Masters in Informatics Engineering

Internship

Final Report

iOS integration with Internet Chat Services

Tiago André Alves Pereira
tpereira@student.dei.uc.pt

WIT Software Supervisor:
Frederico Lopes

DEI Supervisor:
Hugo Gonçalo Oliveira

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

Acknowledgments

First, I would like to express my gratitude to WIT Software for creating this opportunity and for providing me a fantastic atmosphere during this one-year internship.

I would like to express my gratitude to both supervisors, Frederico Lopes and Hugo Gonçalo Oliveira for their support, patience and time spent along the entire internship. I also would like to thank the whole iOS team of the RCS product for their tremendous help in this learning phase.

Next, a big thanks to all my friends and, particularly, to my soul mate for the amazing experiences that they made me live in the past six years.

Finally, I would like to thank all my family and to give a special thanks to the person who made all of this possible: **my mother**. She is a person who always fights for me affording all the necessary conditions that I need to become someone in life.

Abstract

With the consolidation of the Internet on mobile devices, applications that provide telecommunications services began to emerge. These applications (known as OTTs) take advantage of the structural conditions of the Internet and have become a threat to Mobile Network Operators. In fact, as these applications are free and have thus great acceptance by the society, this is causing a significant loss of revenue by the Mobile Network Operators.

To mitigate this problem, the GSM Association created the joyn initiative, which aims to provide Rich Communications Services to the end users including enhanced calls, messaging and media content share. The Mobile Network Operators realized that, to keep their customers, they should provide similar services (considering the OTTs) using the same policy: for free! These services are requested to software companies accredited by the GSM Association who develop joyn-branded products, as is the case of WIT Software, SA.

The main objective of this project is to enrich the WIT Software joyn-branded product, differentiating it from the competition, by integrating the existing solution with popular Social Networks. After the integration it will be possible to check the list of Social Network friends, check their availability in the chat service to communicate with them within the joyn application developed by WIT Software, SA, thus sharing countless life experiences.

Keywords

"Internet Chat Services", "Social Networks", "joyn", "Rich Communication Services", "RCS", "Telecommunications", "Mobile Network Operators", "Over The Top Applications", "iOS".

Resumo

Com a consolidação da internet em dispositivos móveis, novas aplicações que fornecem serviços de telecomunicações começaram a emergir. Essas aplicações (conhecida por OTTs) tiram partido das condições estruturais da internet e são consideradas uma ameaça pelas operadoras de telecomunicações móveis. De facto, como estas aplicações são gratuitas e têm, assim, uma grande aceitação por parte da sociedade, causam uma significante perda de receita monetária às operadoras de telecomunicações móveis.

Para mitigar este problema, a GSM Association criou a iniciativa joyn, cujo objetivo é fornecer Rich Communications Services aos utilizadores finais incluindo chamadas melhoradas, troca de mensagens e partilha de vários conteúdos. As operadores de telecomunicações móveis perceberam que, para manterem os seus utilizadores, teriam que fornecer serviços semelhantes (considerando as OTTs), usando a mesma política: gratuitamente! Esses serviços são requisitados a empresas acreditadas pela GSM Association que desenvolver produtos com a marca joyn, como é o caso da WIT Software, SA.

O principal objectivo deste projeto é enriquecer o produto da marca joyn da WIT Software, diferenciando-o dos competidores diretos através da integração do mesmo com as redes sociais populares hoje em dia. Após esta integração, é expectável um utilizador possa verificar a sua lista de amigos da rede social, visualizar o estado de presença desses mesmos amigos e através do serviço de Chat contactá-los partilhando assim inúmeras experiências de vida, tudo isto dentro da aplicação joyn desenvolvida pela WIT Software.

Table of Contents

Index of Figures

Index of Tables

Glossary

Term	Description
API	An Application Programming Interface is a group of methods or access points that software provides so that external parties can make use of their functionalities with no need to know the intrinsic processes.
Burndown Chart	Burndown Chart is a graphical representation of a Team's outcome at a given point of a Sprint. It plots the evolution of the remaining points – Sprint's User Stories still opened – and the remaining hours – corresponding to the Tasks still not closed inside each User Stories.
Callback	In computer programming, a callback is a piece of executable code that is passed as an argument to other code, which is expected to call back (execute) the argument at some convenient time. The invocation may be immediate as in a synchronous callback or it might happen at later time, as in an asynchronous callback.
DoD	The Definition of Done is a document that states the requirements that must be met before a Backlog User Story is marked as done.
GSMA	The GSM Association (GSMA) is a group of mobile operators and companies related to the mobile market, created in 1995, to define standards for mobile communication systems.
Impediment	In Scrum, an impediment is something required for the Team Members to finish a Task that is outside of their scope in the project. Impediments are assigned to the Scrum Master so that he can solve them as soon as possible.
joyn	joyn is the brand created by GSMA to identify the applications that comply with their specifications to deliver Rich Communication Services to the end user.
MNO	A Mobile Network Operator is both a provider and manager of the technologies and infrastructures required to supply mobile communication services for their users, such as messaging or telephony.
Mockup	Regarding this project, a Mockup is both a graphical illustration of a feature's use cases and a proposal for the User Inter- face (UI)/User Experience (UX) of that feature in the current product.
OTT Application	Over The Top applications refers to applications that deliver multimedia content over the Internet, like audio and video calls between two end users.
Planning Poker	Planning Poker is an activity performed by the Team Members in order to estimate the effort for each User Story in the Product Backlog, based on their Velocity.
PoC	A Proof of Concept is a demonstration of certain feature's

Term	Description
	feasibility inside a product that is obtained through its successful implementation.
Product Backlog	Product Backlog is an artefact that contains all the ideas for the project, in the form of User Stories. These stories are organized by the priority given by the Product Owner, and are also assigned an effort that is the estimate given by the Team Members, as a result of the Planning Poker activity.
Prototype	The prototype was really important and can be seen as an early sample, model or release of a product built to test a concept or process or to act as a thing to be replicated or learned from.
RCS	Rich Communication Services is a global initiative by GSM Association (GSMA) to compete the Over the Top applications that are penetrating the mobile communications market at a fast pace.
Redmine	Redmine is a web platform for project management that includes various tools to check the project state, such as charts, task boards, etc. This tool was used in the project because it is widely used inside WIT Software and has support for the Scrum framework.
Scrum	Scrum is an iterative and incremental agile development process/framework. It is task priority-oriented, meaning that it seeks to fulfil tasks with higher priority first.
Sprint	A development Sprint is a fixed implementation period, where the Team Members focus their effort in developing the features that match the User Stories they compromised to. Sprint duration is usually between one and four weeks; in this project, two weeks Sprints took place.
Sprint Backlog	Sprint Backlog is composed by the User Stories that are selected by the Team Members to be accomplished along a Sprint. Each of these User Stories is divided into smaller tasks, that must detail the steps the Team must take in order to complete a story.
Sprint Meeting	Sprint Meeting comprises two Scrum activities at once in our project: Sprint planning and Sprint reviewing. The first consists of planning the next Sprint, deciding which User Stories will be performed. Its counterpart is Sprint reviewing, which stands for a brief review on the previous Sprint outcome.
Task	A Task is a smaller assignment, subset of a User Story, which states something that must be implemented as part of that functionality. The tasks for each User Story may be defined when that story is assigned to a development Sprint, defining specifically what needs to be done to mark the story as closed.
User Story	A User Story defines a possible user action with the product in order to state a project requirement in the everyday

Term	Description
	language. A User Story can comprise multiple Tasks that define meticulously what must be created by the development Team Members in order to mark the story as done.
Velocity	The Velocity of a Development Team is given by the sum of the effort points of the finished User Stories inside a Sprint. Velocity is used at a new Sprint's beginning, when of the selection of the User Stories that the Team will compromise to deliver.
Waterfall	The waterfall model is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation, and Maintenance.

Acronyms

Acronym	Description
AP	Access Point
API	Application Programming Interface
B2B	Business-to-Business
DoD	Definition of Done
FT	File Transfer
GC	Group Chat
GSMA	Groupe Speciale Mobile Association
HTML	HyperText Markup Language
IM	Instant Messaging
IMS	IP Multimedia Subsystem
JID	Jabber ID
LTE	Long Term Evolution
MMS	Multimedia Message Service
MNO	Mobile Network Operator
OTT	Over The Top
PoC	Proof of Concept
RCS	Rich Communication Services
RCS-e	Rich Communication Suite – enhanced
SDK	Software Development Kit
SIM	Subscriber Identity Module
SMS	Short Message Service
SP	Service Provider
VoIP	Voice over IP
WCS	WIT Communicator Suite
WMC	WIT Mobile Communicator
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

1. Introduction

The present document reflects the work done at WIT Software, SA during the year-long internship part of the Masters' degree in Informatics Engineering in the Department of Informatics Engineering of the University of Coimbra.

This work was supervised by Frederico Lopes, engineer at WIT Software, SA and Hugo Gonçalo Oliveira, PhD professor at Department of Informatics Engineering of the University of Coimbra.

This internship took place at WIT Software, a software development company specialized in rich and unified communications for Mobile Operators and Mobile Internet companies [1]. It was founded in 2001 and was originally located in the Pedro Nunes' Institute (IPN).

Nowadays the company operates in three different business units. The Telco unit that is mainly focused on Rich Communication Services (RCS) and Unified Communications providing converged solutions based on the IP Multimedia Subsystem (IMS) for voice (VoIP, Mobile VoIP and Voice over LTE), messaging (SMS, MMS and IM), Rich Communication Suite (RCS-e, RCS2.0 and RCS 5.0/5.1) and Multimedia Telephony Services (MMTel). The Mobile unit, which has been developing mobile services in Android, iOS, Windows Phone, Blackberry, J2ME and Symbian for Telco Operators, Banks, Utility and Media companies. The TV unit that is mainly focused in the development of software applications for Cable and IPTV operators. The products provided by this unit go from TV Widgets Platform, VoD Storefront, Converged Communications, Contextual TV Apps and TV Everywhere solutions.

The present internship is integrated into the Telco unit and aims to develop a module for iOS that will enrich the unit's main product: WIT Communications Suite (WCS) [2].

1.1 Context

During the past years we have seen a society that is completely dependent on the telecommunication services in their daily lives. The telecommunication solutions reached a new level, providing new experiences to the end users over the Internet. Nowadays, it is quite easy to launch applications that provide telecommunications services over the Internet because of their low cost of production and deployment. The result of these facilities is the range of applications that provide such services in the electronic markets.

Normally, the aforementioned applications are called Over The Top (OTT) applications. This term is used to mention a content or service provided through an infrastructure that is not controlled by a Service Provider (SP), in this case the SP are the Mobile Network Operators (MNOs). Initially, the focus of such applications was audio and video content but, recently, they have expanded to covering much of the content and services available on the Internet.

With the increasing number of smartphones, the use of OTT applications is very tempting because they only have one requirement that is an internet connection, having no cost to the user (only in premium content that does not deprive them of the features they provide). These days, it is completely banal to find Public Access Points (APs) and/or have data plans

with affordable prices that further contribute to the utilization of such application. In telecommunications, OTTs bring major benefits compared as some telecommunications services offered by the MNOs, capturing the people to their usage and therefore to make them successful. Then, with the adherence of people, services such as Short Message Service (SMS) and Multimedia Messaging Service (MMS) tend to be overcome in the near future giving rise to features like Chat and File Transfer of the OTTs applications.

The decreasing use of services provided by the MNOs makes them lose their negotiating strength to customers, thus leading to a significant fall in the revenues worldwide. A good example is Skype (OTT application), which, according to a research [3], 43% of mobile operators view as a major threat to their revenues. Skype has almost 300 million active monthly users spending a total of 2 billion minutes per day on Skype, equating to 730 billion minutes a year. On average, each of these active users spends just over 7 minutes per day, or 2,555 minutes a year using Skype communication services. Skype is costing the Telco industry $100 million per day and a staggering $36.5 billion per year.

In order to face this problem, operators had to look at the OTT applications through two different perspectives:

- The first is facing the applications as a threat, because they lead to a huge loss in revenue since the services they provide are not used.
- The second is to view the situation as an opportunity because it is embedded in a market with many segments to be explored and the society is not fully linked to the OTT applications that they use.

Considering the second perspective, the MNOs must address the problem by placing themselves side by side with these applications. Therefore, they must provide a service that brings all the features, provided by OTTs, together combining such an offer with an innovative design that leads people to use them.

Based on the above words an entity named GSM Association (GSMA) emerged. The GSMA is an association of telecommunications operators and related companies with a goal of defining standards that should be used by operators worldwide in regards to the development and production of the Telco services. So, to ensure that the operators are able to enter the market and countering the emerging OTT applications the Rich Communication Services (RCS) initiative started [4]. For MNOs, deploying RCS is key to ensure their service retains relevance for consumers, keeping them connected and offering them competitive solutions with alternative applications. RCS delivers the messaging service step-change the consumer is seeking, while enabling the continuation of operator-centric messaging services. The scope and variety offered by RCS is a route to developing more targeted, engaging customer propositions – for example via chat-based games, mobile learning, smart ads and promotions – all helping to drive 'stickiness' and revenue.

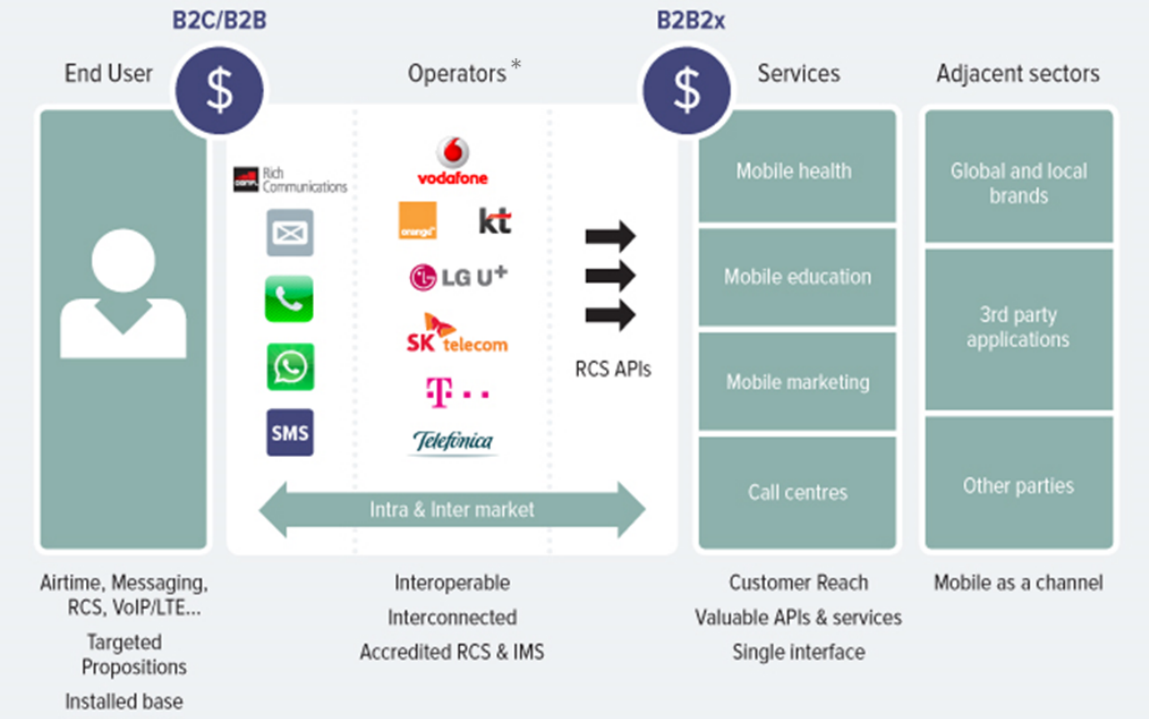

Figure 1. RCS Business Model [5]

In Figure 1, it can be seen RCS offering an opportunity to expand core products and services while building upon the unique operator propositions of Ubiquity, Global interoperability, Quality of Service assurance, Security and Privacy management, Trusted Service provider. By enabling operators to generate new revenue streams through the creation of apps and Business-to-Business[1] (B2B) services, RCS can be a route to getting a return on the IMS investment the operators made.

The RCS specification details how the applications should be developed, since their internal architecture, the features they must have and the key points that the user interface should take, can be considered as a manual of good practices that should be adopted and implemented. This specification is a document that is iterated regularly and at the moment had five different releases. What differs between releases is the increased set of features that enrich the specification. The products developed under this specification are known as joyn-branded products and must be interoperable over the different MNOs, which means that, assuming all operators would deploy a solution of this type, the features would be available regardless of the carrier nature, such as the simple SMS and call services.

Some MNOs considered the threat of the OTTs as an opportunity and are currently providing joyn-branded products. These products are provided for software companies that operate in the telecommunications field as WIT Software. WIT is a company accredited by the GSMA and has been developing joyn-branded products for some years ago, being a company in the forefront of the specifications as can be stated in the following image:

[1] Business-to-business (B2B) describes commerce transactions between businesses, such as between a manufacturer and a wholesaler, or between a wholesaler and a retailer. Contrasting terms are business-to-consumer (B2C) and business-to-government (B2G).

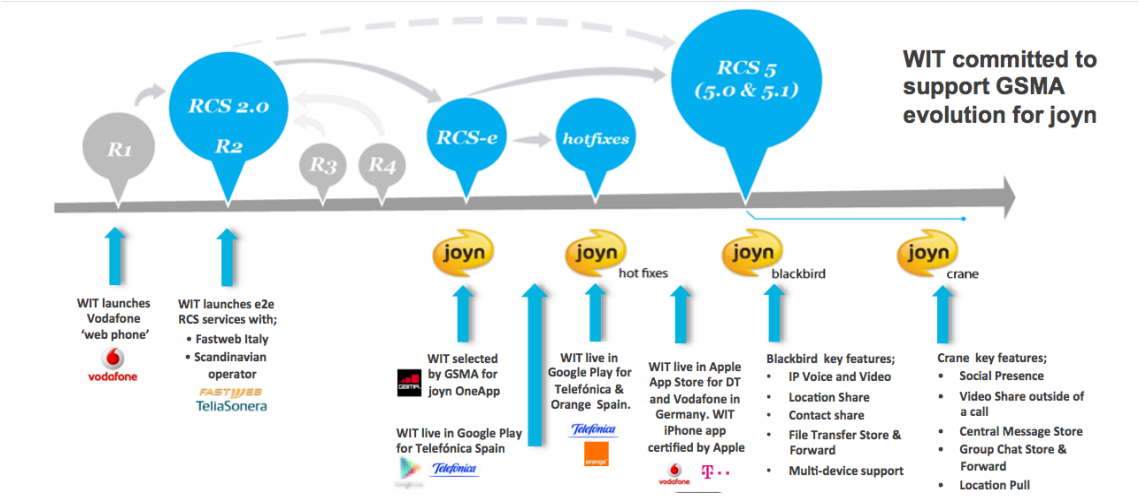

Figure 2. WIT Software joyn products evolution

The company has been developing products since release 1 and was recently accredited for release 5 (joyn Blackbird). Having an accredited product in a certain release, presupposes that it provides all the features that are in the specification. Through the State Of The Art survey, it was verified that there are several software companies that are accredited in the same release that WIT Software, SA. So, to differentiate us, we must innovate and not get caught up in what is in the specification because, as stated earlier, the GSMA is composed for telecommunications operators and related companies, of which, WIT Software, SA is a part of, and may thus suggest new features for the specifications. Due to this innovation, the topic of this internship arises: iOS integration with Internet Chat Services.

1.2 Motivation

There is a great need to introduce the Internet Chat Services in the product developed by WIT Software, SA. Through a research, that later was part of the State Of The Art review, three major Internet Chat Services were recognized and they deserve special attention due to the number of active users. If many people are using these services on a daily basis, it means that a successful integration on the RCS product could be a plus, bringing more people to its utilization. Furthermore, there are few applications on the market (OTTs) that make this type of integration and have huge success worldwide, such as Skype that allows its users to perform conversations with Facebook users (social network with more than 1 billion active users).

To sum up, the motivation of this internship is to enrich the company's product with an innovative feature. This innovation allows the company to stand out against its direct competitors providing, to the MNOs, something that the other companies don't have, allowing them to counteract the OTT applications recovering the revenue they have lost in the recent years.

1.3 Goals

For many students, the internship is the first contact with the world of work. So, the supposed goals of this project can be divided in three major perspectives: technical goals from the features perspective, internship goals and company goals.

1.3.2 Technical goals

Regarding the technical objectives of this project, at the end of internship it is expected that a user can:

- **Authenticate** through a Social Network account in the RCS application by WIT Software, SA.
- **Load remote contacts** from that Social Network (friends). In this functionality it is important to consider one specific point:
  - ⇒ Contacts Information: In addition to the load of remote contacts, some details about the same friend should be obtained (i.e. birthday, hometown, current city). This information is useful and can be used to develop other type of functionalities like Contact Enrichment (Sync.me-like [6]) or birthday alerts.
- Display the Social Network **contacts' presence**. This objective is related to the previous one and will allow a user to know when its contacts are online and available to chat.
- Make **single conversations (Chat)** with Social Network friends in the RCS application. It may also be possible to send notifications and emoticons/emojis in a conversation, providing all the expected UX in a service of such type.
- **Send files (File Transfer)** to Social Network friends through cloud storage services. In a chat conversation a user can send rich content to its contacts (i.e. photos, videos, audio).

Another aspect that should be noted is the **Group Chat** functionality (announced at the internship proposal) that was not referred in the previous topics. The reason for that is the fact that the functionality is impossible to be done due APIs constraints. However, a later section of this document (Solution Architecture) will detail this inconvenient. In order to make the internship more complex, a new feature replaced this previous:

- **Share** received content (text, images and videos) on Facebook. This feature will be available to all Chat data sources and for two platforms (Android and iOS).

1.3.2 Internship goals

The main objective in this perspective is to consolidate the knowledge acquired during five years of Software Engineering at Department of Informatics Engineering of the University of Coimbra. This is an opportunity to put into practice the knowledge acquired during the academic period and thus gain experience in the development of services and software in an organizational environment with real clients and real implications.

1.3.3 Company goals

The company is making an investment in the trainees, providing all the resources needed to develop software with quality. So, in this view, the main goal is to finish the project successfully. This means that the requirements presented to the University have been met thus bringing added value to the product in which they are integrated because, at this

moment, none of the features that will be implemented are part of the GSMA specifications. If these features are not part of the GSMA specifications they can be distinguishing factors when compared to direct competitors who develop joyn-branded applications.

So, to summarize, the aim of the company is to make an investment in my work in order to receive a module that will enrich the main product and may differentiate it from its competitors. In other words, it is expected that at the end of this internship, the company's joyn application will enable its users to communicate with their friends in the most popular social network (with more users worldwide).

1.4 Document Structure

This report is organized in five chapters, including this introduction and thee following:

- **State Of The Art:** This section is a key part of this report because it allows to evaluate the viability of the product through the analysis of existing solutions and identifying the benefits and the weaknesses that can be explored.
- **Approach:** This section describes the followed software development methodology, the planning, as well as the requirements of the project and the risks associated to these requirements.
- **Solution Architecture:** This section provides an overview on the application architecture and describes the technical decisions made.
- **Implementation challenges and Evaluation:** This section is intended to explain the most important decisions taken during the implementation phase and all the tests made to ensure the project quality.
- **Conclusion:** This section summarizes all the work done during the one-year internship, highlighting the lessons learnt.

