the files are sent to the servlet in the Updates server and written to filesystem. This multiple step was very inefficient and took very long with small files to fulfill its job. Yet without being so inefficient the sending of the files to the Web Portal, number 8 on the list could also be improved as consumed some time. In between, the polling from the Web Portal to the server were also irrelevant. With this can be concluded that the bottleneck was the transmission of files between both ends. The creation of the servlets improved a lot the performance in terms of memory but unfortunately, the responsiveness of the system which improved didn't get the best times. To fix this, a possible solution could be used a dedicated file server to increase the throughput keeping a low latency and avoiding the server's participation in this flow.

Chapter 9

Conclusions and Future Work

As the internship has come to an end and all work has been finished as well as this report, its time to take conclusions presenting a global overview of the past year and also presenting some thoughts regarding future work for this project. To do so, this chapter is divided into 3 sections: firstly, presented an overview of the internship objectives and its success criteria. Secondly, a small thought of what could be done to improve this project in the future. Finally, a personal opinion from the author regarding the whole year and its outcome.

Updates Protect Goals

Software engineering projects have many ways to evaluate success, mostly related to project management. This internship denoted great indicators regarding the software engineering goals, having for example early and well-defined requirements, a good planning process, easy team communication. By the final of the first semester, the requirements were already set and understood, planning and risks were controlled and implementation was developing according to the expectation.

With the ending of the second semester and the delivery of the final product it was realised that this project had a total of 103 functional requirements and 6 non-functional requirements which were mostly related to the identified features and included a rate of 100% of must-have and should-have requirements met and most of the nice-to-have requirements were also met. The consequence of this was the creation of a complete product of mobile protection way less dependent from the existing communication system which used text messages. The creation of such features in form of an independent Android SDK, alongside the new communication system through Google Cloud Messaging were the core of this internship being totally portable and making this product unique. For all these reasons, Updates Protect met the company goals being a great POC with a great potential to become a successful app.

Future Work

One big concern of the trainee was to develop and to create a product that found no problems in the future relative to its extensibility. The ability to grow and be improved in the future is sometimes very important as systems need to be prepared to evolve. This solution follows this concern and is easily extensible and there are some directions it can follow to in order to be improved. Firstly, the development of the livestream feature would be a pleasant addition to this solution as it would give a bigger control to the user on his

device's camera. In addition to that, there is still one other feature that could be built. As the system is now free from the dependency of the SIM card to receive messages from the server and adding to the fact that the app is a system app, the success rate of this app could rise to very high values as it could be totally able to recover from factory resets and SIM cards removals. With the addition of these two features, Updates Protect would by far distinguish itself from the other available apps in the market.

Final Thoughts

This final opinion is given looking back at this internship and the year past and as it is more personal, will be written in the first person.

It was a hard year, a year that combined a very challenging internship with a master's course to be concluded and some other issues. However, this project was the most motivating challenge I ever faced and gave me a great positive obsession with it, that lead me to give my best and to set the expectations the higher possible. The outcome of these 9 months are all positive and cover some different personal and professional aspects.

This professional experience marked the beginning of my life as a Software Engineer. At Wit Software, S.A. I had the opportunity to join a real-life project and learn a lot about the software creation lifecycle, starting from scratch with only the project idea and afterwards investigating, designing, architecting, debating, developing, testing and deploying the result. It is indeed a huge challenge to be a Software Engineer and master all these processes and this project really amazed me because it gave me the opportunity of passing through all this steps of Software Engineering and in the meantime I still learned so much about new technologies such as Android programming, new architecting and development strategies and much more from which I gained so much knowledge. As I finish this report I believe I can state that this year was extremely positive and I became a better person and professional.

Bibliography

- [1] There are officially more mobile devices than people in the world, Accessed: 12-01-2016
<http://www.independent.co.uk/life-style/gadgets-and-tech/news/there-are-officially-more-mobile-devices-than-people-in-the-world-9780518.html>
- [2] WIT Software, S.A., Accessed: 20-12-2015
<https://www.wit-software.com/company/why-wit/>
- [3] Factory Reset, Accessed: 04-01-2016
https://en.wikipedia.org/wiki/Factory_reset
- [4] Antivirus market share January 2015, Accessed: 16-10-2015
<https://www.opswat.com/resources/reports/antivirus-and-compromised-device-january-2015#antivirus-vendor-market-share>
- [5] Trackview, Accessed: 14-10-2015
<http://trackview.net/about.html>
- [6] Lookout, Accessed: 15-10-2015
<http://www.crn.com/news/mobility/300077009/lookout-harnesses-channel-as-segue-into-enterprise-with-new-security-product.htm>
- [7] IT project failure rates: facts and reasons, Accessed: 05-01-2016
<http://faethcoaching.com/it-project-failure-rates-facts-and-reasons/>
- [8] Google Cloud Messaging, Accessed: 16-10-2015
<https://developers.google.com/cloud-messaging/>
- [9] Google Cloud Messaging Architecture, Accessed: 03-12-2015
<https://developers.google.com/cloud-messaging/gcm>
- [10] Android Version History, Accessed: 08-10-2015
https://en.wikipedia.org/wiki/Android_version_history
- [11] Eardroid Website, Accessed: 14-10-2015
<http://www.eardroid.com/>
- [12] Eardroid Anti Theft on PlayStore, Accessed: 14-10-2015
<https://play.google.com/store/apps/details?id=com.eardroid.android&hl=en>
- [13] Cerberus Website, Accessed: 14-10-2015
<https://www.cerberusapp.com/>