This document is also complemented by a set of appendixes that give a detailed description about the chapters mentioned above:

- **Appendix A – State of The Art:** In this document, a detailed description of the current solutions and services is presented.
- **Appendix B – Approach:** This appendix details the software development methodology used and the planning of the project.
- **Appendix C – Application Use Cases:** This document corresponds to the designed application Use Cases.
- **Appendix D – Architecture:** This document provides a detailed architecture of the developed features as well as some technical decisions made during the elaboration.
- **Appendix E – Evaluation:** This document provides a detailed plan about the performed tests: functional, networking and usability.

2. State Of The Art

The focus of this chapter is to analyse the current market and the applications that compete against the internship product. Analysing the market makes it possible to understand the saturation level of the economic sector in which the product falls, thus estimating the success, impact and execution risks that the product may have. It is very important to conduct a comparative analysis between the OTTs too because it makes it easy to realize how our product can penetrate in the market and its viability. After exploring the functionalities, we are able to understand their strengths, their weaknesses and what we could make better than them.

Firstly, a list of companies (direct competitors) and OTTs applications (indirect competitors) that were found will be presented. The application is considered a competitor if it has all or part of the features that the WIT Software product. As for the companies, they all act in the same market share than WIT Software, SA.

Afterwards, another focus of this document is the social networks (internship context). Some popular Social Networks will be presented together with the features they provide to the end users.

An overview about the application/service will be presented, as well as some information about its users, main features and rating on electronic markets. Before proceeding to the next sections, it's important to clarify the following aspects:

- When possible, the Monthly Active users will be presented as reference measure because it is the value that better reflects the success of the application. If such information is not available, the number of registered users will be presented. This value isn't so accurate once it only reveals the amount of accounts created and the service may have a greater percentage of registered users that don't use the service at all.
- Another aspect is the mobile applications rating in the electronic markets. Two different ratings will be displayed. One for the USA App Store and another for Google Play Store. The USA App Store was chosen because this internship relies on iOS development and the USA store is a worldwide reference. We cannot take strong conclusions from this measure because an application can be a worldwide success but not used in the USA, or not embraced by iPhone/iPad users. However, this risk is irrelevant once the application rating is mostly considered more as a curiosity than a variable for the analysis. The Google Play Store was chosen too because it is the most used store for mobile applications.

Notice that this section is just a short summary of the performed study. To consult an extensive analysis, refer to the **Appendix A - State Of The Art**.

2.1 Competitors

The competitors of the RCS [4] product developed by WIT Software will be detailed in this section. Two types of competitors were considered:

- Direct competitors: companies that develop RCS [4] based software for telecommunications operators and compete directly with WIT Software;
- Indirect competitors: emerging applications that were developed (mostly) by independent companies and compete directly with the services that the operators provide.

To analyse and compare these entities, it is necessary to define a set of properties, which are functionalities provided to the end users that enrich the user experience.

2.1.1 Most important features

The telecom operators provide three major services that are known for most users, such as Voice Calls, Short Message Service (SMS) and Multimedia Message Service (MMS). Nowadays, with the continued evolution of the technological field some applications have emerged and changed the concept imposed by these three services, thus introducing more sophisticated mechanisms that give a better user experience.

There is a larger set of features, but only a subset will be presented in this section. For a detailed overview, refer to the **Appendix A - State Of The Art, Section. Most important features**. The features chosen are the following:

- **Contact Discovery:** capability to identify contacts on native address book that are using the same application.
- **Network Address Book:** capability to centralize the address book across a wide range of communications services.
- **Chat:** capability to exchange instant messages with another contact.
- **Message State Notifications (in Chat):** capability to see the state of the sent messages.
- **Typing Notifications (in Chat):** capability to receive/send typing notifications (when a contact is writing).
- **Group Chat:** capability to exchange instant messages with multiple contacts at the same time in the same chat entry.
- **Message State Notifications (in Group Chat):** capability to see the state of the sent messages.
- **Typing Notifications (in Group Chat):** capability to receive/send typing notifications (when a contact is writing).
- **Chat File Transfer (CFT):** capability to transfer a file from sender to receiver and store the file in the receiver's file folder.
- **Group Chat File Transfer (GCFT):** capability to transfer a file from sender to multiple receivers and store the file in the receivers' file folder.
- **Location share:** capability to share one location (current, place on map, etc)
- **Contact Share:** capability to share vCard information

- **IP Voice Calls:** capability to make voice calls over IP.
- **IP Video Calls:** capability to make video calls over IP.
- **Voice Messages:** capability to send messages with voice content to another contact.
- **Video Messages:** capability to send messages with video content to another contact.
- **Stickers:** capability to send/receive stickers to other contacts in the chat room.
- **Emoticons:** capability to send/receive emoticons to other contacts in the chat room.
- **Emojis:** capability to send/receive emojis to other contacts in the chat room.
- **Online Status:** capability to see the contact's presence.
- **Profile Picture:** capability to associate a picture to a contact profile.
- **Theme and/or Skins:** capability to change the applications layout with themes and/or skins.

2.1.2 Direct Competitors

The joyn initiative by the GSMA [7] is a great effort that the MNOs are adopting, trying to bring back the clients for their services. There are specifications accredited by the GSMA that MNOs must follow to provide an RCS-branded application [4], however these solutions are very few. The main reason is all about the initial investment that is required by the outsourcing companies to develop and deliver a joyn product. Another reason is that as we are in a phase of adoption, the specifications of joyn are rapidly changing and companies fail to follow this evolution.

Nowadays, there are four companies that compete directly with WIT Software because they provide a resembling product based on the same specifications' documents. With a lot of effort, each of them attempts to differentiate their product continuously improving it, trying to provide a better outcome (more features, best way to do something) to the MNOs and eliminate the competition.

The next four companies are the direct competitors:

- **Jibe Mobile** (USA)
- **Nable Communications** (Korea)
- **Neusoft** (China)
- **Summit Tech** (Canada)

For a detailed overview about these companies, refer to the **Appendix A - State Of The Art, Section. Direct Competitors.**

Comparative Analysis

In order to compare them, we can make a detailed analysis about the features the direct competitors provide. The amount of features that the applications provide is one of the selection criteria by MNO's. Before the analysis it is important to mention that the question mark (?) means that the feature was not analysed. It is quite difficult to study all these applications because they are not available in the markets that we have access. The blank

space means that the application doesn't have the feature and the check mark (✓) means the applications has the feature.

The following table compare the features of the various entities:

	WIT Software	Jibe Mobile	Nable Communications	Neusoft	Summit Tech
Contact Discovery	✓	✓	✓	✓	✓
Network Address Book	✓	?	?	?	✓
Chat	✓	✓	✓	✓	✓
Message State Notifications (in Chat)	✓	✓	✓	✓	✓
Typing Notifications (in Chat)	✓	✓	✓	✓	✓
Group Chat	✓	✓	✓	✓	✓
Message State Notifications (in Group Chat)	✓	✓	✓	✓	✓
Typing Notifications (in Group Chat)	✓	✓	✓	✓	✓
CFT	✓	✓	✓	✓	✓
GCFT	✓	✓	✓	✓	✓
Location Share	✓	✓	✓	✓	✓
Contact Share	✓	✓	✓	✓	✓
IP Voice Calls	✓	✓	✓	✓	✓
IP Video Calls	✓	✓	✓	✓	✓
Voice Messages	✓	?	?	?	?
Video Messages	✓	?	?	?	?
Stickers	✓				
Emoticons	✓	✓	✓	✓	✓
Emojis	✓	✓	✓	✓	✓
Online Status			✓	✓	✓
Profile Picture	✓	?	?	✓	✓
Theme and/or	✓				✓

	WIT Software	Jibe Mobile	Nable Communications	Neusoft	Summit Tech
Skins					
Integration with Internet Chat Services					

Table 1. Direct Competitors Features

As previously mentioned, it is very difficult to obtain all the functionalities of the direct competitors. The applications aren't reachable and everything filled has been based on the companies' websites. Functionalities like "Network Address Book", "Video and Voice Messages" and "Profile Pictures" weren't found for all products.

Another aspect that was mentioned earlier is the fact that all products are accredited by the GSMA with the same specifications. So, it is expected that the features are fairly similar. However, we can highlight the fact that WIT Software has a particular concern with the chat experience and it is the only company that implements Stickers on their product. It should also be noted that Neusoft, Nable and Summit Tech have a different concept of presence by giving the user the power to manage their availability. In this type of application, this feature is very important because it gives a different feel to users, indicating who's available in order to start some kind of interaction.

To conclude this analysis, it is important to highlight the fact that no company provides the Internet Chat Services functionality in their products. This feature is the core of this internship and will be implemented because it brings an increased value to the product developed by WIT Software. If everything goes well during the internship, this could be a very important factor to highlight the company from its competition.

2.1.3 Indirect Competitors

Nowadays, the use of technology in our daily lives is completely trivial. The telecommunications solutions reached a new level, providing new ways of communications to the end users. In the last years Internet based applications growth almost exponentially and the main reason behind is the fact that there is no need of a large prerequisites subset, as well as a very low cost. The principal prerequisite is an Internet connection (Wi-Fi or carrier data plans) and most applications have free/freemiun access to their features.

The usage of this type of applications starts with curiosity but then it becomes a vicious circle when friends invite other friends and so expand the number of the OTT users. These applications can easily adapt to the improvements made to the infrastructure of cellular networks (i.e. 3G/4G LTE), delivering rich content across multiple devices in a reliable away.

A study [8] conducted by Mike Hibberd (editorial director at Telecoms.com, Mobile Communications International magazine and Banking Technology) says "Informa Telecoms & Media has predicted that global annual SMS revenues will fall by US$23bn by 2018, to US$96.7bn, down from US$120 billion in 2013. The decline in global SMS revenues will largely be caused by the continuing adoption and use of over-the-top (OTT) messaging applications in both developed and emerging markets."

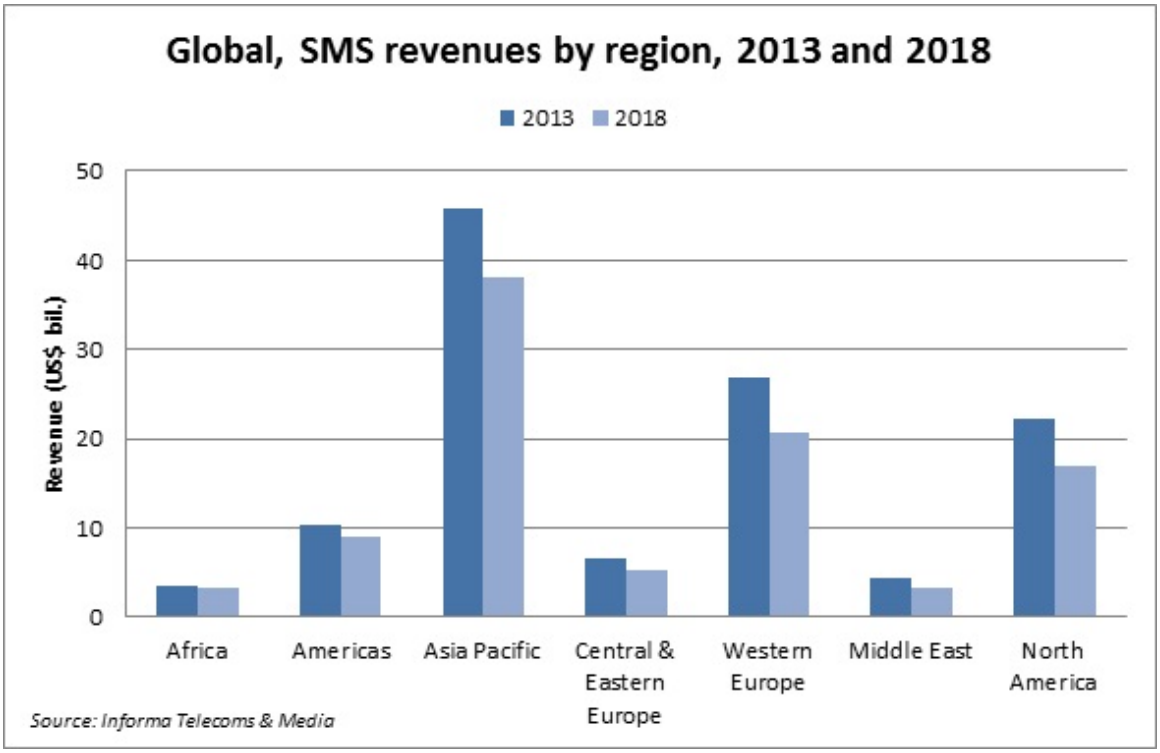

Figure 3. SMS Revenues Prevision (2018)

This chart just confirms the threat that OTTs imposes. This value refers to the near future but we can imagine what might happen to the operators' revenues with the continuous development of the technology and consequently the appearance of new applications. Well, it is obvious that revenues will fall continuously if the operators continue to do nothing. However, they are trying to mitigate the problem with the joyn initiative.

For the present document an analysis about 25 OTTs players, the indirect competitors, was made. The most significant applications based on their market share are presented in this section. For each, a short overview will be given encompassing the main features, distinctive features, when found the active users (when not found the nearest value, registered) and the rating in the electronic markets. For a detailed overview, including all the 25 OTTs players analysed, refer to the **Appendix A - State Of The Art, Section. Indirect Competitors.**

It is important to mention the fact that this analysis was taken in late October 2013. As these applications are constantly growing, some reported figures might be outdated by the time of reading this document.

WhatsApp Messenger [9]

Users: 350M monthly active [10]

App Store Rating (USA): 4,5/5 stars (113 802 Ratings) [11]

Google Play Rating: 4,6/5 (4 482 688 Ratings) [12]

WhatsApp Messenger is an over-the-top application released in 2009 by WhatsApp Inc. It is the main product of this company, formed by 2 ex Yahoo! workers. This app, like most of others apps, is a cross-platform application exclusive for smartphones.

By 2012, WhatsApp saw its growth explode substantially from 2 billion messages sent per day to 10 billion, an honourable achievement by the company. These days, it runs in 6 different operating systems. The iPhone OS and Android OS are the most significant and reached a very large set of million active users per month who sent more than 31 billion messages and 325 million photos per day [13] [14]. The service is free in the first year and after that, the user has to pay 0,99 US Dollars per year.

Main features:

- Contact discovery of the contacts using WhatsApp through an ID that consists in mobile number associated to the SIM card.
- IM (1-to-1 and group till 50 persons and 50 parallel group sessions)
- File transfer (video, images and audio)
- Localization and vCard[2] shares
- Conversation history backup

Distinctive feature(s):

- Ad-Free application. WhatsApp refuses to use this business model like most other OTTs. They say that people are tired of ad content, and they just want to provide something that works and saves people's money, making their lives easier [15].

Skype [16]

Users: 299M monthly active [17]

App Store Rating (USA): 3,5/5 stars (354 961 Ratings) [18]

Google Play Rating: 4,0/5 (1 482 834 Ratings) [19]

Skype is an application that allows users communicate through Internet. The communication is made by different ways since instant messaging until video-calls passing by the normal calls.

Two Kazaa Company workers released Skype first version in August 2003. Since then, the company business was bought by eBay in 2005, four years later was bought by a group of investors, and two years ago by Microsoft for 8,5 billion US dollars [20]. Skype always shown a great potential so the Microsoft (one of the major world powers) decided to make the most expensive deal ever since its existence

In February 2013, Skype was estimated to have around 800 million registered users. Another study in August 2013 shown a record of 70 million users in a concurrent session, in other words, Skype's servers have been able to allocate 70 million users at the same time [21].

[2] vCard is a file format standard for electronic business cards. vCards are often attached to e-mail messages, but can be exchanged in other ways, such as on the World Wide Web or instant messaging.

Main features:

- Contact discovery by the registered ID (normally an email address). Skype is totally independent of any Telco operator, so there is no need of having a SIM Card.
- File Transfer (video, images and audio)
- Free IM (1-to-1, group to 300 people), Voice-calls (25 people limited) and Video-calls (1-to-1).
- Video Conference till 10 simultaneous people through premium account.
- Full integration with Windows Live Messenger.
- Integration with Facebook

Distinctive feature(s):

- It allows a user to communicate with other people that don't have Skype account without using the native smartphone calls. In other words, it allows users to communicate with a normal cellphone that belongs to an operator like Vodafone or Orange inside the application itself. What make this feature stands out is the fact that Skype has great national and international tariff plans making the calls much less expensive than some telco operators [22].

Viber [23]

Users: 200M registered [24]

App Store Rating (USA): 4,5/5 stars (90 544 Ratings) [25]

Google Play Rating: 4,4/5 (1 511 350 Ratings) [26]

Viber is a cross-platform application for smartphones (recently with a desktop app) that consists in an OTT app with a huge market share offering various features to the users. It was developed by Viber Media, inc.

In the end of 2010, Viber has released an iPhone application with the goal of competing with Skype. Due to his great success, one year later, a pre-release Android app was available in Play Store but limited to 50000 users in order to maintain the good availability and scalability of the server-side. Nowadays, Viber runs in 10 different platforms including the most popular desktop operating systems (Mac OSX, Microsoft Windows).

Main features:

- IM (1-to-1 and group until 40 persons)
- Voice and Video calls (over IP)
- File transfer (video, images and audio)
- Doodle feature
- Full integration with native address book and native message system.
- Contact discovery of the contacts using Viber through an ID that consists in mobile number associated to the SIM card.

Distinctive feature(s):

- Interoperability between desktop app and mobile app. This is a really nice functionality that allows the user to transfer an active call for the desktop app to mobile app without switching off the other party. It is seen as an innovation because no other application does the same yet [27].

Facebook Messenger [28]

Users: 56,7M monthly active [29]

App Store Rating (USA): 4,5/5 stars (21 166 Ratings) [30]

Google Play Rating: 4,3/5 (1 128 850 Ratings) [31]

Facebook Messenger is a software application that allows users to perform text and voice communication. Due its full integration with Facebook native chat on web app, Messenger lets Facebook users talk with friends on both platforms.

One of the best things about this app is the fact that users can communicate with their friends without logging in the Facebook official application. It is much easier for the user and much lighter for the smartphone.

In some countries (like India, Australia, South Africa), Facebook Messenger allows users without a Facebook account to register in the application and enjoy the features that the application provides. All of this is possible by associating a phone number to the application [32]. Such a feature can be seen in two ways:

- An incentive to users without a Facebook account to join the social network.
- A threat. Why? Because Facebook indirectly is challenging the traditional mobile SMS texting services of telecommunications.

Facebook Messenger was released on August 9, 2011 iOS and Android and, lately (two months later) for BlackBerry OS. These days, Facebook application totalize 192 millions active users per month on Android, 147 millions on iPhone, 48 millions on iPad and Messenger application totalize 56 millions in all platforms.

Main features:

- IM (1-to-1 and group)
- File transfer (video, images and audio)
- Voice and Video calls over IP [33]
- Social Network integration

Distinctive feature(s):

- IM notifications on Android are very user friendly and an innovation. When a user starts a conversation with one friend, besides the notification on notifications bar, a popup appears in the main screen (like a bubble) and shows the friend's photo and the number of unread messages. Users can tap that bubble and directly read the

messages and respond to them.

iMessage + FaceTime [34]

Users: 250M monthly active [35]

App Store Rating (USA): n.a.

Google Play Rating: n.a.

iMessage is a free distributed service for text messaging developed by Apple Inc. It is available for iOS and Mac OSX and was presented by Scott Forstall in June 2011 at the WWDC [36]. Initially iMessage was released only for iOS devices and could be questioned due iOS restrict limitation, however Apple responded promptly indicating that there are over 200 million active iOS devices that are the target audience.

If users communicate with each other using iMessage and both of them have Internet connection, the message will be sent and received by iMessage. If both or neither one has Internet connection, the message will be sent and received by traditional method through the operator. This mechanism is transparent to the users. These days, iMessage allows a user to communicate with OSX devices that have iMessage configured.

Only by itself the iMessage is a text-message application with some features like File Transfer. Although, if integrated with FaceTime, it lets the users make voice and video calls over IP.

According to some numbers [35], Apple has approximately 250 million users using iMessage. Other source indicates that Apple's servers dealing with 2 billion messages sent by iMessage per day [37].

Main features:

- IM (1-to-1 and group)
- File transfer (video, images and audio)
- Voice and Video calls over IP
- Cloud storage integration

Distinctive feature(s):

- Full interoperability with all devices. The first thought of iMessage was to let users communicate with others apple users that have iPads or iPods (without Telco operators) to enhance the user satisfaction.

Other main feature is the fact that one Apple user can communicate using different devices and maintaining the same source contact without loosing any type of data.

Comparative Analysis

After this overview about the top five OTTs a detailed comparison of their features is provided in table 2. The amount of features each one covers is one of the selection criteria by users. It is important to mention that it wasn't possible to find some features of studied applications. This sections is a short version of the performed study, to a detailed overview refer to **Appendix A - State Of The Art, Section. Indirect Competitors.**

	WhatsApp	Skype	Viber	Messenger	iMessage + FT
Contact Discovery	✓	✓	✓	✓	✓
Network Address Book	✓	✓	✓	✓	✓
Chat	✓	✓	✓	✓	✓
Message State Notifications (in Chat)	✓	✓	✓	✓	✓
Typing Notifications (in Chat)	✓	✓	✓	✓	✓
Group Chat	✓	✓	✓	✓	✓
Message State Notifications (in Group Chat)	✓	✓	✓	✓	
Typing Notifications (in Group Chat)			✓	✓	
CFT	✓	✓	✓	✓	✓
GCFT	✓	✓	✓	✓	✓
Location Share	✓		✓	✓	✓
Contact Share	✓	✓	✓		✓
IP Voice Calls		✓	✓	✓	✓
IP Video Calls		✓	✓		✓
Voice Messages	✓			✓	
Video Messages		✓	✓		
Stickers			✓	✓	
Emoticons	✓	✓	✓	✓	

	 WhatsApp	 Skype	 Viber	 Messenger	 iMessage + FT
Emojis	✓	✓	✓	✓	✓
Online Status	✓	✓	✓	✓	
Profile Picture	✓	✓	✓	✓	✓
Theme and/or Skins	✓				
Integration with Internet Chat Services		✓		✓	

Table 2. Indirect Competitors Features

This comparison shows that Viber is the most complete OTT in terms of features in this top. It has a good acceptance by users (both store ratings are high) and a significant number of users. Thus, this can be seen as a real threat to the revenue of MNOs.

The second most complete OTT in terms of features is Facebook Messenger. It offers an excellent usability and is constantly growing. But Messenger has both an advantage and a drawback. The drawback is the fact that it can be only used by people who have a Facebook account but, Facebook has more than 1 billion users which makes it an application with tremendous potential.

In terms of features, the differences between Skype and Facebook Messenger are minimal. Skype is an application with a huge number of users (possibly because it is the most known and oldest at international level) and has many positive aspects. One of the most significant is the fact that it enables also to chat with Facebook contacts (internship core) beyond the normal chat service. The Facebook chat is limited in terms of features because limitations of the public API to third parties. However, this feature is a plus and a distinctive aspect that makes the application very tempting and worldwide used.

WhatsApp is the application with more success looking at the number of active users. We can and should ask why since, as compared to the others, it is a poor application in terms of functionalities. It only allows users to take advantage of a chat service and is the only one not having the IP Voice Calls. So, why do people use it? The answer lies in how they do things. WhatsApp provides just a small subset of features, but it gets the job done well enough for the most part, capturing people to use it and providing a seamless service not to stop using.

Finally, the iMessage service provided by Apple. Despite being only an application available in the Apple "world", it has a quite significant number of active users. This number is due to the fact that the application is native on devices made by Apple (iPhone, iPad, iPod Touch, Mac) and is full interoperable between user's devices. Another reason is the set of applications that can integrate with it making it extremely powerful (this analysis only took into account the integration with FaceTime).

2.2 Social Networks

By nature, human beings are social creatures which are expected to spend much of our time socializing with each another. With the evolution of the Internet, people began to share their ideas and feelings in a new form of communication by the Social Networks. They started as a new trend, but now are part of everyday life for the world population. The world has changed from keeping a Rolodex with lists of contacts to now knowing everything about everyone. It is easy to see what your friends ate for dinner, how fussy their kids have been and how they feel about Mondays. As much as people may decry the useless personal information that some people provide on social networks, most of them cannot bring themselves to cancel an account for fear that they'll miss out on important information.

Nowadays, there are social networks for all purposes. Some of them to share photos (i.e. Instagram), others to share general interests (i.e. Pinterest), others to share the professional path (i.e. LinkedIn) or to share all the contents together (i.e. Facebook, Google+). Also, there is a concept of popularity associated with each social network. However, the "popularity" term behind this is subjective because it depends on the position of the target audience. A clear example of this is the social network V Kontake, possibly unknown in our country but the most used in Russia surpassing Facebook [2]. Another similar example is the case of QZone, unknown most of the globe but dominant in China with an extremely high value of users [3].

Considering prior words, we must look at worldwide popularity and leave out the niches. Thus, the social networks that stand out above the others are:

- Facebook
- Google+
- Twitter
- LinkedIn
- Instagram

The number of active users is the criteria to order the previous list. This is a key aspect that reflects the popularity of the social network. Facebook dominates the market with nearly 50% of all the world's Internet users as active users [4].

In the context of the present internship, it makes sense to mention only social networks that provide instant messaging services. So, this section will focus only on Facebook, Google+ and Twitter. However, for an analysis of the remaining refer to **Appendix A - State Of The Art, Section. Social Networks.**

Facebook [38]

Users: 1190M monthly active [39]

App Store Rating (USA): 4/5 (2 756 123 ratings) [40]

Google Play Rating: 3,8/5 (10 273 306 ratings) [41]

In 2004, Mark Zuckerberg and four of his classmates, computer science students in Harvard University, created Facebook. Initially (named facemash) was intended to be an online

version of the university's "annual yearbook", compiling photos from female students and placing two side-by-side wondering which is the hottest girl. Due to its huge success, Facebook was quickly expanded to others Ivy League universities and, after that, Facebook began accepting members of any part of the world (minimum age 13+). Today it is one of the most successful companies worldwide and its current value is around 100 billion US dollars [42].

As previously mentioned, Facebook offers a highly diverse set of features. The main are [43]:

- Groups: "Share and keep in touch with the important groups in your life."
- Search: "Find people and content on Facebook."
- Events: "Organize gatherings, respond to invites, and keep up with what your friends are doing."
- Locations: "Share where you are and find friends nearby."
- Gifts: "Buy real gifts for friends to celebrate birthdays, new jobs and other big moments."
- Notifications
- Chat
- Useful SDK that allows developers to integrate Facebook with their applications in an intuitive way.

Nowadays, Facebook is available via web browser and has an official version for mobile devices (more precisely iOS and Android) which led to higher growth that is translated in the following image [44]:

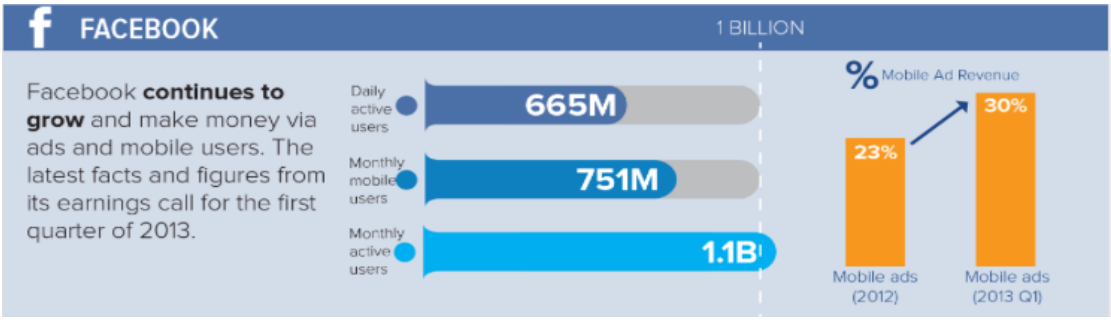

Figure 4. Facebook Facts

Google+ [45]

Users: 359M monthly active [44]

App Store Rating (USA): 4/5 (53 553 ratings) [46]

Google Play Rating: 4,1/5 (614 078ratings) [47]

Google+ (or Google Plus) was created by Google Inc. and was released in the late June 2011. Launched for field tests, the service was invite-only thereby limiting users of the same due its huge demand [48]. Days later, every user had 150 Google+ invites, expanding the

network to finalize the field tests. Also in the testing scenario, three weeks after the initial launch, Google+ had 20 million users, extremely high value which is translated into a great acceptance by the general public [49]. Nowadays, two years after Google+ release (September 20th, 2011), it has around 359M monthly active users. This value is much smaller compared to Facebook, however Google is not just focusing on building the largest social network. Google+ is also an important tool that helps the company identify and authenticate users across all its services, including search, Gmail and YouTube

The main features of this Social Network are [50]:

- Google+ Circles: "Circles allow users to group followers according to interests or any other criteria you wish to assign (for example, current customers, prospects or competitors)."
- Hangouts: "Think of a hangout like an instant chat room that you control. Creation of hangouts allows you to schedule online group events, such as a live Q&A session, that allows multiple people or entire groups to participate."
- Google+ Communities: "Google+ Communities allow you to create, or participate in, focused groups of both individuals and companies who share a particular interest."
- Insights: "Google+ recently introduced analytical insights that allow you to see key stats from your Google+ dashboard including how many views your business listing has received, how many times you've been viewed in local search and how many people are engaging with you."
- Google Places: "Google Places are separate business listing pages that are perfect for local businesses which allow you to display reviews, photos, key business information like opening times and contact details, as well as displaying a fully integrated Google location map."

Nowadays, Google+ is available via web browser and has an official version to mobile devices (more precisely iOS and Android). Between Q2 2012 and Q1 2013 (Q means quarter) Google+ registered a strong growth with its active users growing by 33% [44]:

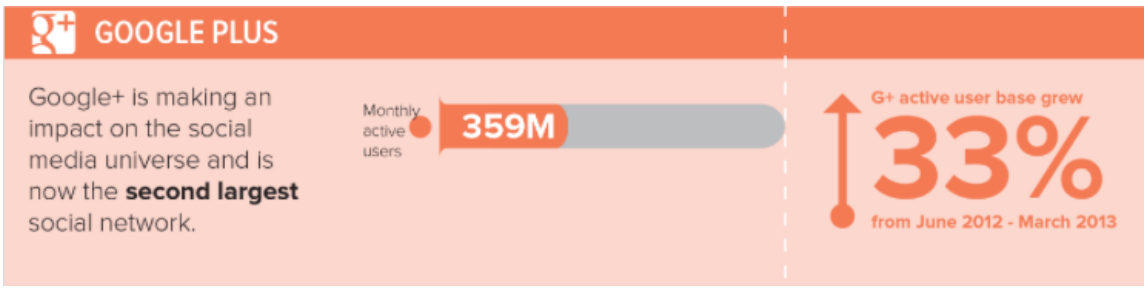

Figure 5. Google+ Facts

Twitter [51]

Users: 232M monthly active [52]

App Store Rating (USA): 3,5/5 (256 989 ratings) [53]

Google Play Rating: 4,0/5 (1 501 156ratings) [54]

Twitter is a Social Network that allows users to communicate via tweets (text messages limited to 140 characters) updating what is going on in their lives along with links to things they think are interesting, funny, or useful to their followers ("following" being essentially what "friending" is on other social networks). In July 2006, Jack Dorsey, Evan Williams, Biz Stone and Noah Glass released Twitter and today, it has about 913M registered users [55] in which 232 are monthly active users.

The concept of friendship works differently from most of the social networks as mentioned above. Twitter users choose who they do and don't follow. They have total control of what news they receive on their homepage.

Another key factor behind Twitter's success is the fact that this social network is widely adopted by celebrities. The easier way in which they can write tweets to promote their image is decisive in the choice of the social network, thus bringing more users [56].

The Twitter's main features are [57]:

- Tweets: "A tweet is a piece of text no longer than 140 characters. Spaces and punctuation count. Think of it as a blog entry, a bitesized blog entry."
- Following and Followers: "Following is Twitter's word for Subscribing or Friending."
- Conversation (or instant messaging): "Direct Messages are private. Only the sender and recipient can see them."
- ReTweets (RT): "ReTweeting is when someone repeats someone else's tweet, so their own followers can see the original message."
- Hashtags (#hashtags): "Hashtags let you add categories or keywords to your tweet. Using hashtags allows people to aggregate all the tweets on a subject. They consist of the hash sign, #, and a keyword with no spaces."

Nowadays, Twitter is available via web browser and has an official version to mobile devices (more precisely iOS and Android). We can also post Tweets by SMS without having the application on the mobile device. Figure 6 summarizes the current success of Twitter [58]:

Figure 6. Twitter Facts

Conclusions

The chat services provided by the social networks are very important. They connect lots of people worldwide and it is a plus to integrate such a service with an external application like the joyn product developed by WIT Software, SA. However, it is necessary to make choices concluding which social network would make sense to integrate. The three distributed systems provide the capability to make conversations with other friends but they are mainly different in terms of active users and purposes of use. So, it was necessary to perform an additional study with the goal of choosing one of them to include in the joyn product.

The next three images show the top 10 activities that users have on these networks registered in Q2 2013 [59]:

	Millions Active Facebook Users	Millions Active Facebook Users	Millions Active Facebook Users
Uploaded and share photos	476.59	239.17	108.37
Messaged with friends on a one on one basis	374.55	216.23	82.86
Commented on a friend's post	388.08	201.82	94.44
Commented on a friend's photo or video	359.53	184.84	86.55
Posted comment about my daily activities	349.60	184.71	87.86
Clicked Facebook 'like' button	348.16	150.64	84.87
Followed a group or like a page created by a brand	291.99	110.60	66.94
Watched video clips created by other internet users	310.76	125.06	75.58
Shared a link to an article	287.82	112.19	67.37
Shared videos created by other internet users	259.86	93.31	54.64

Figure 7. Facebook chat service usage

Figure 7 shows that Facebook chat (messaged activity on the image above) is very popular and is just behind the "commenting posts" and "upload and share photos" activities.

In the case of mobile devices (internship scope) this activity lies in the second place indicating that it is one of the most used in this social network.

Figure 8 is related to Google+:

	Millions Active Google+ Users	Millions Active Google+ Users	Millions Active Google+ Users
Uploaded and share photos	206.74	103.54	60.20
Clicked a Google+ '+1 button'	129.43	63.78	39.46
Checked in at a location	112.06	71.85	37.51
Watched video clips created by other internet users	124.89	55.54	39.52
Messaged with friends on a one on one basis	131.18	93.82	43.82
Shared links to blogs	136.42	66.88	39.25
Watched full-length TV or films	121.95	46.08	39.94
Shared a link to an article	127.44	62.95	41.43
Commented on a friend's post	141.83	80.10	49.33
Posted comment about my daily activities	135.12	76.24	49.07

Figure 8. Google+ chat service usage

Google+ service chat (messaged activity on the image above) is one of the most used in this social network. However, it does not stand out as much as the service on Facebook and there are plenty of activities with a very similar number.

In terms of mobile devices, the prospect is already different. It's the second most popular social network activity indicating that a possible integration with an application would be an alternative to consider.

Figure 9 is about Twitter:

	Millions Active Twitter Users	Millions Active Twitter Users	Millions Active Twitter Users
Posted comment about my daily activities	140.86	94.15	51.63
Uploaded and share photos	171.75	96.24	55.23
Commented on a friend's post	126.26	84.71	46.43
Messaged with friends on a one on one basis	115.12	87.04	38.33
Shared links to blogs	118.08	65.60	40.86
Shared a link to an article	109.15	61.26	39.10
Followed a group or like a page created by a brand	99.31	52.96	39.01
Commented on a friend's photo or video	107.36	70.08	42.39
Shared videos created by other internet users	97.08	53.10	34.91
Shared photos from a company or brand	104.31	59.61	34.03

Figure 9. Twitter chat service usage

As verified, the number of users that use the chat service is quite similar to Google+. However, taking into account the description given about Twitter, this social network is intended for another type of communication/utilization and an integration of an existing application with is not a priority.

To complete this section, a general conclusion on the choices is left:

- The high number of users that adopted Facebook Chat and actively use it reveal the huge success of this service. It is significantly more successful than others and the difference isn't just on one platform but in all (web, mobile phones and tablets).
- Another aspect is the fact that the purpose of this internship is to integrate an existing application (iOS) with Internet Chat Services. As seen above there are many services available, however the most promising one is Facebook.

Finally, the best thing to do is to direct us by the most promising, considering the target audience. If we follow the social network with a more active use of the chat service, the probability of success is high in relation to other social networks. Therefore, the chosen social network to integrate is the Facebook. So, instead of implementing services that would be slightly used in a single platform, it may be decided to implement the service in the Android platform before.

3. Approach

The internship followed a software development methodology already adopted by the company. This chapter starts by describing this methodology. Afterwards, the development plan defined in the first weeks of the project is presented, followed by their current status and deviations of the original plan. This section ends with an enumeration of the risks associated to the project development that will be presented and detailed as well as mitigation plans to these risks.

It is important should be mentioned that a more detailed analysis can be consulted in the **Appendix B – Approach.**

3.1 Methodology

WIT Software, SA has been adopting an incremental and iterative development process based on agile methodologies. This opposes to the Waterfall model, where a strict and complete planning of all development phases is done at the beginning of the project. Since humans are not perfect at estimating the time needed to complete a task, the existence of possible failures that lead to delays regarding what was planned would be expected. Unexpected situations, which were not thought before (i.e. process becomes unpredictable when new tools and technologies are used) and change the planned, may also occur. So, this adoption is in response to the frequent changes in requirements made during a real project. It also allows a better monitoring of the project progress by performing successive evaluations during development and thus enabling possible adjustments that are necessary.

In the current project, iOS integration with Internet Chat Services, an agile methodology, named Scrum, was adopted. Scrum is a different approach of planning, where the team can adjust the priority of the requirements along the project, considering possible changes.

Although Scrum is a simple methodology to understand, it has several principles that will be explained in the following subtopics.

3.1.1 Scrum Roles

The Scrum methodology consists of three different roles [60]:

The Scrum Team should be autonomous and self organised. They are responsible for estimating the size of requirements driving from a tactical perspective and making their own design and implementation decisions. They track the progress of their own work with the guidance of the Scrum Master and the team commit to delivering increments of software being accountable to the product owner for delivering as promised.

The Scrum Master is responsible for coaching and guiding the team, creating a trustful and inclusive environment, facilitating team meetings and negotiations with the product owner and removing organisational impediments to the team. He ensures that the process moving forward and the values and principals of scrum along with its framework are followed.

The Product Owner is the final authority on the requirements for the product. He manages the end user and stakeholder's expectations, prioritizing the product backlog,

release planning and providing clear and testable requirements to the team. He also collaborates with the team, end users and stakeholders ensuring that the goals are met and they accept the software at the end of each sprint.

3.1.2 Process

The Scrum process begins with the preparation of the **Product Backlog** by the Product Owner. The Product Backlog is a set of requirements to be achieved within the scope of the project and they are presented in the form of user stories[3] with estimation points. The user stories are used to give a less technical sense of the requirements and the team members establish the estimation points through Planning Poker[4].

After defining the Product Backlog, the scrum master and the team set a period of time called **Sprint**. This is usually a period between 1-4 weeks where the team is committed and focused, developing a set of features (defined in the Product Backlog). At the end of the Sprint, an increate to the final product is expected.

A **Sprint Backlog** is prepared at the beginning of each sprint. This is a set of features taken from the Product Backlog with an associated value that is the sum of each feature points. This value indicates the number of points that the team is committed to perform and can be called **Velocity**. Each member of the team has their personal speed, in other words, is able to perform X points during a sprint. Over 2/3 sprints, the Velocity tends to become constant and thus makes it easier to understand what can be done over a sprint, estimating the progress of the project more accurately.

On each day of a sprint, the team holds a daily Scrum meeting called the **Daily Scrum**. These meetings are typically held in the same location and at the same time each day and are strictly time-boxed to 15 minutes. The daily Scrum meeting is not used as a problem solving or issue resolution meeting. Issues that are raised are taken offline and usually dealt with by the relevant subgroup immediately after the meeting. During the daily Scrum, each team member answers the following three questions: What did you today? What will you do tomorrow? Are there any impediments in your way?

In the final of each sprint, an increate to the final product is expected. This means that the team had produced a usable increment in the software that could be shippable. So, at the end of each sprint, a **Sprint Review Meeting** is held. During this meeting, the Scrum team shows what they accomplished during the sprint. Typically this takes the form of a demo of the new features. Participants in the sprint review typically include the Product Owner, the Scrum team, the Scrum Master, management, customers and developers from other projects.

Figure 10 illustrates all the process previously mentioned [61].

[3] In software development and product management, a user story is one or more sentences in the everyday or business language of the end user or user of a system that captures what a user does or needs to do as part of his or her job function, in other words, user stories are a quick way of handling customer requirements without having to create formalized requirement documents.

[4] Planning poker, also called Scrum poker, is a consensus-based technique for estimating, mostly used to estimate effort or relative size of user stories in software development.

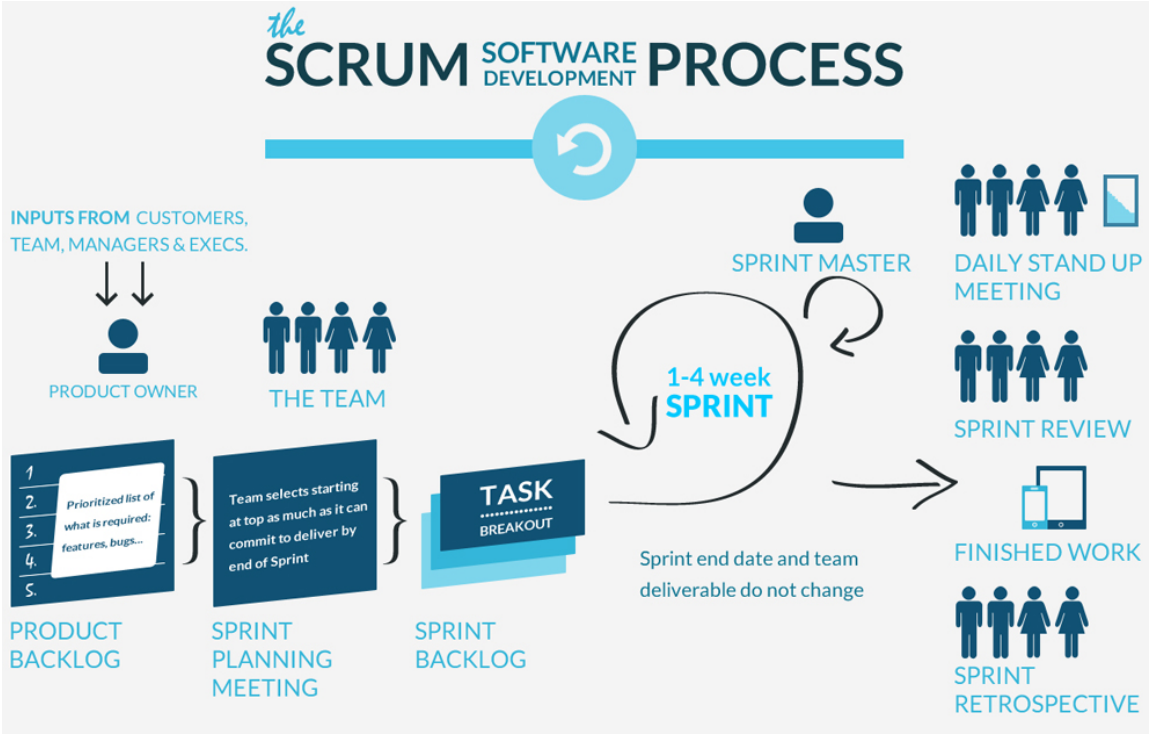

Figure 10. Scrum Process [62]

3.1.3 Definition of Done

Another key point in the development using Scrum, is the Definition of Done (DoD). It's considered as a tool for bringing transparency to the work that a Scrum Team is performing and is related more to the quality of a product rather than its functionality. When a Scrum Team works in order to develop a product, it is crucial to understand when a task is completed and can be called "Done".

Normally, the DoD is a clear and concise list of requirements that a software increment (functions, modules, etc.) must meet. Until this list is satisfied, a software increment isn't done.

The DoD has been defined at the beginning of the project along with other trainees in the same area. It is very useful because it enables the team to know how much work should be selected for a given Sprint Backlog. Furthermore, a common DoD helps to:

- Baseline progress on work items
- Enable transparency within the Scrum Team
- Expose work items that need attention
- Determine when an Increment is ready for release

A personal DoD was designed with the contribution of the WIT Software's trainees in the same area as this internship. For more details, refer to **Appendix B - Approach, Section. Definition of Done.**

3.2 Planning

As previously mentioned, the adoption of an agile methodology means that a complete and rigid plan will not be done at the beginning of the project. So, it is real important to explain how the planning is built in order to clarify the readers as the work is done.

First, it was necessary to define the roles of Scrum. In the scope of the project, the three roles that Scrum sets are being used. Two people are carrying these roles:

- **Scrum Team:** Tiago Pereira
- **Scrum Master:** Tiago Pereira
- **Product Owner:** Frederico Lopes

Once the Product Owner is the internship supervisor and only the trainee composes the team, the role of Scrum Master is also performed by the trainee. As we are in an internship environment, and it is the trainee who communicates directly with the supervisor, facilitating team meetings and negotiations, it makes perfect sense to play the role.

The trainee also contributed to the development of the Product Backlog. It was defined by him and subsequently prioritized by the Product Owner.

In this project, the Sprint period is two weeks and a Sprint Review meeting is made at the end of that time. The Team/Scrum Master and the Product Owner attend this meeting and it usually has duration of two hours consisting in a presentation about what was done in the previous Sprint Backlog and what will be done in the next. The Product Owner is responsible for giving feedback to the performed User Stories (requirements) and the User Stories that were intended for developing.

For more information about the Product Backlog and Sprint Backlogs refer to **Appendix B - Approach, Section. Product Backlog/Sprint Backlog.**

Since this internship is inserted in a field where there is tremendous competitiveness, the development of two-week plans allow the introduction of new requirements through the project, something that would not be possible if a Waterfall model had been taken. These requirements may have urgent priority and may be requested by customers, which means they have to be made before they were planned. If an agile methodology weren't followed, the project planning would be completely ruined.