- [14] Cerberus anti theft on PlayStore, Accessed: 14-10-2015
<https://play.google.com/store/apps/details?id=com.lsdroid.cerberus&hl=en>
- [15] Avast Website, Accessed: 14-10-2015
<https://www.avast.com/anti-theft>
- [16] Avast Anti-Theft on PlayStore, Accessed: 14-10-2015
https://play.google.com/store/apps/details?id=com.avast.android.at_play&hl=en
- [17] Trackview Website, Accessed: 14-10-2015
<http://trackview.net/index.html>
- [18] Find My Phone on PlayStore, Accessed: 15-10-2015
<https://play.google.com/store/apps/details?id=us.trackview&hl=en>
- [19] Prey Website, Accessed: 15-10-2015
<https://preyproject.com/>
- [20] Prey on PlayStore, Accessed: 15-10-2015
<https://play.google.com/store/apps/details?id=com.prey&hl=en>
- [21] McAfee Mobile Security Website, Accessed: 15-10-2015
<https://www.mcafeemobilesecurity.com/>
- [22] Security & Power Booster on PlayStore, Accessed: 15-10-2015
<https://play.google.com/store/apps/details?id=com.wsandroid.suite>
- [23] Lookout Website, Accessed: 15-10-2015
<https://www.lookout.com/>
- [24] Lookout Security & Antivirus on PlayStore, Accessed: 15-10-2015
<https://play.google.com/store/apps/details?id=com.lookout&hl=en>
- [25] REST, Accessed: 01-11-2015
https://en.wikipedia.org/wiki/Representational_state_transfer
- [26] User Stories: An Agile Introduction, Accessed: 24-10-2015
<http://www.agilemodeling.com/artifacts/userStory.htm>
- [27] Software Requirements, Accessed: 18-10-2015
https://en.wikipedia.org/wiki/Software_requirements
- [28] Simon Brown,
The Art of Visualising Software Architecture Communicating software architecture with sketches, diagrams and the C4 model, 2015.
- [29] Leslie Lamport, *L^AT_EX: a document preparation system*, Addison Wesley, Massachusetts, 2nd edition, 1994.
- [30] MVC, Accessed: 25-10-2015
<https://en.wikipedia.org/wiki/Model-view-controller>
- [31] Comparing Java Web Frameworks, Accessed: 25-10-2015
<http://static.raibledesigns.com/repository/presentations/ComparingJavaWebFrameworks-ApacheConUS2007.pdf>

- [32] STRUTS 2 vs SPRINGMVC: Know the Difference & Choose the Best One Based On Your Requirements, Accessed: 25-10-2015
<http://www.cygnnet-infotech.com/blog/struts-2-vs-springmvc>
- [33] Russell Sears, Catharine van Ingen, Jim Gray,
To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?, Microsoft Research, University of California at Berkeley, April 2006.
- [34] Database System vs File System, Accessed: 29-10-2015
<http://raima.com/database-system-vs-file-system/>
- [35] Lean Software Development, Accessed: 13-10-2015
https://en.wikipedia.org/wiki/Lean_software_development
- [36] Burn Down Chart, Accessed: 20-01-2016
https://en.wikipedia.org/wiki/Burn_down_chart
- [37] Marco Vieira,
Project Management, Module 3: Planning & Tracking - Risk Management, October, 2014.
- [38] Marco Vieira,
Project Management, Module 3: Planning & Tracking - Estimation, October, 2014.
- [39] Simon Brown,
http://www.codingthearchitecture.com/2014/08/24/c4_model_poster.html
- [40] Android IntentService,
<https://developer.android.com/reference/android/app/IntentService.html>
- [41] Tiles framework,
<https://tiles.apache.org/>
- [42] Composite Pattern,
https://en.wikipedia.org/wiki/Composite_pattern
- [43] Jersey RESTful Web Services in Java,
<https://jersey.java.net/>
- [44] Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web,
<http://getbootstrap.com/>
- [45] Gentella Admin template,
<https://www.bootstrapzero.com/bootstrap-template/gentella-free-bootstrap-admin>
- [46] CrackStation securing passwords,
<https://crackstation.net/hashing-security.htm>
- [47] XML Path Language,
<https://www.w3.org/TR/xpath/>
- [48] Android System Permissions,
<https://developer.android.com/guide/topics/security/permissions.html>
- [49] Android Cursor,
<https://developer.android.com/reference/android/database/Cursor.html>