Until the end of this internship sixteen sprints were performed. The following list describes these sprints. However, this description is high-level, for more information them refer to **Appendix B - Approach, Section. Product Backlog/Sprint Backlog.**

- **Sprint #0:**
  Date: 29-10-2013 to 12-11-2013
  This was the warm-up sprint and it was reserved to perform the competitors' analysis, reformulate the product backlog and some features regarding the prototype application.
- **Sprint #1:**
  Date: 12-11-2013 to 26-11-2013
  In this sprint the application *mockups* and use cases were defined. The prototype development was also begun in it.

- **Sprint #2:**
  Date: 27-11-2013 to 10-12-2013
  This sprint was essentially marked for Chat implementation on the prototype application.
- **Sprint #3:**
  Date: 11-12-2013 to 24-12-2013
  The prototype was closed in this sprint. Some use cases were reformulated according the feedback collected from the prototype.
- **Sprint #4:**
  Date: 26-12-2013 to 07-01-2014
  An impediment due to festive season (Christmas and New Year) was emerged. The trainee needed answers that could only be given by other engineers who were on vacation, thus failing to perform the planned tasks.
- **Sprint #5:**
  Date: 08-01-2014 to 22-01-2014
  The first version of the architecture document was written in this sprint. The Share on Facebook feature using the Facebook SDK was implemented.
- **Sprint #6:**
  Date: 22-01-2014 to 05-02-2014
  This sprint took a large time period and it was intended to perform all the internship's documentation such as the interim report and its appendices.
- **Sprint #7:**
  Date: 05-02-2014 to 04-03-2014
  This sprint aggregates two different sprints and was entirely reserved to perform taks regarding the Share on Facebook feature. This allocation was done because the feature was very important for a demo made by the company in Barcelona and it was important to have it at 100% for that.
- **Sprint #8:**
  Date: 05-03-2014 to 18-03-2014
  Share on Facebook reformulation in order to be performed using the native mechanisms. It was reformulated for both platforms Android and iOS.
- **Sprint #9:**
  Date: 19-03-2014 to 01-04-2014
  This sprint was intended to perform all the features related to the Remote contacts, including their social information and their presence status.
- **Sprint #10:**
  Date: 02-04-2014 to 15-04-2014
  Use cases reformulation in order to be presented with the new iOS skin. This sprint was also intended to setup the environment in order to start the modifications in the communication layer.
- **Sprint #11:**
  Date: 16-04-2014 to 29-04-2014
  The first modifications in the communication layer started in this sprint. The external framework that is responsible for the connection to the Facebook Chat was included in it.
- **Sprint #12:**
  Date: 29-04-2014 to 13-05-2014

Some of the chat's functionalities were implemented in this sprint such as *isTyping* notifications and the sending of emoticons/emojis. This sprint had an unexpected task that consisted in a presentation to the company CEO, indicating the internship progress.

- **Sprint #13:**
  Date: 14-05-2014 to 27-05-2014
  The chat functionality was closed in this sprint. The File transfer feature was also started in this sprint, creating the necessary entry points to perform it.
- **Sprint #14:**
  Date: 28-05-2014 to 10-06-2014
  The file transfer functionality was closed in this sprint. All the tasks involving the development of code were also closed in this sprint.
- **Sprint #15:**
  Date: 11-06-2014 to 27-06-2014
  This was last internship's sprint and the last four user stories were closed in it. It was reserved to perform all the teaks regarding the internship's documentation including the final report and its appendices.

Another aspect that has been talked about is the fact that the Product Backlog is a repository of requirements. The requirements are detailed in user stories and have associated effort points and priorities.

In the Scrum methodology, the traditional requirements document isn't used. Instead, a Product Backlog is prepared and details the high-level requirements, allowing the team to consult it along the project. So, in the internship scope, the requirements document is the product backlog and this can be seen in the Table 3.

Story #	Category	Priority	User Story
…	…	…	…
6	APIs and Protocols	Very Low	Google: As a trainee I want to know how Google API works to integrate my application with the social network.
7	Authentication	High	Facebook: As a user I want to authenticate with my Facebook account.
8	Authentication	High	Facebook: As a user I want to logout from my Facebook account in order to disconnect it from joyn.
9	Authentication	Very Low	Google: As a user I want to authenticate with my Google account.
10	Authentication	Very Low	Google: As a user I want to logout from my Google account in order to disconnect it from joyn.
11	Chat	High	Facebook: As a user I want to make single conversations with my Facebook contacts.
12	Chat	Very Low	Google: As a user I want to make single conversations with my Google contacts
…	…	…	…

Table 3. Subset of the project requirements

Table 3 is just an excerpt of the 104 defined requirements. It is also important to note that these are sorted alphabetically by category. These categories will meet the high-level requirements presented in the internship proposal and group the user stories that are part of these requirements. The initial high-level requirements in the proposal are:

- Study the protocols and extensions of the Internet Chat Services (Facebook Chat, Google Talk, …)

- State Of The Art elaboration
- Requirements analysis
  - ⇒ Product Backlog
- Prototyping some functionalities
- Solution Architecture
- Development (1st Semester):
  - ⇒ Authentication
  - ⇒ Loading the remote contact list
  - ⇒ Chat
- Demo for demonstrations
- Interim internship documentation
- Development (2nd Semester):
  - ⇒ File Transfer
  - ⇒ Group Chat
  - ⇒ Presence and Status
- Functional and non-functional tests
- Final demo for demonstrations
- Final internship documentation
- **Share on Facebook (new feature)**

In the Backlog preparation, some requirement names were changed and other requirements were grouped into a single category. The categories composing the backlog are as follows:

- **APIs and Protocols:** consists in the "Study the protocols and extensions of the Internet Chat Services (Facebook Chat, Google Talk, …)" requirement defined n the proposal.
- **Authentication:** corresponds to the "Authentication" requirement in the proposal
- **Chat:** corresponds to the "Chat" requirement in the proposal.
- **Documentation:** This category combines various requirements of the proposal – "State Of The Art elaboration", "Requirements analysis", "Solution Architecture", "Demo for demonstrations", "Interim internship documentation", "Final demo for demonstrations" and "Final internship documentation".
- **File Transfer:** corresponds to the "File Transfer" requirement in the proposal.
- **Group Chat:** corresponds to the "Group Chat" requirement in the proposal.
- **Presence and Status:** corresponds to the "Presence and Status" requirement in the proposal.
- **Prototyping:** corresponds to the "Prototyping some solutions" requirement in the proposal.
- **Remote Contacts:** corresponds to the "Loading the remote contact list" requirement in the proposal.
- **Social Network Utils:** corresponds to the "Share on Facebook" functionality.

As previously mentioned, the requirements are distributed across these categories with priorities. For a detailed analysis about the requirements refer to **Appendix B - Approach, Section. Product Backlog.**

Finally, it is important to emphasize that the "Functional and non-functional tests" category was not created in the Product Backlog. In an agile methodology, the tests are made throughout the project, unlike the Waterfall methodology that reserves a phase for testing. In Scrum, the testing phase is associated with each feature. Whenever a piece of software is created, it is tested up to be ready for demonstration. Figure 11 shows the difference between the two models.

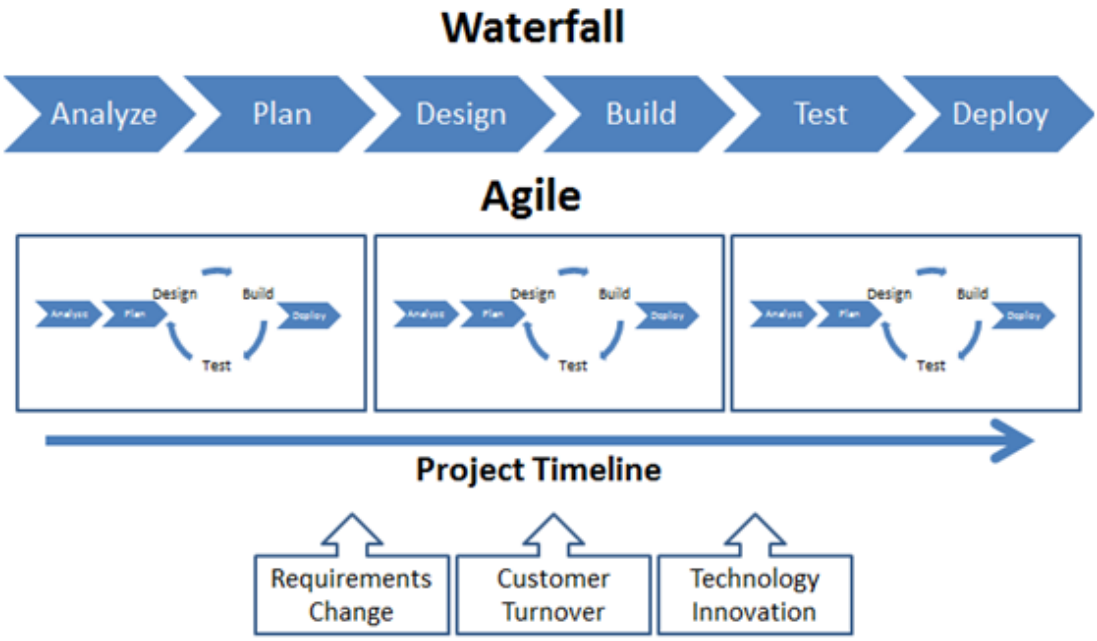

Figure 11. Agile vs Waterfall

3.3 Risks

There are always risks associated with a software project. These may have different sources but can all contribute to the failure of a project. Prevention and mitigation strategies for some risks may be prepared in order to minimize the impact that they can have on the project. These strategies should be reviewed periodically because we are in an era in which technology evolves very quickly and changes can happen.

At this time, the following list indicates the risks that are visible in this phase of the project and which were collected by drawing the Product Backlog. The list may grow as new analyses are made.

R.1. API Requests

Since the integration is done with an external API, the data are obtained through requests and those requests can be changed. If this happens, the requests made become invalid and the external server responds with error messages.

- Explanation: Social networks are constantly innovating and sometimes need to change some features to introduce new ones. There is always a small chance of this happening. For example, about three years ago Facebook completely changed their public API. All the requests made by third party applications had to be reformulated.

- Mitigation Plan: There is no effective plan to solve this risk. The product has to adapt to the changes made in the APIs.

R.2. Offline API

An API can be offline for two reasons: the first can be an overload of requests and the second is because it was removed to the public.

- Explanation: The best known social networks have millions of daily visits, which cause several third-party applications looking for integration. Sometimes the use of the API can be so exhausting that it becomes temporarily unavailable (offline). But sometimes, the API could be removed from the public, leading to the impossibility of carrying out any operations associated with the social network.
- Mitigation Plan: When the API is temporarily unavailable, the answers can be treated in order to not degrade the user experience. Obviously, the features are also unavailable. If the API is completely removed, there is nothing that can be done because there is no control over it. To remove the integration with their social network is the only solution.

R.3. Changes in the communication protocol

In the project, the Extensible Messaging and Presence Protocol is used for the communication and is defined by standards [63]. For some reasons the standards could change making the development code invalid.

- Explanation: There is a large community that sets the XMPP standards protocol. This community is constantly innovating, trying to optimize what already has and trying to add new functionalities to the protocol via extensions (XEPs) [64]. The communication is based on XML streams and these streams can be altered in their composition.
- Mitigation Plan: This risk is very small and is easily mitigated. When a change exists in the contents of a stream, only this stream must be modified, and the modification is fast and easy to be done.

R.4. iOS updates (could result in deprecated code)

Apple usually releases gradual updates for its operating system. When it comes to major updates (i.e. iOS 6 to iOS 7) a lot of stuff is modified internally (code level).

- Explanation: In order to optimize the earlier versions of the operating system, a refactor is made whenever a new version of iOS comes out making some code obsolete.
- Mitigation Plan: The application must be supported by iOS6 and above. Then, as good practice, all implemented code should be done taking into account the last refactor done to iOS7. Thus, the probability of becoming obsolete is much smaller.

R.5. Strong market competition

As shown before, this internship is part of an area where there is a strong market competition.

- Explanation: It is quite difficult to affirm any product in this market because of the high offer provided to the end user.
- Mitigation Plan: It is necessary to be at the forefront of innovation and maintain this property in the goals of the internship. For this purpose, is necessary to keep updated the set of features that they provide through a progressive study about the applications on the market.

4. Solution Architecture

A good architecture is a key point to have a development without problems. It is necessary to do a detailed analysis on how the features should be developed and integrated into the project, generating a document that will define guidelines for the future implementation. These guidelines should define the structure, behaviour and views of the system.

This type of information must be explained on an architecture document where a complete analysis is made. Other teams/developers can access it and easily understand how the system must work and how the features should be implemented. It is also important to mention that this document usually contains terms that are common in the world of computer engineering, assuming that the reader has some knowledge about them (APIs, Queries, Protocols, etc.).

An important aspect was the establishment of the Use Cases. This document serves to verify how the features should appear in the product and helps to notice what classes should be modified. To consult the mentioned User Cases, refer to the **Appendix C – Use Cases**.

Notice that the class diagrams relating to the product will not be shown in this section. As the project code is proprietary, all its information (classes, details of classes) cannot be published, following attached. This section is just a short summary of the system architecture showing how the product will communicate with external APIs. To consult an extensive analysis containing all the trainee's contribution please, refer to the **Appendix D – Solution Architecture**.

This chapter is divided into seven sections:

- **iOS Basics:** an overview on the iOS world and explains some basic guidelines that should be followed during the architecture establishment and development phase.
- **WMC environment:** an overview on the WIT's RCS product including the network infrastructure and its internal organization. It is important to understand how the product is organized in order to integrate the new features.
- **Authentication:** a brief overview on the Authentication feature architecture.
- **Facebook Share:** a brief overview on the Facebook Share feature architecture.
- **Remote contacts, presence and contacts' information:** a brief overview on the Remote contacts, presence and contacts' information features architectures.
- **Chat:** a brief overview on the Facebook Chat feature architecture.
- **File Transfer:** a brief overview on the File Transfer Chat feature architecture.

Note: the **Appendix D – Solution Architecture** also has a chapter explaining the APIs used. This chapter consists in a previous study that was performed in order to achieve the proper know-how that is needed to project the architecture and start the development phase.

4.1 iOS Basics

The main requirement of the application is the iOS support, which means that the application must run in iPhone and iPad devices. This internship consists of the integration of an external service in the company's RCS product, so it is expected to follow the same conventions during the development phase.

The source code is written in Objective-C using small pieces in C++ that make the connection to the communication stack used. The UI part follows an **MVC** (Model-View-Controller) pattern, having this pattern a fundamental role in several components and mechanisms of the Cocoa Touch framework that are the application base for iOS. This pattern is represented in the following image:

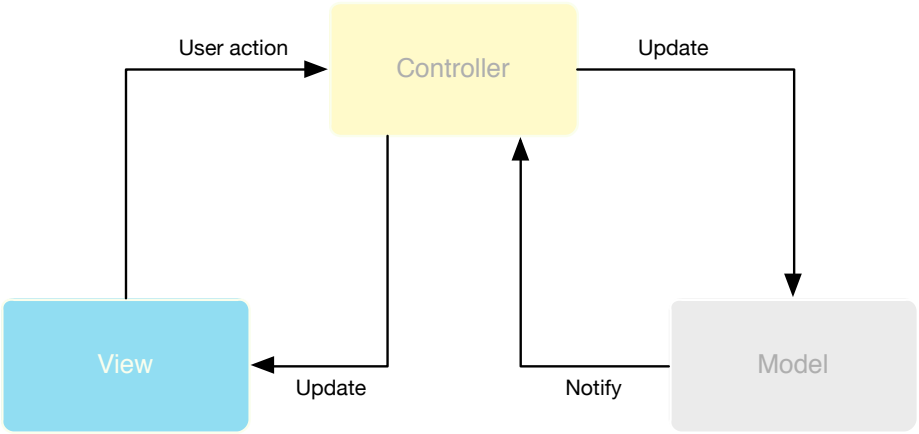

Figure 12. Cocoa MVC

Figure 12 illustrates the interactions between these three components.

• Model – objects that encapsulate the specific data of the application and define the business logic that manipulates and processes that same data.

• View – object that is seen and used by the user to interact with the application. Its main purpose is to display the data from the application's model objects and how to interact with them.

• Controller – object that acts as an intermediary between models and views. This way, controllers are responsible for notifying the views about modifications on the models, and vice-versa.

Given the support of two devices, iPhone and iPad, it becomes necessary to define strategies to promote code reuse as well as shared components between both devices. Since the internal architecture of the iPhone and iPad is the same, at model level we don't need to do anything because the isolation and centralization of this layer already ensures its full reuse for both devices. Another aspect to keep in mind is the set of application interfaces that are very similar between both devices, except the respective dimensions and the specific behaviours of each device. So, to promote the reuse of common components between devices we use inheritance mechanisms to reuse logic between controllers. The **creational** pattern **Abstract Factory** is also used in order to abstract the initialization of the

components of the two platforms. Using this pattern, the respective controller should be initialized in runtime, avoiding each controller from having information on which component should be initialized.

Apple uses a prefixes convention to name classes. This integration followed the same conventions because the product already uses it. Thus, all created classes have the *WMC* prefix.

Another important aspect is the fact that all the iOS applications that go to background have a period of time when they are still active. After that period, the application is considered as suspended. This happens because when a low-memory condition occurs, the system may purge suspended apps without notice to make more space for the foreground app.

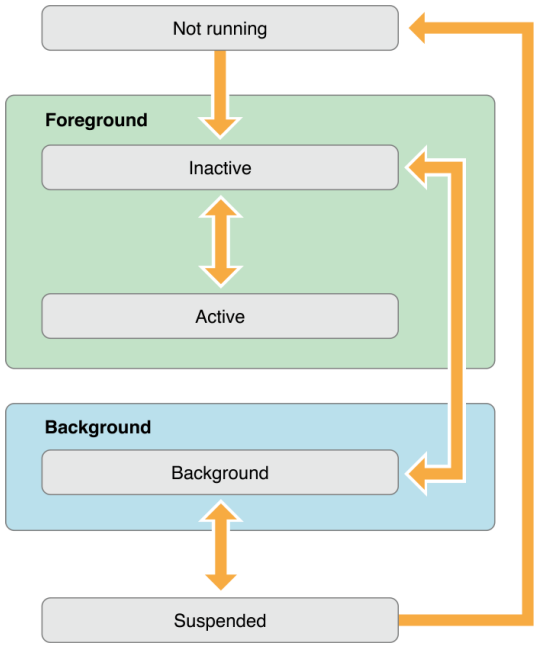

Figure 13. Application States [65]

According to this, there are five different application states:

• Not running – The app has not been launched or was running but was terminated by the system;

• Inactive – The app is running in the foreground but is currently not receiving events. An app usually stays in this state only briefly once it is in transitions to a different state;

• Active – The app is running in the foreground and is receiving events. This is the normal mode for foreground apps;

• Background – The app is in the background and executing code. Most apps enter this state briefly on their way to be suspended. However, an app that requests extra execution time may remain in this state for a period of time;

• Suspended – The app is in the background but is not executing code. The system moves apps to this state automatically and does not notify them before doing so. While suspended, an app remains in memory but does not execute any code.

So, it is necessary to have a special care dealing with sockets or HTTP requests that have an associated timeout when the app remains in background.

The iOS applications are known for offering good usability. To achieve this goal it is necessary to make all the API calls **asynchronous** in order to give the user a better UX. So, an **asynchronous** request is made and when the response arrives, it fires an event that notifies the interested party. All the operations related to this request are running in secondary threads that are called **queues** in iOS.

However, if we are talking about an **UI operation** (i.e. update a table, update a contact details view), the operation has to be performed in the main thread in order to see the results. Dispatching a block to the main queue is usually done from a background queue to notify that some background processing has finished (i.e. asynchronous API call). By iOS convention, the main thread is responsible for updating the UI. If we don't dispatch for the main thread the UI update is done in background and it will never be visible to the user. This way, in this project all the API calls are done in background so that the main thread doesn't get locked, and consequently, the UI stays fluid. When a response arrives we switch to foreground (main thread) in order to update the user view.

To perform the previous operations the **Grand Central Dispatch** (GCD) is used. That is a technology developed by Apple to **optimize** application's support for systems with multi-core processors and other symmetric multiprocessing systems.

4.2 WMC Environment

The RCS application provides rich contents by connecting to an IMS network as demonstrated in Figure 14.

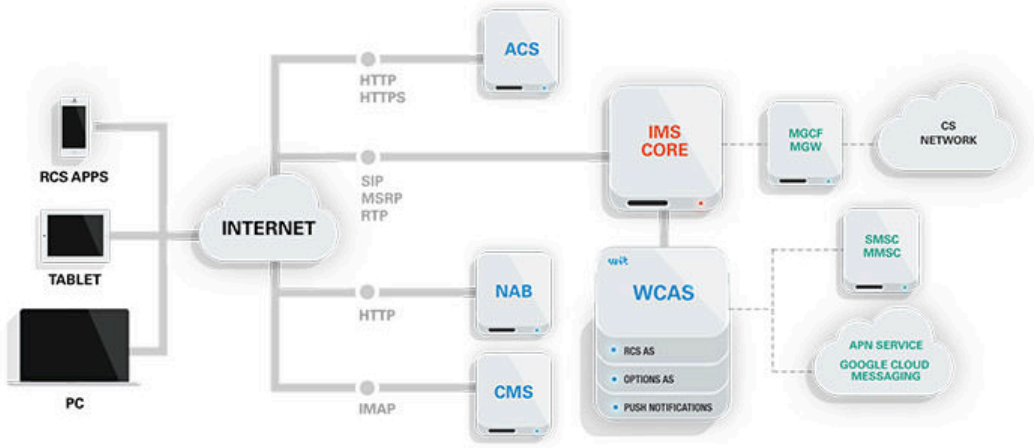

Figure 14. WIT's RCS app in an IMS Infrastructure

To develop a successful application in this environment the main concerns are the proper tools to communicate, and this is reflected in the structure of WIT Software's RCS solution. The application uses a communication stack designated as COMLib. Its main goal is to provide all the communications components that the application needs like messaging, voice and video calls, etc. Once there are several dependencies adjacent to this component, a lot

of libraries are required to process media or to provide security for the communications, which are also present in it.

COMLib is composed by several services that provide all the communication dependencies. It follows a Service-Oriented-Architecture (SOA) architecture in order to achieve a better organizational and configuration flexibility, something that could be difficult with other type of architecture. This communication stack is implemented in C++, which means that can be used by all the platforms of the RCS product (iOS, Android and Windows). However, to be used by these platforms the access needs to be wrapped in order to provide the integration with different programming languages. In iOS case, there is no need to do a wrapper to expose the code to the UI because we can call functions in C++ from an Objective-C file.

Such an application requires a strong interaction with platform-specific tools, such as the access to Internet connection state, the location sensors or the radio strength, and those vary across platforms. However, to keep the UI cleaner, the access to platform utilities is executed at the same level of the communications, and then for each platform a specific implementation applies.

On top of the COMLib is an UI layer that represents all the views and components visible to the end user, the controllers for those views and a set of managers to execute the tasks, which involve interactions with the stack. Figure 15 shows an illustration of the global architecture of the application as it was before the modules created along this internship:

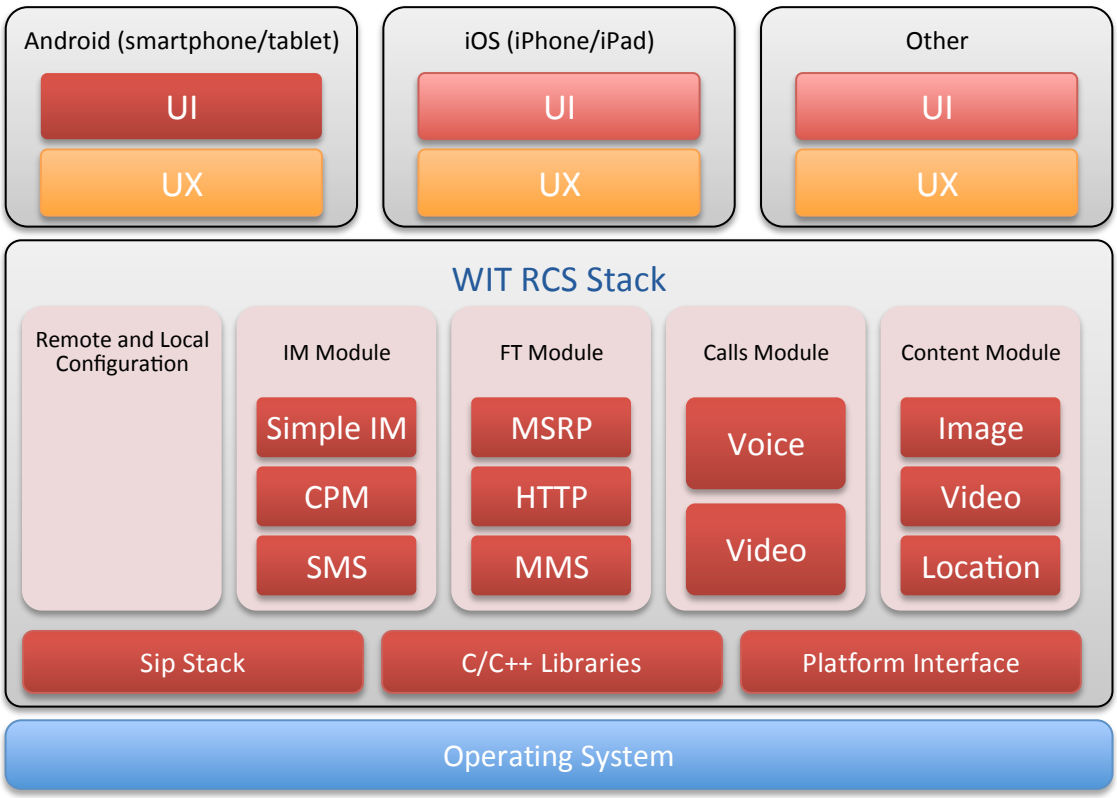

Figure 15. High-level architecture of the communication stack

As mentioned above, it is possible to clearly identify two distinct layers of the application on top of the operating system. Regarding the COMLib module, the sub-layer that provides interaction with the UI/UX is the COMLib API. It composes a set of different APIs and is

the entry to access the different modules of the application, and to register callbacks in order to receive information when some event is triggered inside those modules. In the particular case of iOS application, this API is composed by C++classes.

Thereafter we have the application modules, providing all the core features for communications. For simplicity, some modules are omitted in the previous figure, once they are out of the internship's scope. Along with the communication modules, there is a database access module that is responsible for store all the exchanged data between peers, including messages, file transfers, etc. There is also a module in this layer responsible by the application configurations, which sets the modules that should be running on start up, and also to manage eventual remote configurations that can be received when connecting to the application server, in order to change the application services accordingly to the respective client.

The bottom layer of the communications library is composed by the dependencies of the above-mentioned modules, which are mainly composed by C and C++libraries, a SIP module to provide the communications and also the platform interface, which defines all the platform specific accesses that the application needs to run properly. This interface is then extended regarding the platform in which the application is running.

Regarding the UI/UX layer, the best way to show the internal architecture is to give a high-level package diagram of the code. So, the next figure illustrates the main groups that are target in this internship:

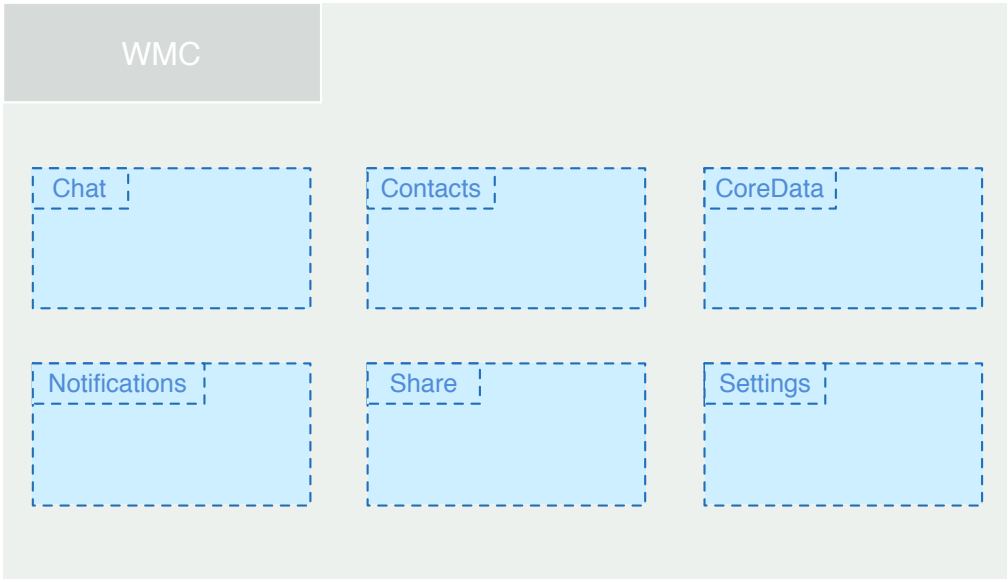

Figure 16. Package diagram including the application main packages

In most of the cases, the packages in this layer are divided in four different sub-groups: controllers, managers, models, and views. Some may ask why the packages are divided in four groups instead of the three corresponding to the MVC. The answer is simple: managers are controllers, but they are considered a different type for two reasons. First of all, they are the only objects that interact directly with COMLib in order to retrieve information; secondly, they are instantiated only once upon the launch of the application – singletons – in order to be accessed by "normal" controllers via their shared instance. This way, it is avoided the assignment of a specific instance to each class that needs to access that manager, which would be very inefficient due to the size of this project.

The **Chat** group also contains a set of sub-groups, which connects the UI to the manager, which handles the tasks that require interacting with COMLib, such as sending a message, or deleting a conversation. This interaction is done by the inclusion of data sources on chat manager who in turn interact with the COMLib. These controllers also go in their specific folder.

The **Contacts** group is responsible for all the contacts management in the application. There are two different data sources to get contacts: native address book and network address book. The four default sub-groups plus operations are responsible to trigger events that will modify the address book and managers compose this group.

The **CoreData** group provides generalized and automated solutions to common tasks associated with object life-cycle and object graph management, including persistence. This is composed by a .xcdatamodeld file that is the database model as diagram, sub-group models representing the objects that can be persisted and managers that implement all the logic needed to persist and retrieve objects from the database.

The **Notifications** group is responsible for all the notifications that the application shows. This group implements a **protocol,** which declares the methods expected to be used for a particular situation. All the classes that include this protocol can automatically use its methods.

The **Share** group is responsible to manage all the accesses to the external files in order to be shared in the application. It provides a set of methods that allow a user to get files from Gallery/Cloud and share them on a chat conversation.

The **Settings** group is composed of all the objects, which compose the application settings, and all its views. Additionally, there is a specific manager controlling all the events and actions related to the application settings in order to achieve the desired behaviours.

The next sections will explain all the above modifications in a deeper level of detail, including the interaction between the classes modified and all the logic behind it.

4.3 Authentication

In order to deploy features that depend on third-party services, it is necessary to implement the authentication mechanisms in order to get access to the user's account of that service. Regarding the proposed features, it is necessary to implement authentication mechanisms for all the social networks and cloud storages that will be integrated with this RCS application. The present scope of the internship is focused on **Facebook** and **Dropbox**. Both these services provide official SDKs for iOS, which contain a set of methods that ease the interaction with the respective services.

As the architecture for the authentication with these two services is very similar, the present section will merge both services in one architecture design for the sake of simplicity.

The authentication with both Facebook and Dropbox was developed mainly at the UI/UX level. The entry point for authentication is through the application settings, as illustrated in Figure 17. Once social networks and cloud storages are different kinds of service, two different menus were created in order to wrap them accordingly.

Figure 17. Authentication flow for both Dropbox and Facebook

The application settings are populated with the content of the AppConfig.xml file. It contains a set of identifiers, each of them accompanied by an active parameter that allows enabling/disabling each setting easily.

These identifiers are assigned to a setting type according to the intended behaviour. The application already has a set of setting models that have its specific behaviour and design.

The diagram representing the final state of the architecture for the feature is illustrated in Figure 18. This functionality was developed mainly at the UI/UX level. It is important to note that this diagram is high-level in order to respect the product's company privacy. For a detailed diagram that explains all the trainee's contribution, refer to the **Appendix D – Solution Architecture: Authentication**.

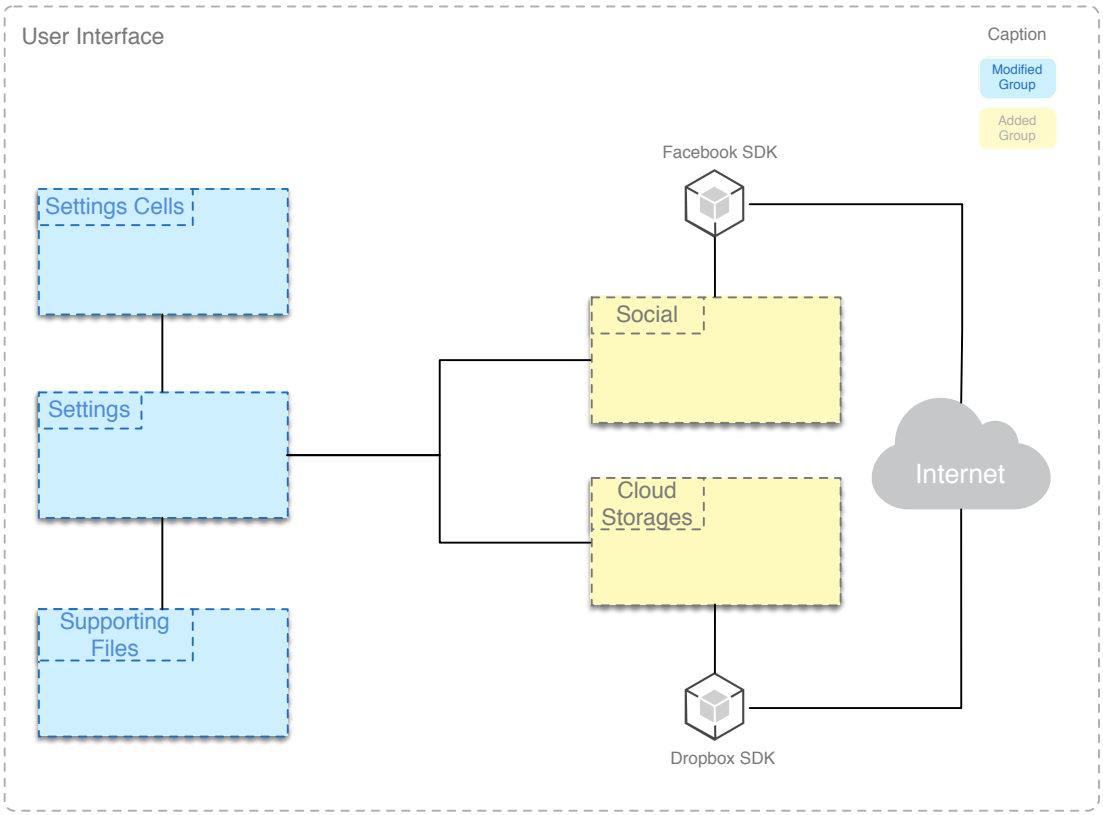

Figure 18. Dropbox and Facebook authentication architecture

In order to add Cloud Storage and Social Network settings, two new identifiers were created and added to the **XML** file used to set the active settings. Furthermore, these identifiers were assigned to **a custom cell** model because it is intended to run a custom block implementation that contains all the necessary logic to connect the application with both Facebook and Dropbox SDKs, as well as to display a specific view for each service. The content displayed by this view varies according to the session state, i.e., the view presents a login button if there is no valid session, or presents a set of service options in case there is a valid session.

4.4 Facebook Share

This feature was not initially planned and was developed in order to be demonstrable in the Mobile World Conference in Barcelona, in late February. In short, it consists in a quick share of received media on Facebook. This media is received in a chat conversation and can take many formats. The scope of the internship was the **RCS iOS application** but the trainee also developed this feature to the **RCS Android application**.

It is important to note that this feature had two types of implementation. First, the Facebook SDK was used to develop the feature, but a new idea came up consisting of using the native mechanisms of operating systems to perform the share. When this idea came, the first implementation using the SDK was already completed and it allowed a user to:

- Share images (only) in 1-to-1 and Group Chats;
- View a Facebook icon near the chat balloon that contains the image, indicating that it was shared.
- View the image likes and comments near the indicator;
- Tap the indicator and open the image in the Facebook application (if he has it) or Safari for iOS.

Then, before making a second implementation, it was necessary to measure the advantages and disadvantages of each one, as Table 5 shows.

SDK implementation

Pros	Cons
It is possible to view the image likes and photos after a share.	Worst UX.
It is possible to open the post through the application by tapping the Facebook icon.	It is impossible to choose which album the user want to upload the photo.
The shares indicate the application used to share the resource on Facebook. (e.g. via RCS App)	It is only possible to share images.

Table 4. Facebook Share via Facebook SDK - Pros vs. Cons

Native implementation

Pros	Cons
Better UX.	It is impossible to view the image likes and photos after a share.
The user can choose which album want to upload the photo and can associate a location to that upload.	It is impossible to open the post through the application by tapping the Facebook icon.
The user can also choose the target audience that can view the upload (privacy concerns).	The shares don't indicate the application used to share the resource on Facebook. (e.g. via RCS App). Since the native mechanisms are used to perform the action, the share appears via iOS.
It is possible to share text and video in addition to the images.	

Table 5. Facebook Share via Native mechanisms - Pros vs. Cons

The advantages of an implementation are basically the disadvantages of the other. The possibility of visualizing the photo comments and likes are a really nice detail that enriches the UX. However, the possibility of sharing various resources formats is a must and completely demolishes the implementation detail provided by the SDK. Another key aspect is the fact that the native mechanisms provide a better UI for the user to perform the share. This UI allows the users to choose the target album to upload the photos (in this case) and to select the target audience that can see the resource published (this latter is applied to all kinds of formats). After a detailed analysis of the advantages and disadvantages of each implementation, it was decided to implement the **native mechanism** [Figure 19].

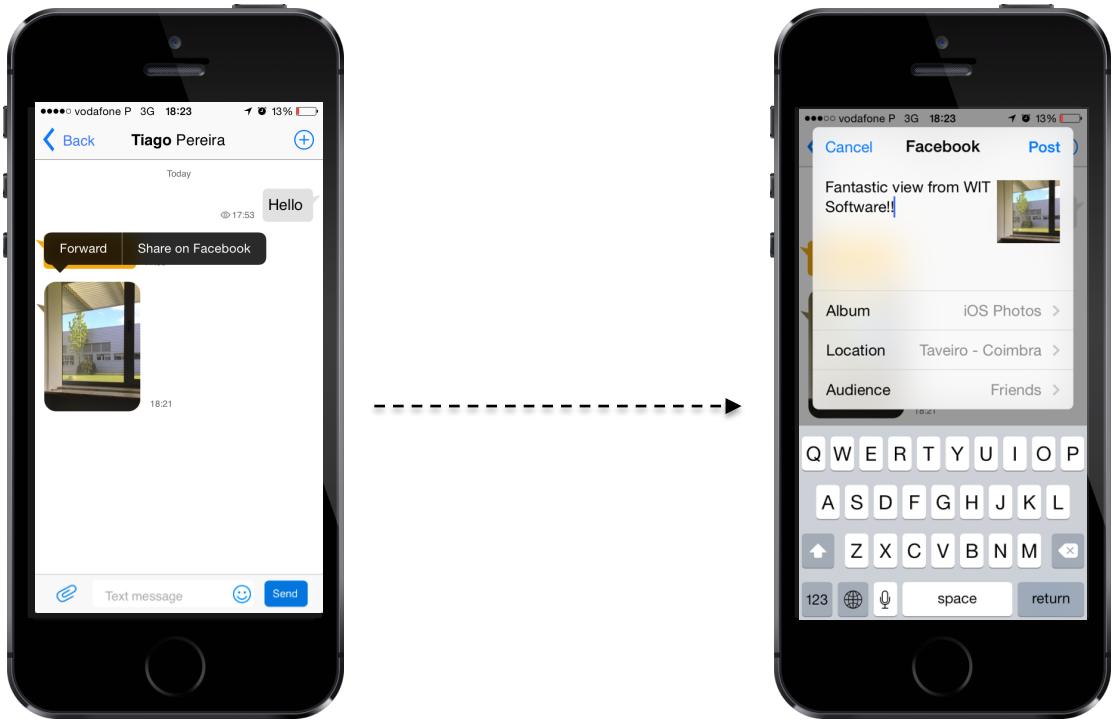

Figure 19. Facebook Share: native mechanism

The diagram representing the final state of the architecture for the feature is illustrated in Figure 20. This functionality was developed mainly at the UI/UX level. There was no need to exchange the communication stack in order to achieve this feature. It is important to note that this diagram is high-level in order to respect the product's company privacy. For a detailed diagram that explains all the trainee's contribution, refer to the **Appendix D – Solution Architecture: Facebook Share**.

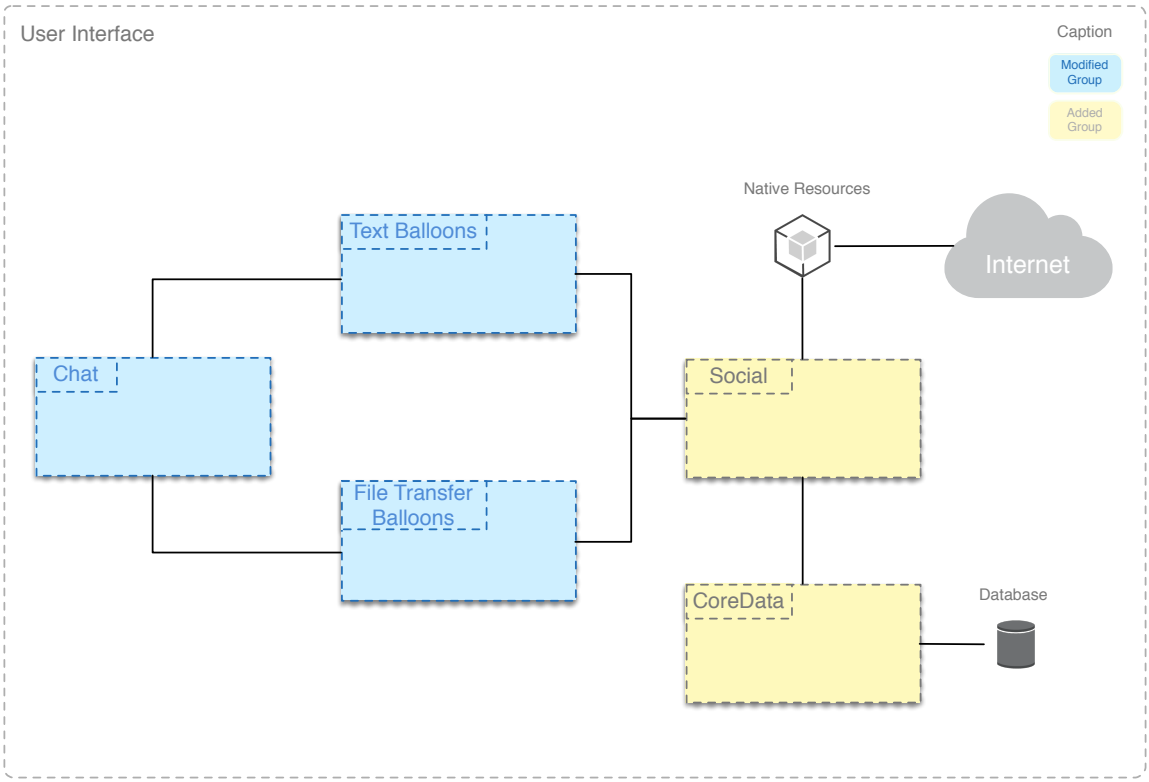

Figure 20. Facebook Share final architecture using native resources

Through the previous diagram, it can be seen that this feature was mainly developed at **Chat**, **Social** and **CoreData** levels. Once this feature is triggered in a chat conversation (1-to-1 or group chat) there was a need to modify some classes in order to achieve the desired behaviour. The chat group also contains the Text balloons and the File Transfers balloons that are the entry point that the users can use to share content on Facebook. These balloons needed to be modify in order to present the share option when a user presses them and to present the Facebook indication after the share. They are also responsible to make the share request to the Social group, which in turn makes the necessary API calls to share the content. To keep a log of the shares, it was necessary to create a new database model that records all the shares made.

4.5 Remote Contacts, Presence and Contacts' information

The loading of remote contacts, presence and contacts' information were grouped in the same section because they focus on the same principle. They can be obtained in the same way through the same API call, **avoiding unnecessary** calls. To perform these functionalities the Facebook Graph API must be used, more specifically the FQL technology mentioned in the first chapter. These functionalities were developed mainly at the UI/UX level. So, there was no need to exchange the communication stack in order to achieve these features.

In short, the Remote Contacts feature allows a user to load and visualize his Facebook friends in the contact list of the RCS application. The user can also see the contacts' presence in the social network by a colored indicator: blue means online and grey means offline. If a user wants to see some details about their friend (birthday, hometown) or even if they wants to start a chat conversation, they selects the target contact and are able to perform these operations [Figure 21].

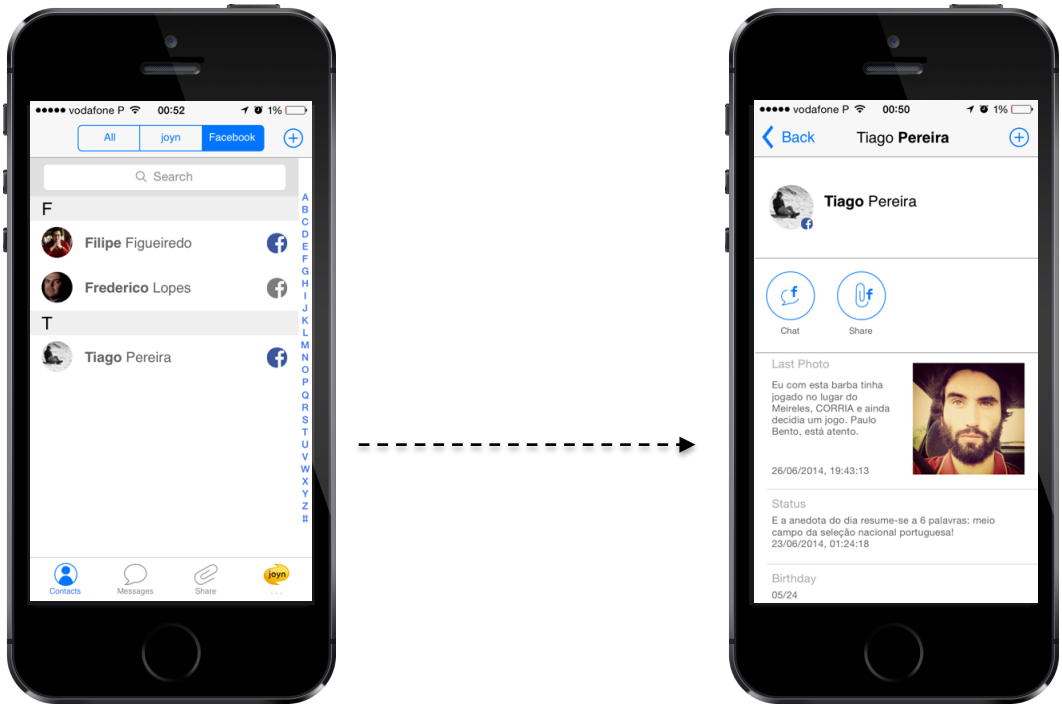

Figure 21. Remote Contacts, Presence and Contacts' information

The following diagram representing the final state of the architecture for the feature is illustrated in Figure 22. This functionality was developed mainly at the UI/UX level. There was no need to exchange the communication stack in order to achieve this feature. It is important to note that this diagram is high-level in order to respect the product's company privacy. For a detailed diagram that explains all the trainee's contribution, refer to the **Appendix D – Solution Architecture: Remote Contacts, Presence and Contacts' information**.

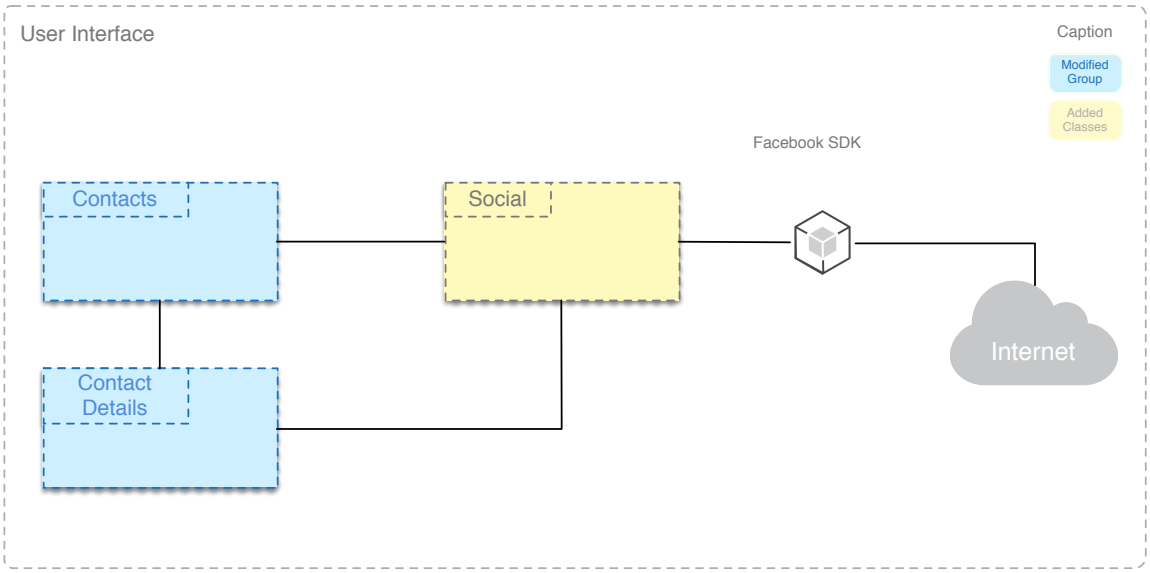

Figure 22. Remote contacts, presence and contacts' information final architecture

As the first step to achieve these features, there is a need to modify the Contacts group in order to support the Facebook contacts. The contacts are retrieved through two different providers. The task here was to create a new provider that connects to Facebook and it is responsible for retrieving the social contacts: **SocialContactsProvider**. This provider intends to get and manage all the contacts that are retrieved from the Facebook. This implements all the operations that the existing providers implement, however mechanism like update contacts are not developed because there is no need to edit contacts from a social network once that is a environment with dynamic data.

This provider makes direct use of the **Social** module as can be seen in the Figure 19. This manager conducts all the API calls, including the Remote Contacts, Presence and contacts' information as well.

An important aspect at this point is the app permissions to get the necessary information about the Facebook users. At the login time, a set of permissions must be defined in order to obtain the necessary information about the users in a future phase. So, the permissions passed at login regarding the contacts load are:

- "friends_photos": get the users' published photos.
- "friends_online_presence": get the user presence status on the social network.
- "friends_status": get the latest status that the user post on the Social Network.
- "friends_work_history": get the friends' work history indicating the company, role and time period.
- "friends_eductaion_history": get the friends' educational path indicating the institute, concentration and finishing year.
- "friends_birthday": get the users' birthday date.
- "basic_info": get the user name, born location and current location in the social network.

After the permissions are set, an API call using FQL can be made to retrieve the desired information. This is a multi-query is used to get the information and, once the operation is **asynchronous** it should be performed in a secondary thread (background) to maintain the

UI fluid. When the response arrives, an event is fired from the manager and the provider is notified. After the retrieval of the information, it is parsed into contacts and they are **cacheable** avoiding constant API calls. An important aspect is the contact presence. The presence status is a field that cannot be cached since it is quite susceptible to changes in a short time periods. Whenever an interaction with a contact is triggered (view contact's profile, start a conversation, etc.), the social network presence is checked and the contact is updated. So, it's guaranteed that the presence is always updated between contacts' interactions.

When a user selects a Facebook contact from the contact list, its details are displayed. This view shows the contact name in the social network, its presence and set of basic information. This basic information includes the current city where the he lives, his birthday, the last status that has been posted in the social network and his last photo post. Both are followed by publication date.

However, Facebook provides much more information than indicated above. This information can also be displayed in the contact details. If a user wishes to se more information about the contact, he should press the "more options" button and select the option "more info".

4.6 Chat

The chat functionality is achieved by recurring to another Facebook API, the Facebook Chat API. This API provides an interface to connect messaging clients to the Facebook Chat service. In short, a user can chat with their friends through the RCS application with all the allowed details.

By the time of the implementation, the RCS product already had conversation modules shown in the communication stack – COMLib – as explained in the previous sections. A contact can exchange messages through the instant messaging module (Simple IM) or through the circuit switched network (SMS). The goal of this implementation was to give a third way to exchange messages between clients (Facebook IM).

Figure 23. Facebook Chat

In order to follow the product's rules, it was necessary to develop and modify some logic in both communication and UI layer. It is the first feature that needs to be developed in both sides. To keep the document concise, the COMLib layer will be explained at first. Figure 24 illustrates the architecture of the chat module after the trainee contribution.

It is important to note that the diagram presented in this section is high-level in order to respect the product's company privacy. The COMLib is the company's most important component and it is extremely important to not give details about its internal structure in a document that will be public. So, for a detailed diagram and analysis that explain all the trainee's contribution, refer to the **Appendix D – Solution Architecture: Chat**.

The first thing to notice is the C++/Obj-C layer on the top of the C++ layer, which converts all the method calls, objects and callbacks from iOS application.

All the services provided by the COMLib could be used through an API and the chat service isn't an exception. This service has many implementations, as previously stated, and it uses a main controller to control all those implementations.

This controller is considered the core of the service that implements all the logic needed to the service's proper work. It has a set of available jobs with different behaviours. A job is worker that performs tasks in background in order to maintain the application fluid and responsive (e.g. sending a message, notification). To achieve the internship goals, some jobs needed to be modified and it was necessary to create new ones to perform some tasks related with the Facebook Chat.

The controller is also responsible for maintaining a reference to a set of handlers. These handlers have a specific technology and control the specific chat data source. It was necessary to create a new one to handle the Facebook Chat Service.

Finally, the controller has a set of callbacks that are used to communicate to the UI according the event nature (e.g. received message, notification).

The new handler has an instance of an XMPPAgent that is responsible for controlling all the logic about this protocol. This agent implements all the methods used by a default chat service like send message, send notifications, and others. It is also responsible to maintain a set of callbacks that are used to notify the respective handler that an event was fired like a message received.

This handler is also used to connect the necessary streams to the Facebook Chat server using a third-party lib named as XMPPFramework.

For more details about this lib, as well as details about the architecture in the Figure 24, refer to the **Appendix D – Solution Architecture: Chat**.

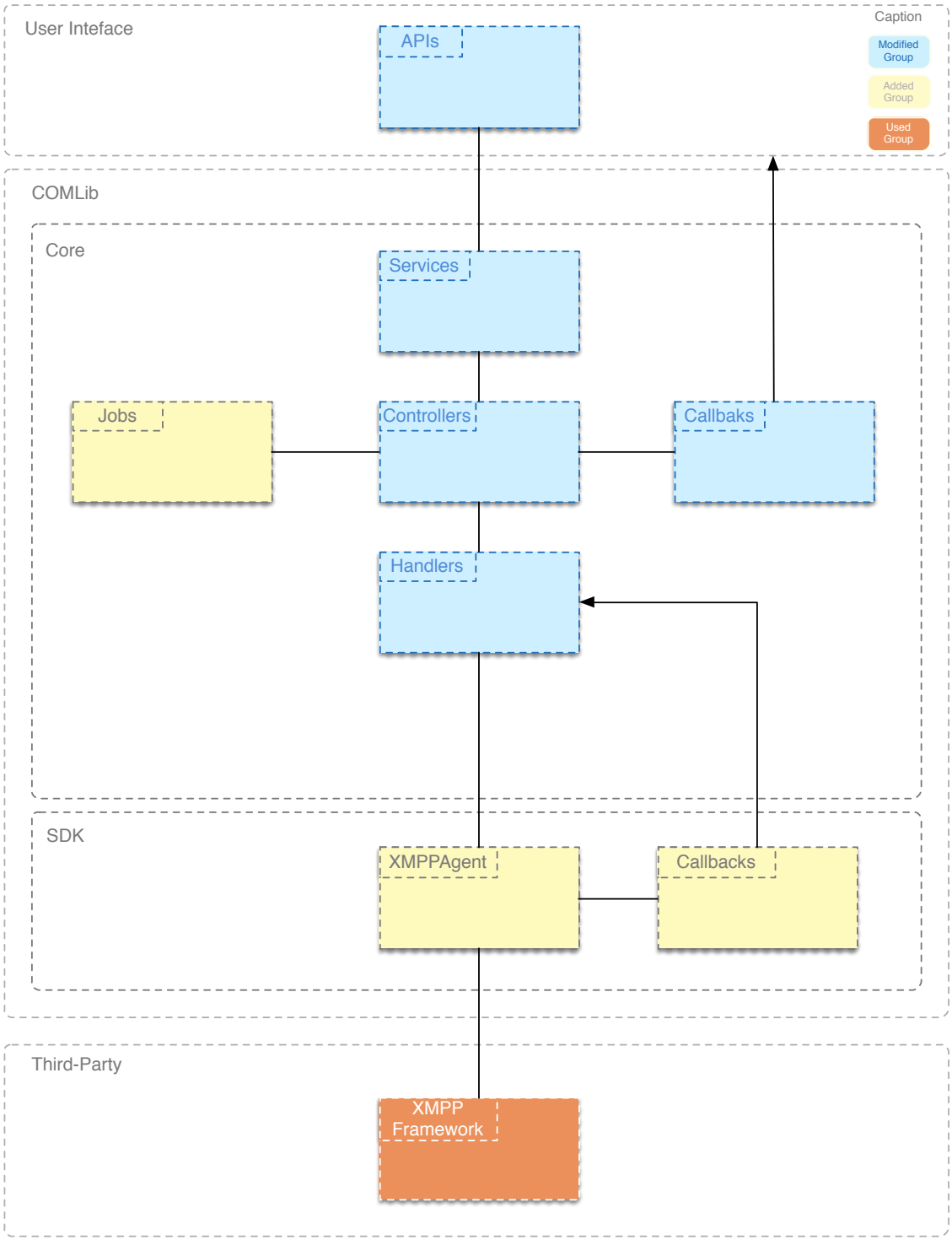

Figure 24. COMLib's chat module after the Facebook Chat logic

Regarding the UI layer, this feature is much simpler when comparing with the COMLib part, where basically all the heavy work is done. Most part of the work here consisted of modifying classes instead of creating new ones. Since the methodology of the team was followed in the communication layer, the UI was basically prepared to support the Facebook Chat. There was a need to make some adjustments to what already existed. In Figure 25. all the modifications and additions to the UI classes are illustrated:

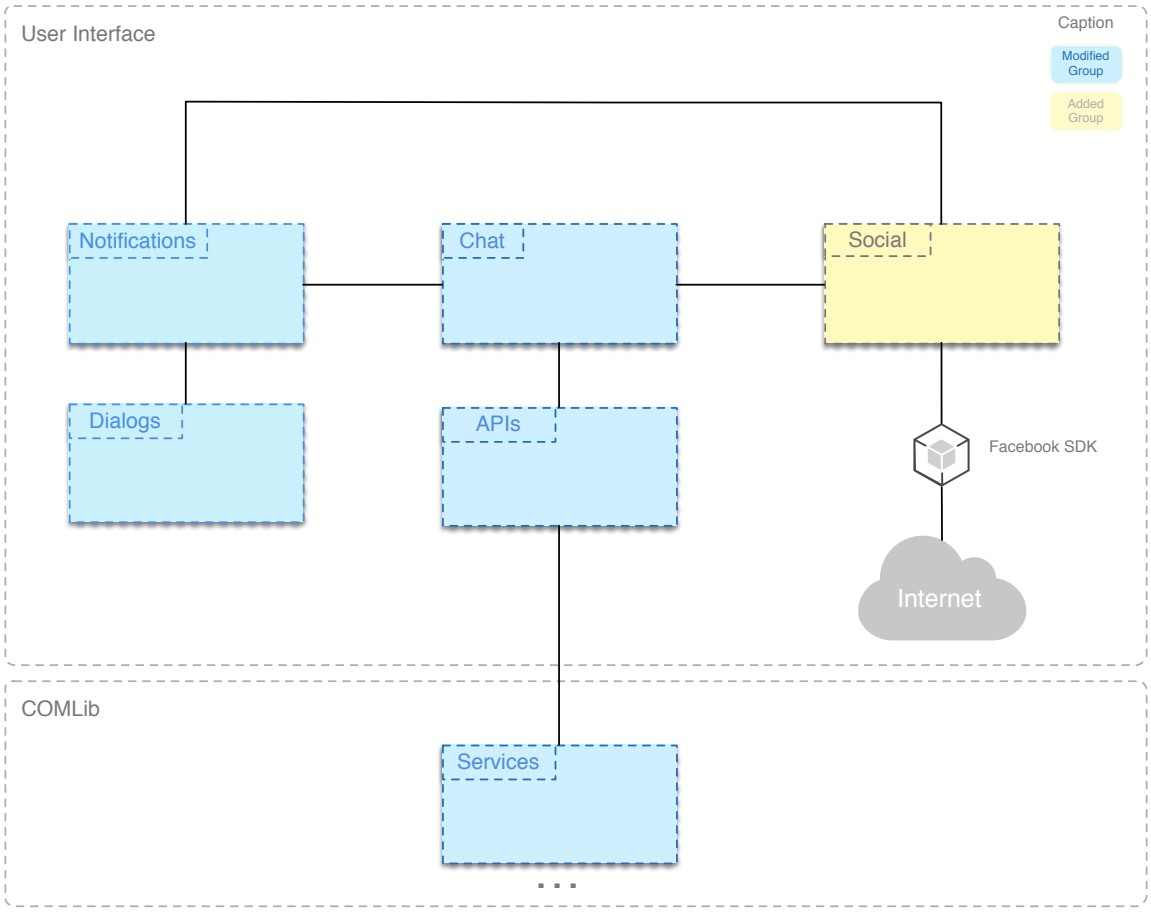

Figure 25. UI architecture supporting Facebook Chat

Through the previous diagram, it can be seen that this feature was mainly developed at **Chat**, **Social** and **Notifications** levels. An API is used in order to make the bridge between the UI and the COMLib. Some methods modifications were done in this API. The Chat group was modified in order to support the new data source (Facebook Chat Service) and it was modified in some views to show the messages with the proper indication – Facebook conversation. When a message is received and the user is in a different menu of the chat conversation, a local notification is fired showing some information about the message: user's photo, user's name and a preview of the message. It was necessary to modify the Notifications group in order to achieve the desired behaviour. The Social group contains all the necessary logic to enable the Facebook Chat service in the RCS product, including the authentication mechanism used by the TCP sockets in the communication layer.

4.7 File Transfer

The current section is responsible for defining the architecture for the File Transfer feature using Dropbox. This feature relies on the previously defined authentication architecture, once they will only be available if the user is logged in to its Dropbox account.

This feature can be divided into two components: save and share (send a file) from Dropbox. The save feature was not initially planned but it was implemented because it enriches the internship and the file transfer functionality. Both these features have many similarities in terms of classes, and they rely on Dropbox Core API. Their architecture is thus presented together in order to not overflow this chapter and to provide their shared context. However, each of the flows are explained separately in order to fully understand each of the features.

Save to Dropbox

This feature allows to save any received file (e.g. image, video, audio, document) from a single or group chat session to the user's Dropbox account.

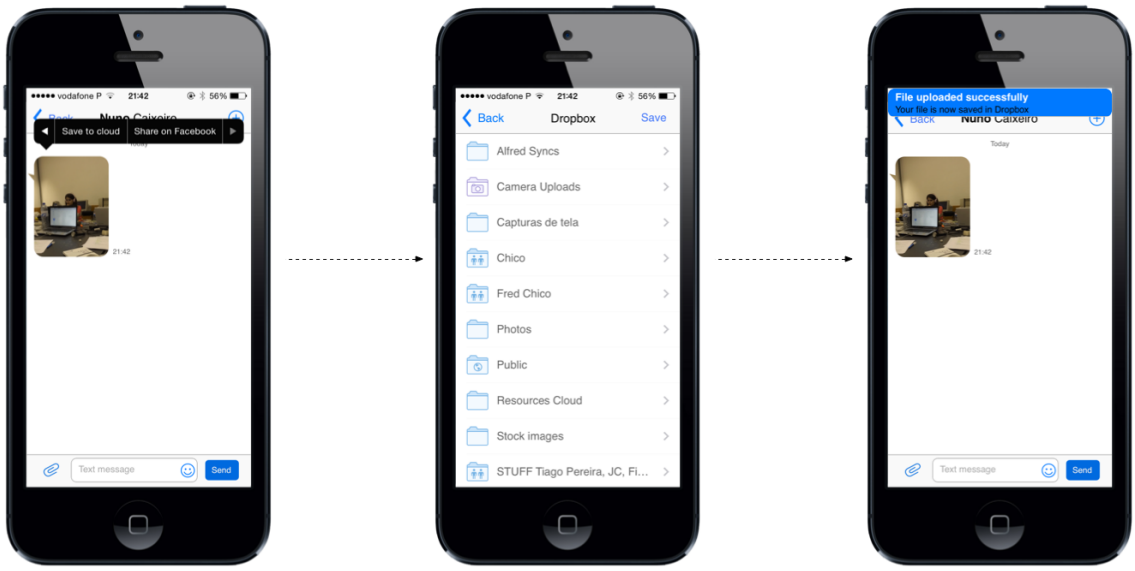

Figure 26. Save to Dropbox flow

Figure 26 briefly shows how this feature works. After receiving a file transfer, the user long presses on the respective balloon in order to see the possible interactions with that file. One of them is "Save to cloud", which will present a new view with all the clouds the user is logged in within the application. After selecting the desired cloud service, the user may browse through all the folder hierarchy of their cloud account by interacting with the presented folder explorer. After navigating to the intended folder, the user needs to press the "Save" button in order to start the uploading process. Pressing that button will display back the previous chat window to the user, where they can see a progress bar below the balloon they selected. After the upload is completed successfully, a local notification is triggered in order to inform the user about the conclusion of the process. If any error occurs during that process, a local notification will warn the user about the problem that prevented the file from being saved to his cloud account.

Share from Dropbox

A user can share a file from its Dropbox account via two entry points within the application: a chat window or the share menu within the application's main screen. Figure 27 displays both these entry points.

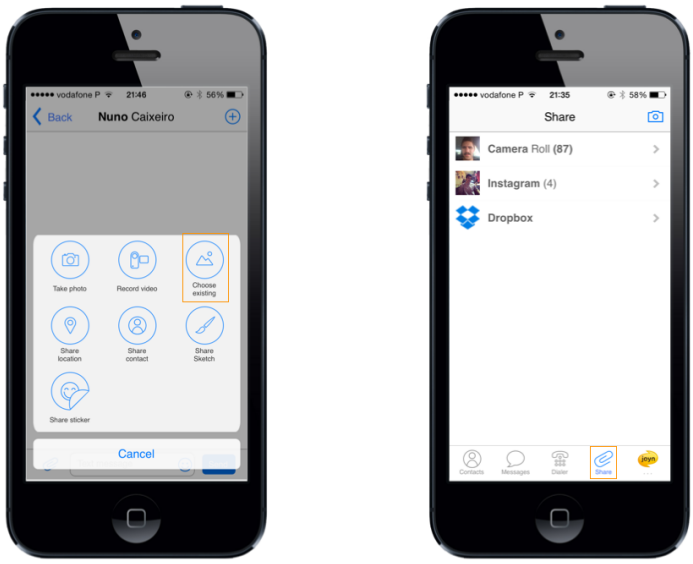

Figure 27. Send File from Dropbox entry points

Both the available entry points will start the feature's flow illustrated in Figure 28. A list with all the media sources pops up, which includes the clouds the user is logged in within the application. Selecting the Dropbox entry will once again show them a browser where they can navigate through their account's hierarchy. However, unlike the previous feature, the browser displays all the available content, whether they are files or folders. The user may then select files and, after the selection is terminated, press the "Send" button in order to send those files. Pressing that button will display the respective chat window where the user can see the outgoing balloons with the intended files.

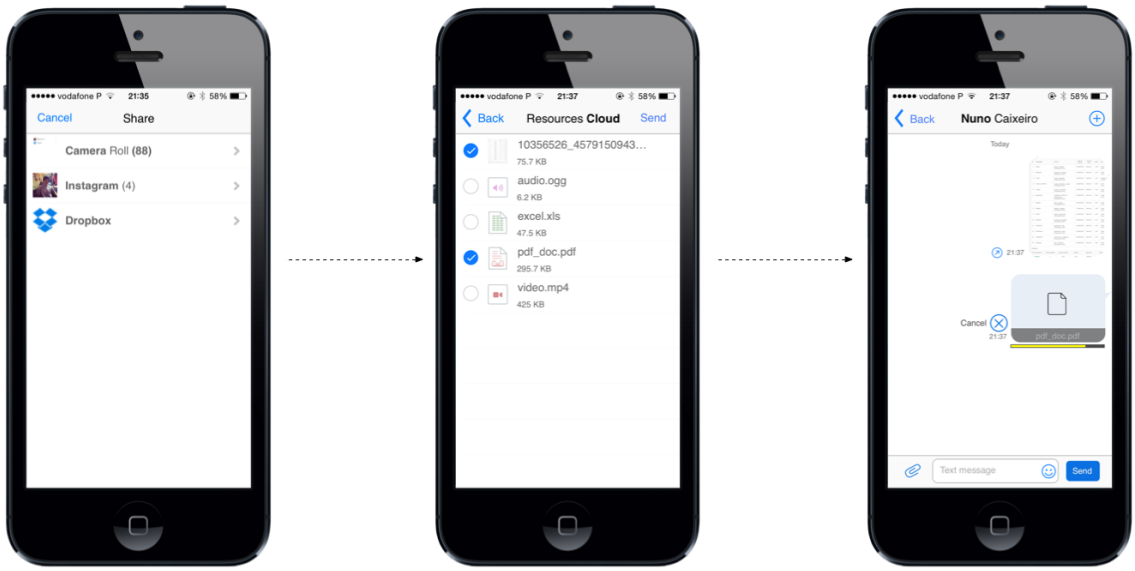

Figure 28. Send File from Dropbox flow

The diagram representing the final state of the architecture for the feature is illustrated in Figure 29. This functionality was developed mainly at the UI/UX level. There was no need to exchange the communication stack in order to achieve this feature. It is important to note that this diagram is high-level in order to respect the product's company privacy. For a detailed diagram that explains all the trainee's contribution, refer to the **Appendix D – Solution Architecture: File Transfer**.

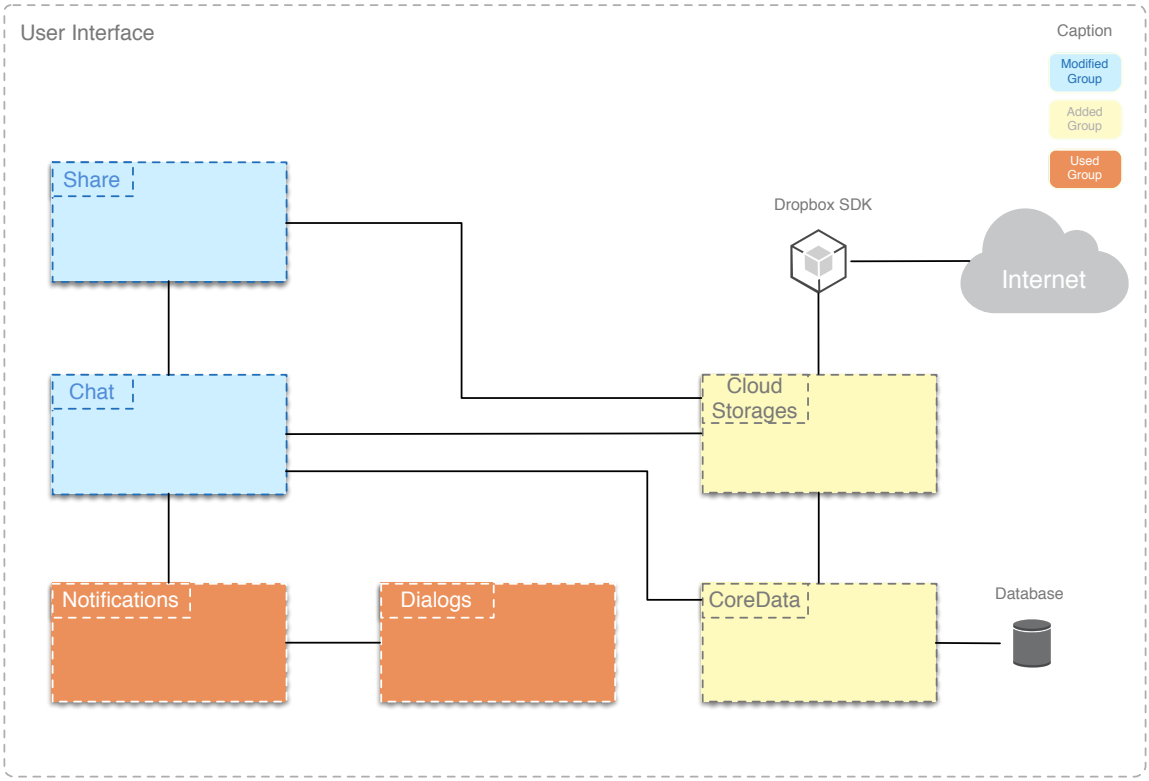

Figure 29. File Transfer via Dropbox architecture

This feature was mainly developed at the **Chat**, **Cloud Storages** and **Share** levels. Once the files are exchanged via chat, there was a need to modify this group in order to support a new File Transfer mechanism via Cloud Storages. The Share group provides all the proper views to pick the desired files and methods that are used by the Chat to send those files. The Cloud Storages group is responsible for all the logic that uses the Dropbox SDK in order to complete the actions. This File Transfer mechanism works by generating a public link that will be sent to another contact, i.e., when a user selects one or more files, the corresponding public links are generated through the Dropbox SDK and they are sent to another contact. Once they reach the other party, they are viewed as files and the Chat group is responsible for setting up the proper views to show these links that are, in reality, files. In order to **optimize** the UI that is presented to the user, the **CoreData** group is used. This usage consists of saving the Dropbox entries information (e.g. files size, files thumbnails, etc.) in order not to make unnecessary API calls. When a file is saved on Dropbox the **Notifications** group is used in order to generate a local notification. Once the process is **asynchronous** the user should know when it terminates. The Notifications group used custom **Dialogs** to represent the local notifications with the proper content.

5. Implementation challenges and Evaluation

There are always challenges to overcome when developing a software product. This chapter explains the implementation process and the evaluation of the solution. At first, the most important implementation decisions as well as the problems solved will be presented. To complete the chapter all the evaluation mechanisms performed regarding each feature in order to evaluate the final solution of the product are presented as well.

5.1 Challenges and Problem Solving

Sometimes the processes planned in the beginning of a project are not the best way of doing things. There are often problems in the software development that are not thought at an early stage. In the internship's scope, most of the potential problems were predicted far ahead and the architecture for the concerned features was designed with those problems in mind. Still, some of them were impossible to predict and there was a need to create strategies to address them. These strategies have not required major changes in regards to the architecture but they involved several challenges to the trainee.

This section will briefly explain the implementation of each feature, and it will focus on the main challenges they brought to the project, along with the respective alternatives/solutions found.

5.1.1 Share on Facebook

The Facebook Share feature was the first to be implemented, representing the major challenge at the UI level in this internship. This magnitude is due to the fact that it was the first contact with the WIT's RCS product and its dimension goes beyond any project in which the trainee had participated. However, with some initial effort this challenge was overcome successfully and the rest of the development went well until the end, such that its result id ready to be included in the product.

This feature has two implementations until this actual state. Firstly, the Facebook SDK was used in order to achieve it. This SDK has a lot of information in its documentation, which made its development easier. However, the idea of using the native iOS mechanisms to make this feature emerged. The reason for that is because they make the feature much more familiar for the user than a random share mechanism inside the application.

The native mechanisms provide a friendly interface for the developers, working as a wrapper which role is to call low-level methods that allow the application to make a share on the Social Network. The native framework used provides interfaces that support the sharing of text and photos but the ability of sharing videos was also required to complete this feature. To reach that point, it was necessary to understand the low-level methods and a custom interface was made in order to have the same behaviour provided by the native resources provided. It took an extra effort to understand the low-level framework methods, however, the obtained results were quite satisfactory for the company and the company's clients.

5.1.2 Remote Contacts, Presence and Contacts' information

During the development of this feature, two great challenges emerged. The challenges were to understand how the information should be requested in order to spare the user's data plan and how the RCS product should handle so many asynchronous external API requests.

In first place, it was necessary to understand what information could be retrieved and what information the application really needed to satisfy the users. After some internal discussion, it was possible to conclude that it could perform an amount of useful things with the Facebook users' information. Functionalities like Contact Enrichment, Birthday alerts, internal Timelines could be performed using this data source. So, the most appropriate information to achieve those features was gathered. The main concern here is the fact that, if a user has in average 230 friends [66] and if the friends' information has 2,5 KB per friend, it means that for each API request to retrieve the entire friends' list and their information a user will use nearly 0,6 MB. This value is extremely high for cheaper data plans that the carriers usually provide. So, to smooth this problem the trainee found two solutions:

- The API calls needed to be well structured in order to request only the needed information. The headers that are, sometimes, greater than the actual responses have been removed in order to save the data plan.
- Enable cache in types of information that are not very susceptible to changes, while defining proper expire dates.

A key aspect in the API calls is the fact that they are all made by a secure connection. This secure parameter is achieved by adding a flag to the API calls in order to activate the Secure Sockets Layer (SSL).

Another key aspect was to realize how an application that is not prepared for the usage of an external API would deal with many asynchronous requests. Figure 30 illustrates how the behaviour should be implemented.

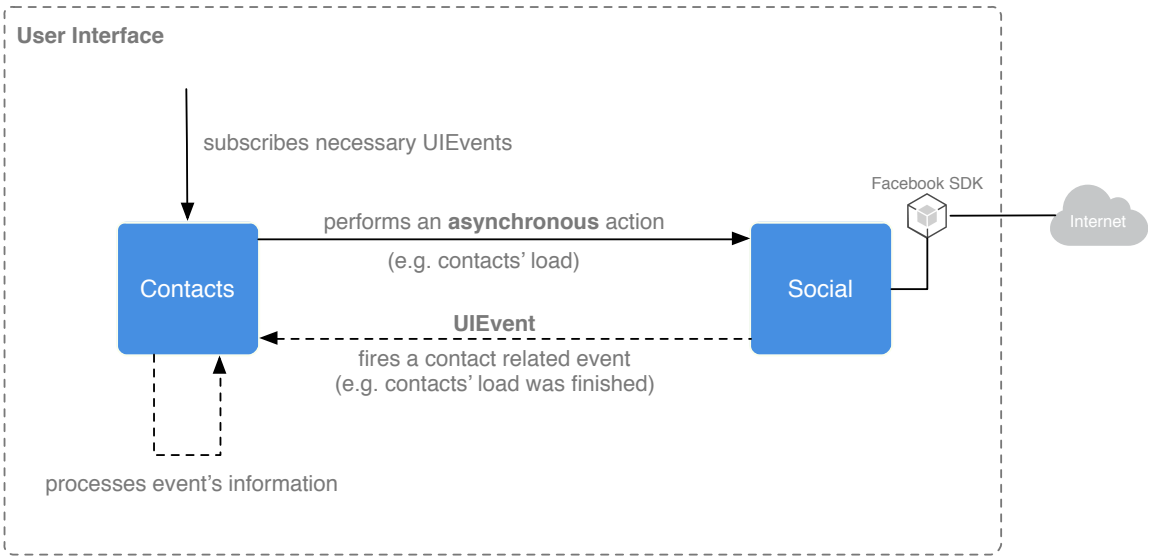

Figure 30. Contacts implementation details: Asynchronous call and response

In order to make the perfect management of the asynchronous call, the Grand Central Dispatch was used, as said in the previous chapter. A pool of reserved queues was created, such that it would only handle the Facebook contacts' logic. It was also necessary to develop mechanisms that allow the queues to communicate between them. To perform this communication, an iOS resource named as *UIEvent* was used. This event allows to notify an application part (that subscribed the event in question) that something was finished. When these parts receive the notification event, they can execute their specific job/behaviour.

In conclusion, all the API calls were meticulously treated always thinking about the end user, including saving mobile data and safety. All the processes related to Facebook contacts are asynchronous using a thread-safe mechanism. When a communication is needed between those threads, an event is fired and all the subscribers can execute their job.

5.1.3 Chat

The chat was undoubtedly the most challenging feature during this internship. This feature required development work in both the UI level and the communication stack level. The UI hides all the manpower and logic beneath it once it followed the default guidelines of the product in order to be familiar to the end users.

The first challenge found was the fact that both the UI and the COMLib needed the Facebook SDK to make their specific job. The UI needs this SDK to perform features such as the load of remote contacts. The communication layer needs this SDK to connect the TCP sockets that will be used to exchange messages between Facebook users. A solution was thought in order to solve this problem as illustrated in Figure 31.

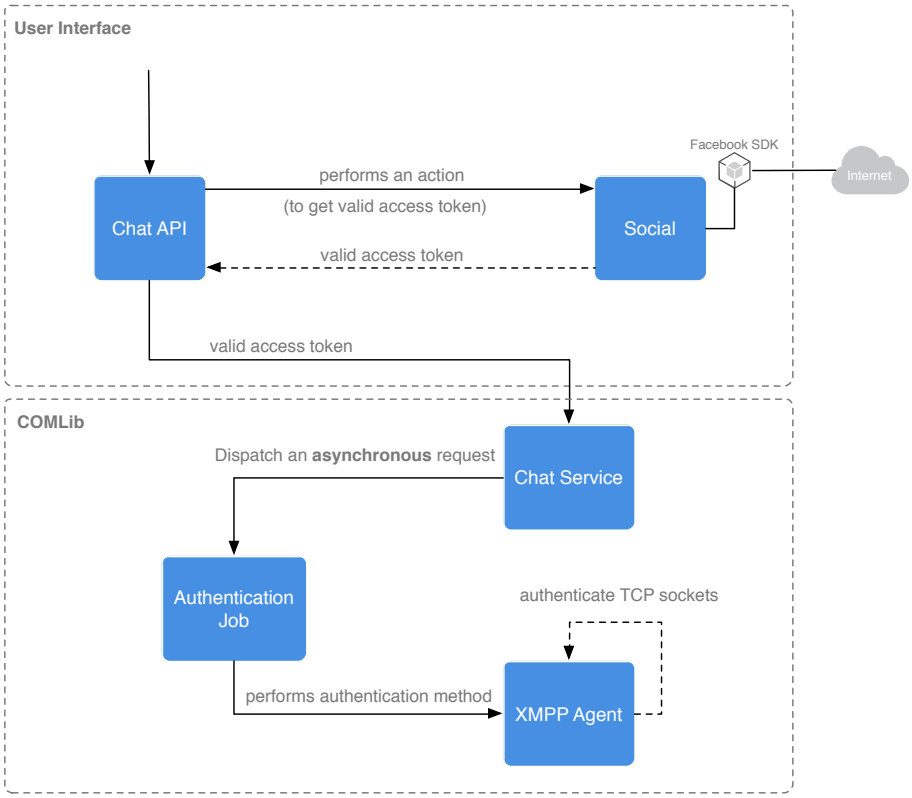

Figure 31. Chat implementation details: SDK on both UI and communication layer

This problem was a headache at the beginning because the code was not structured and prepared for such. However, a solution that solved this problem was found: this solution consisted of the API modification that is used by the UI to access the chat service provided by the COMLib. A new method that can be called by the UI was created and through this method, the UI passes an access token that are used to authenticate the TCP sockets.

At glance, it was a good solution but it led to another problem. Once all the processes in the application are asynchronous, sometimes the API call to pass the access token was made before the chat service was initialized, leading to segmentation errors. To solve this, it was necessary to ensure that the call could only be performed after the service is initialized. For this purpose, a specific job was created which role was to wait for the service initialization and after that to perform the call. This job runs in background ensuring a smooth application and solving the mentioned problem.

The second problem consisted of the contacts identifier, best known as URI. This URI has various formats and it consist of the MSISDN that is a number unique identifying number for a subscription in a GSM mobile network. The COMLib gives support for all those formats including internationalization. However it was not prepared to support a new type of URI, that consists in an unique ID used to identify a Facebook user. The URI logic can be very tricky and very specific, thus leading to a complex resolution of this problem. To solve this, the trainee needed to create a new scheme that helps the communication layer understand which service is responsible for handling that specific URI. Once differentiated by the scheme, the URI could be forward to the specific handler (in this case Facebook Handler) that has all the needed logic to interpret and understand that URI.

At last but not least, another issue was addressed to the Chat functionality. The messages sent from the UI were reaching the communication layer but they did not reach the Facebook servers and consequently they did not reach the intended user. The reason for that was the fact that all the message states were marked as *Dirty*. The communication stack uses the term *Dirty* when a message has an unknown source or destination which means that the scheme that is presented in the message URI is not recognized. In fact, when the specific Facebook scheme was created, the jobs that were responsible for sending the messages should know about its creation. Since they had not knowledge of their existence, the jobs attributed the message status as *Dirty* and the message was discarded. When the scheme was registered in the jobs, this problem has stopped and the messages started to have the expected communication flow. Figure 32 illustrates all the process that was performed in order to solve this problem.

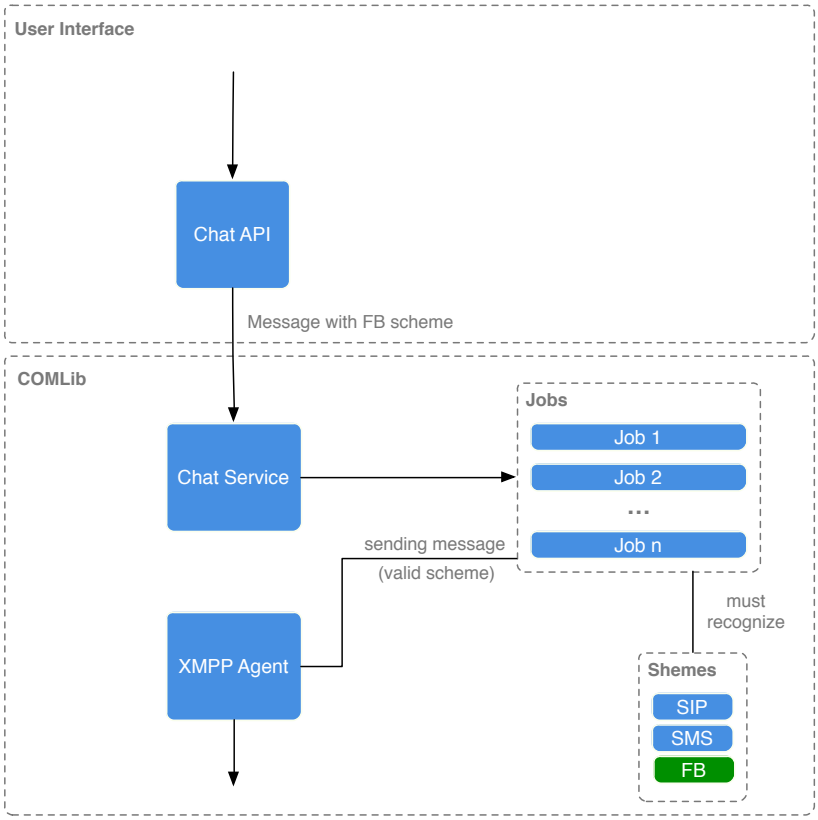

Figure 32. Chat implementation details: scheme problem solution

In conclusion, this was the feature that raised more problems due to its dimension. It was necessary to develop logic in two different environments, including the communication layer, a very complex environment whose learning curve was quite high.

5.1.2 File Transfer

At a first glance, it was thought that this feature would not be possible to develop. The Facebook Chat API only allows the exchange of non-HTML content (text) through it and does not support the usage of file transfer extensions provided by the XMPP protocol.

However, despite the limitations imposed by the social network, the trainee came with a proposal that was discussed and subsequently accepted. Another internship that was taking place at WIT Software consisted of the RCS product integration with Cloud Storages, more precisely the Dropbox cloud. One of the Dropbox main features is the sharing of a public link that enables any user to download a file using the same link. It is true that Facebook does not allow the transmission of html content. Nevertheless, a link is text-based and can be sent through the Chat API. Thus, the user where the link is sent can download the file simulating a file transfer mechanism. In the Facebook platform the user observes the file as a link but in the RCS product the trainee has the power to display a proper view to present the link. In order to smooth the UI, the target file is downloaded via its public link and it is shown in a chat balloon as a file and not as a link similar to the File Transfer technology already presented in this application.

This is not an elegant way to present the feature but it is a smart alternative to the restrictions imposed by the social network.

The most challenging part of this feature was the creation of a generic file explorer that allowed browsing through the Dropbox folder hierarchy, just like in the Dropbox application.

As mentioned earlier, this feature was divided into two diferent featues: Share (File Transfer) and Save from/to Dropbox. Both these features rely on the file explorer created that allows a user to browse the cloud's folder hierarchy and select the intended resource(s). The only difference is that it displays only folders for the Save to Dropbox feature because the user must only select the destination folder, while for the Share to Dropbox feature it displays both folders and files in order for the user to select files inside any folder.

In order for the file explorer to show the files and folders from Dropbox, an HTTP request seeking for the metadata from a certain directory. The response contains all the information related to each of its folders and files. That information is then processed into the necessary data models and then presented to the user. Two different flows may occur:

- **Share from Dropbox:** the user selects the files to send. When he is done, the public links of each files are requested and, upon their arrival, they are sent using the COMLib's File Transfer API.
- **Save to Dropbox:** the user goes to the folder he wants to save the file. When he is done, the folder's name is retrieve and a method from Dropbox Core SDK is called in order to upload the intended file to the cloud. Besides the destination path, that method also receives the path where the file is stored locally.

As previously mentioned, the file explorer shows all the files inside a certain folder when the *Share from Dropbox* mode is active. All files have an associated thumbnail, which is useful in order to provide a quick look about it. This way, it is a must to download these thumbnails in order to enhance the UX provided to end users. However, the download of these small images have a drawback: once the mobile data plan is used it is consumed when performing these actions. This, it is necessary to optimize this behaviour in order to spare the users' mobile data. To achieve that, a mechanism was implementer that consisted in the Core Data usage – persisting these data on iOS device, increases the applications' responsiveness. To sum up, for each listing file is verified whether it has an available thumbnail on Dropbox cloud. If so, the thumbnail is downloaded and persisted in a Core Data entity. Once this process is finished, each time the user returns to that directory and the thumbnail is displayer, it is loaded from the database instead of being downloaded again. This processes leads to major advantages: it reduces the consuming of the mobile data and it increases the file explorer's responsiveness, once the thumbnails are already available to be displayed.

Regarding network-loss, the Dropbox SDK returns an error any time there is loss of connectivity during one activity. This way, it was necessary to listen for those notifications in order to present the necessary informative notifications and dialogs to the user.

5.2 Evaluation

This section reports all the tests performed to the final solution, including all the planned features. They could be divided into two categories: functional and non-functional tests. The first section covers the functional tests performed and all of them grouped by the feature they relate to. Then, follows a set of tests to validate the new feature's performance against network losses/unavailability. The last subsection provides information about the performed usability tests, in order to ascertain the users acceptance of the implemented features.

It is important to mention that battery tests were thought in order to complement the solution's quality evaluation. However, they were impossible to perform bacause the company doesn't have needed resources for that purpose. Apple does not provide any application to control the battery consumption and it has several restrictions for third-party applications that perform this action (only with *jailbreak* that consists in an iPhone hack). The alternative to perform these tests passes by welding an external device to the iPhone battery (it needs to be open) that controls the consumptions. It is a risky alternative and for compliance it was not executed.

Furthermore, an article [67] was found with the research made to evaluate the alternatives informing that the next iOS release (iOS 8) will bring native resources to measure the battery consumptions of each installed app finishing all the existing limitations.

For a detailed analysis including all the test cases grouped by features and displayed in proper tables, refer to the **Appendix E – Evaluation.**

5.2.1 Functional testing

Functional testing is a quality assurance process that allows to find unexpected behaviours in the application, or even potential crashes due to not predicted situations in the implementation phase. In this type of testing, the use cases earlier defined are the core of the operations once they are tested by giving an input and analysing the outcome.

Regarding the current project, when a user interacts with the application it must reproduce the desired result meaning the feature's proper behaviour, enhancing the user experience. Once most of the features that compose this internship are visual (they present visual information to the user upon the intern logic) it is very important to guarantee that all the developed use cases reproduce the predicted outcome.

It was created and executed a total of 76 functional tests in order to ensure the features quality that compose the RCS application:

- **6** – Authentication
- **14** – Facebook Share
- **24** – Remote Contacts, Presence and Contacts' information
- **18** – Chat
- **9** – File Transfer

The testing process has been continuous, meaning that it went along all the implementation phase. This way, it is natural that the tests were executed multiple times until all the bugs were corrected. The following table shows the functional testing results of the first and last runs for all the implemented features:

	First run		Final run	
Feature	Failed	Passed	Failed	Passed
Authentication	0	6	0	6
Facebook Share	0	14	0	14
Remote Contacts, Presence and Contacts' information	3	21	0	24
Chat	3	15	0	18
File Transfer	1	8	0	9

Table 6. Function testing results per feature for both first and last runs

This testing process allowed to find some bugs that could be a problem in a later stage. Among the 76 tests performed, 7 failed in the first analysis

Across the table it is easy to understand the most problematic features during the development stage. The features associated with Facebook contacts had some issues related to how the information was collected and displayed to the users. Sometimes, the events used by threads to communication were not sent and the information that was presented to the user was invalid. The Chat functionality had some bugs at the notifications level. The B-party name and photo were incorrect when a notification was displayed. This problem was related with the contact URI once the application did not recognize the specific URI in order to retrieve the contacts' information to display as local notification. The file transfer had a bug in the first run that consisted in not displaying a warning when a User selected more than six files to send.

A detailed analysis about the functional tests that were created and executed for each feature are described in **Appendix E – Evaluation: Functional Testing**.

5.2.2 Networking

All the implemented features involve an active Internet connection to perform the proper work. Thus, a small set of networking test cases were created for each feature, allowing to understand whether they perform correctly together with connectivity problems or with the total connection loss. Disabling the feature or delaying the execution of its operations are some of mechanisms used in this kind of situations. Table 7 shows all the 15 networking results of the first and last runs for all the implemented features.

	First run		Final run	
Feature	Failed	Passed	Failed	Passed
Authentication	0	4	0	4

Facebook Share	0	3	0	3
Remote Contacts, Presence and Contacts' information	1	3	0	4
Chat	0	2	0	2
File Transfer	0	2	0	2

Table 7. Networking tests results per feature for both first and last runs

The only feature with a network-related problem was the **Presence** feature. The issue was related to the test case N.2.3.4, in **Appendix E – Evaluation: Networking**, which says that a User must see the last contact's presence if there is no 3G/Wi-Fi connection. In this case, the presence is shown through a Facebook icon and the bug consisted of the vanishing of that icon when there is no 3G/Wi-Fi connection. This bug was corrected and the feature now works properly.

A detailed analysis about the functional tests that were created and executed for each feature is described in **Appendix E – Evaluation: Functional Testing**, so it is highly recommended to read this document.

5.2.3 Usability

This is the last subsection of this chapter and it provides all the information retrieved for the performed tests in order to understand the application's usability. A total of **20** people were part of this study, all of them from WIT Software. Once the company does not officially present the features they must remain as private to respect the company's privacy policy.

The test case consists of the following steps:

- For the five main implemented features, a Use Case is selected;
- Then, a user is told what he must do to perform the Use Case and all the actions are recorded (an action is any kind of interaction with the application);
- After that, the user's obtained number is used to be compared with the minimum number of actions to perform the Use Case;

This test allows to understand how easy or difficult it is to interact with the newly implemented features.

To provide a representative sample of the users, they were chosen randomly. As expected, most of them had already been in touch with the RCS application – 14 – whiled others had not. Furthermore, not all the selected users had an iPhone or make use of the iOS on their quotidian. Regarding their devices' operation system, the users sample is as follow:

- 10 iOS users
- 7 Android users
- 2 Windows Phone users
- 1 BlackBerry users

The following subsections of this section present and analyse the obtained results.

Facebook Share

The Use Case chosen to present this feature was a simple photo sharing with a text description via group chat. The use case that the users needed to perform involved selecting a Group Chat with received photo(s) in order to share it on Facebook. The starting point for all the users was the chat inbox.

The minimum steps to reproduce this use case were:

1. Choose the specific group chat
2. Perform a long press on a received photo
3. Choose the *Share on Facebook* option
4. Add the descriptive text
5. Press the *Share* button

Regarding the results of this test, Figure 33 illustrates the number of steps needed per participant to complete the task.

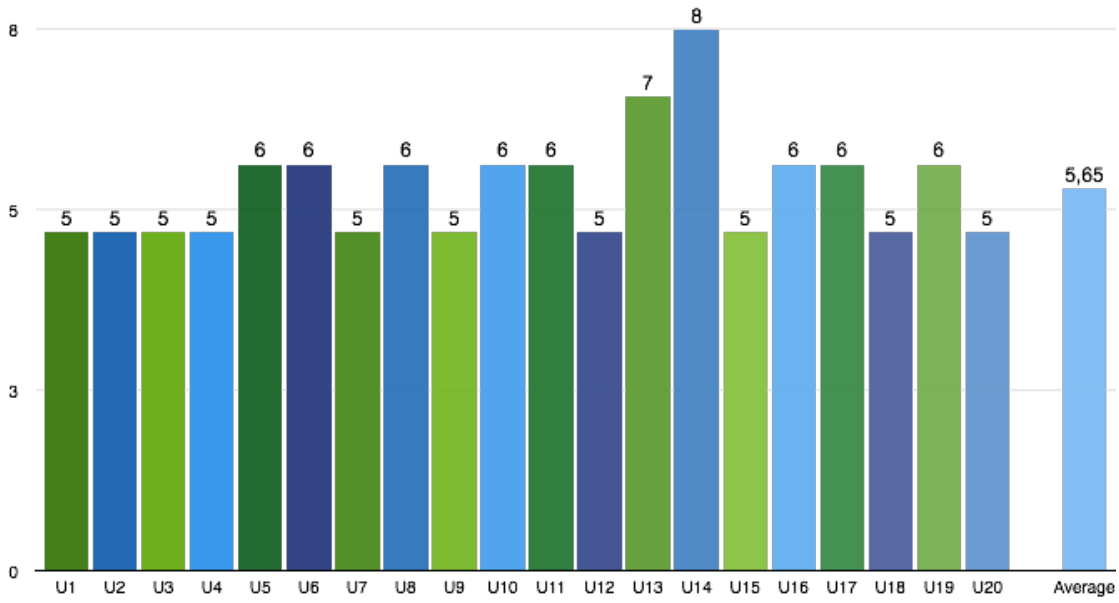

Figure 33. Facebook Share: actions done by the trial set to execute the test case

The result of this Use case test is quite satisfactory, with almost half of the users completing the task in the minimum number of steps. Only one took 3 more actions and another took 2 to perform the 5-action task. The reason for that was the fact that this user selected a different option than *Share on Facebook* when long-pressed the cell, making him redoing all the action sequence to select the proper option. The average number of actions was 5.65 versus the minimum number of 5, which is a good usability indicator.

Remote Contacts, Presence and Contacts' information

Regarding the presented features, the chosen use case can be a little tricky and consists of five actions. In short, the users had to visualize the working career of a contact starting from the chat inbox. During the design of this feature, the UI and UX were changed slightly towards the enhancement of the final outcome.

The minimum steps to reproduce this use case were:

1. Choose a chat conversation
2. Select the *Options* menu in the chat entry
3. Chose the *Go to contact card* option
4. Select the *Options* menu in the contact details
5. Chose the *More Info* option

Regarding the results of this test, Figure 34 illustrates the number of steps needed per participant to complete the task.

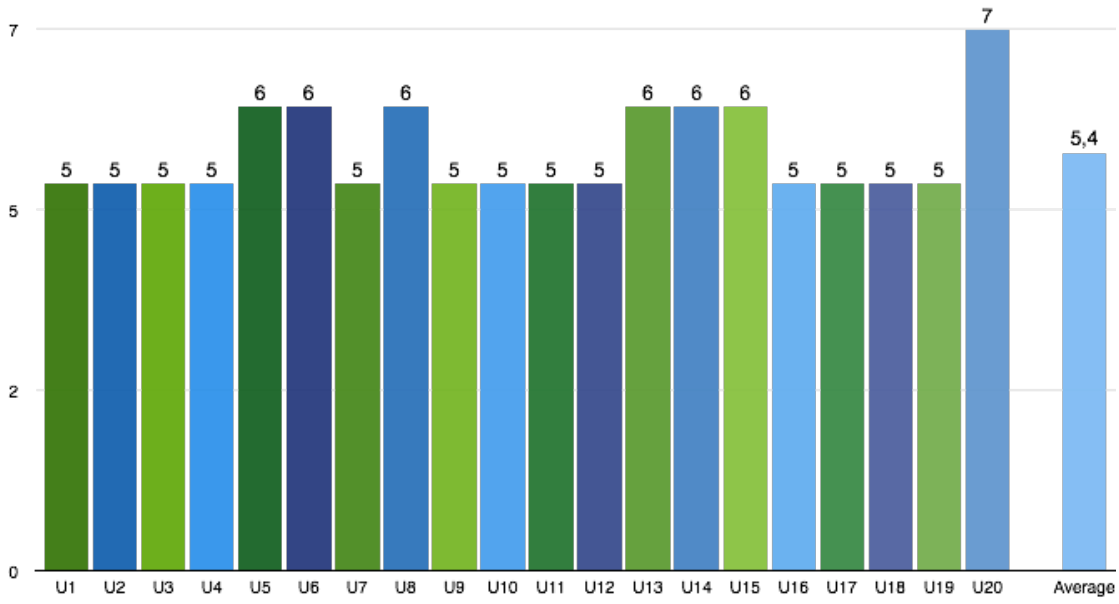

Figure 34. Contacts' information: actions done by the trial set to execute the test case

The result of this use case test is quite satisfactory, with more than a half of the subjects completing the required task in the minimum number of actions. Only a single person took more than 5 actions (user 20) to perform the task. The average number of actions was 5.4 versus the minimum number of 5, which means that all the UI and UX concerns during the development phase resulted in a good usability provided to the users.

Chat

For the Facebook Chat functionality, it was prepared a use case that needed 7 actions to be performed. The users had to create a chat with a Facebook user and sent him a message that contains at the end the soccer ball emoji.

The minimum steps to reproduce this use case were:

1. Select a user from the Facebook contact list
2. Select the *Chat* menu in the contact details
3. The user should select the composer
4. Write a message
5. Press the *Emojis* button in the right of the composer
6. Open the fifth view of the emojis table
7. Select the soccer ball

Regarding the results of this test, Figure 35 illustrates the number of steps needed per participant to complete the task.

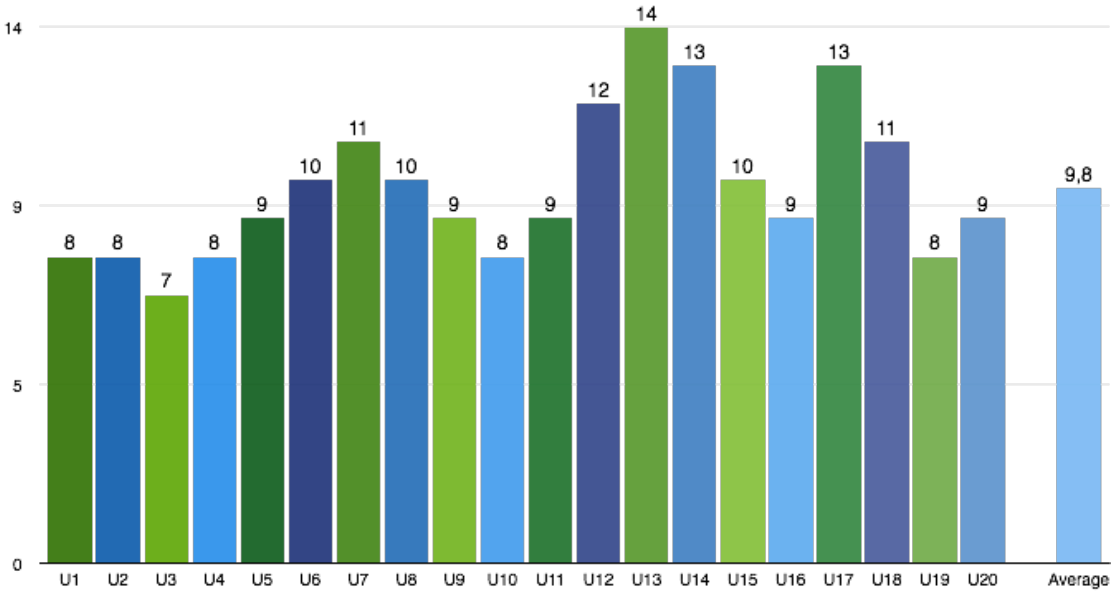

Figure 35. Facebook Chat: actions done by the trial set to execute the test case

The result was not as satisfactory as in the previous features. This use case requires 7 actions to be executed and only one user has successfully completed the task. The worst performance was 14 steps, and two other followed with 13 steps to perform the use case. These extra steps were taken in the emojis table where the users wasted a lot of actions to find the specific emoji. The average number of actions was 9.8 versus the minimum number of 7, which means this feature could be improved in several aspects, more specifically in how it organizes the emojis and makes them available to users.

File Transfer

The file transfer functionality was implemented using the Dropbox cloud. At implementation time, an extra feature was developed allowing a user to save a received file directly on Dropbox. However, to represent this group of features, the share (send file) functionality was chosen because it was the one that led to this integration. The use case which the users had to perform involved selecting a file from the configured account on the device in order to share it with a specific contact. The starting point for all the users was the contact list.

The minimum steps to reproduce this use case were:

1. Choose the specified contact
2. Press the "Share" button
3. Select the "Choose existing" option;
4. Select "Dropbox"
5. Select the desired file by pressing on it
6. Press the "Send" button.

Regarding the results of this test, Figure 35 illustrates the number of steps needed per participant to complete the task.

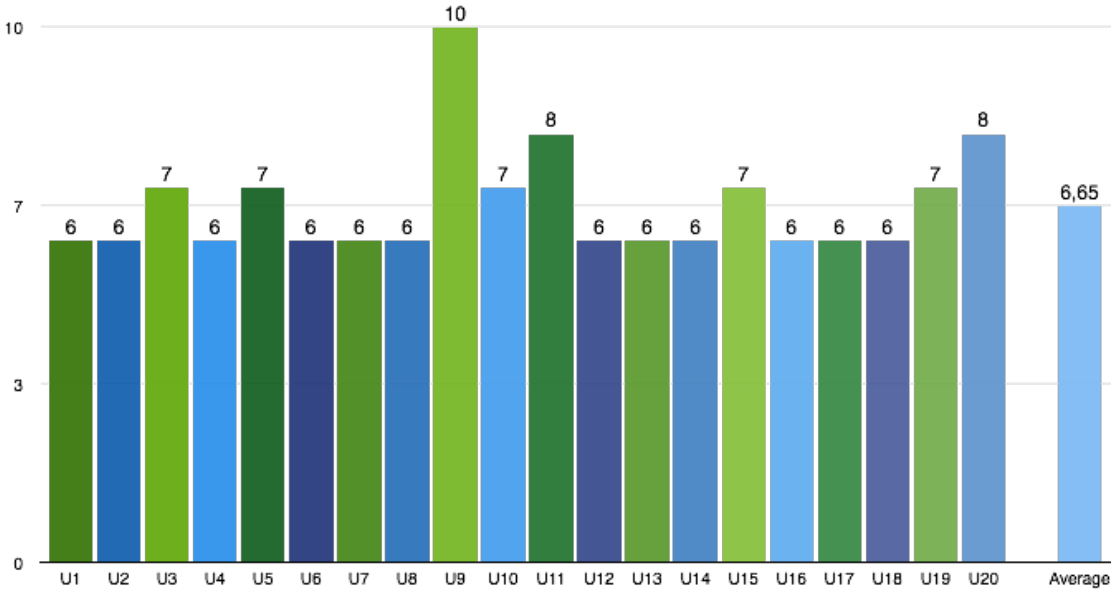

Figure 36. File Transfer: actions done by the trial set to execute the test case

The result of this test was highly satisfactory, with most of users completing the required task in the minimum number of actions. Only a group of five persons took an extra step than those required. The average number of actions was 6,65 versus the minimum number of 6, which means a good usability indicator. However, the following topics explained what happened in some cases:

- The user who did ten actions selected a file that was bigger than 5MB (error size), and then selected a file that had size between 2MB and 5MB (warning size). That led to warnings that need the user confirmation, which meant more actions.

- Five (people who took 7 actions) selected a file with warning size, while two other selected a file with error size (people who took 8 actions).

Conclusion

Regarding the usability of the performed tests, the final results were quite satisfactory. Except the chat use case, all the tasks were completed with a good average number of actions if we compare it with the minimum number required. This way, it revealed a good usability.

On the other hand, the chat feature tests have demonstrated some issues that probably already existed, since the UI is essentially reused. Most of the users found it hard to search for a specific emoji in the emojis table, resulting in nearly three more steps. Some of the users have suggested to delete a set of emojis that probably were not being used. Others have suggested the reformulation of the views into a more intuitive way, giving names to tabs. All these and other suggestions were kindly and gratefully received and both tips will be in mind either to review the feature or to develop another.

Still, the overall results were positive. For the performed tests we can be concluded that the features provide a good usability, and, even the feature that does not can be improved with users' suggestions.

6. Conclusion

This chapter concludes this report with a global overview of the work developed during this internship. In order to explain all the work done, it is divided into four different sections. The first section presents an overview about the internship from a goal perspective. The second one presents an overview about the trainee's contributions for this internship. Next, follows an analysis about the future work, addressing some suggestions for future improvements to the features that were done. The last section presents some final thoughts and lessons learned about the entire internship.

6.1 Overview

From a goal perspective, it is safe to refer that the internship has met the goals defined in its beginning. Here, a small analysis on each of the implemented features is provided.

Facebook Share

Initially, this feature was not in the internship's roadmap. However, its achievement was proposed to the trainee in order to be demonstrated in the Mobile World Conference (MWC) in Barcelona, in late February [68]. It was developed for both iOS and Android applications and the trainee has considered its development a major challenge. The reason for that is due to the fact that the feature should support two different operating systems. Each system has its own project in different languages and structures, so the trainee had to learn two different teams' methodologies (iOS and Android) in order to follow the product's guidelines.

The feature's first version used Facebook SDK to perform all the actions needed. By that time, both applications (iOS and Android) were supporting a simple share of a received image in a 1-to-1/group chat. After the share, an icon was used to indicate that the image was shared on the Social Network and the photo likes/comments were displayed as well.

According to the company's feedback, this feature was a huge success in the MWC and it was included in the product's roadmap for both platforms iOS and Android. Since being included in the product, it was necessary to improve some UX aspects, leading to its second version.

The feature's second version was implemented using the social native mechanisms provided by both iOS and Android. The feature's scope was also extended in order to support the video and text share. This new version has provided a user experience completely different, what charmed the people responsible for product. However, this feature had a drawback. The post likes/comments were impossible to be shown.

After some discussions, involving the trainee and the people in charge of the product, the versions were analysed and the second version's drawback was ignored, prevailing the user experience that the second version provides.

From a goal perspective, this feature implementation was automatically included in the product, what indicates that it was a huge success.

Internet Chat Services

This subsection intends to provide an overview about a set of features, more specifically the authentication mechanism, loading remote contacts with social information, contacts' presence, chat functionality and, at last but not least, the file transfer using a cloud storage service. It makes all sense to group these features, once they were implemented to work with each other in a perfect workflow.

The idea behind this integration is completely brilliant. Nowadays, more and more people are using the social networks as a communication way, giving up on the means used until then. As shown earlier in this document, the usage of traditional SMS service has been falling over the last few years due the emergence of new alternatives where the Social Networks are included. The Social Networks are also a service containing million of users providing a bunch of functionalities that could be integrated in third party applications. A successful integration can result on added value to the product and the company itself.

Regarding the internship, from a goal perspective, this integration was a success, once the outlined features were achieved. Since the authentication to the File Transfer, all the features were implemented with higher strictness, in order to enrich the WIT Software's RCS product.

The simple loading of remote contacts was transformed into a much more complex feature, that created opportunities to explore other types of functionalities using this information. The ability to include the presence in the contact details was a huge improvement by providing an amazing user experience to the end users in a communication application, because this feature gives a perception of the contact's availability near to the real-time.

Regarding the chat service itself, it was developed with the full specifications that the Facebook Chat API provides. As mentioned, this API limits the functionalities provided but, despite these limitations, it provides a good set of functionalities that brought added value to the company's product. Now, with this integration, a user is able to chat with its RCS contacts (via chat/SMS) and also with the Social Network contacts, thus extending the range of reachable peaople using the RCS product.

It is important to refer the Group Chat feature that was not implemented. This feature is very important and it would surely enrich the product. However, the Facebook Chat API does not provide it to third-party applications. This behaviour is somehow expected once Facebook also competes in this market. It has an OTT called Facebook Messenger that provides the full set of functionalities that Facebook chat provide. By providing all the features through a public API, they would be putting themselves in a delicate position that could lead to a decreasing use of Facebook Messenger, something that Facebook does not obviously want.

Another limitation about the Facebook Chat API is the fact that it did not allow to send any type of content except from text. To counter this setback, an alternative was found by taking advantage of Dropbox functionalities. As mentioned in the previous chapters, this cloud allowed the file transfer feature, as well as an unplanned functionality that has enriched the internship and, consequently, the company's product.

6.2 Contibutions summary

During this internship, the trainee was actively involved not only in the development of features that contributed to the enrichment of WIT's platform, but also in the design and decision making process. This work resulted in several significant contributions, which are put together and briefly described here.

At the application level, during the development, the trainee noticed some limitations and incoherencies on the iOS application. They were reported to the iOS development team along with a possible solution to solve them, ensuring the product's quality.

Regarding the implemented features, the trainee had always an active voice in all aspects that defined them. Some features had several discussions in background, in order to decide how they should be presented to the end user and an example of this was Facebook Share. The UI designed for this feature was on the trainee charge, where he made all the decisions. Even when the native implementation was chosen to replace the SDK implementation, the trainee's opinion had some weight. Here, it is important to mention that the Facebook Share feature is already in production, thus contributing for an enriched product provided to WIT Software's clients.

Another contribution/decision made by the trainee was the use of the Dropbox as a middleware to perform the file transfer feature. This idea came up in a discussion with some working colleagues where they were talking about the limitations that Facebook Chat API provides. The trainee already has some background knowledge acquired with the API study and when Dropbox came to discussion, the idea arose and it was transmitted to the people in charge of the product. They found the idea interesting and approved its development.

When the contacts' related features were developed, the trainee realized that Facebook information is very important and it could be used for a series of interesting things. Some market apps are actually using the Facebook information to provide their services to end users. Once we could gather this information, we could also provide these services through the RCS application, thus enhancing these services. So, functionalities such as Contacts Enrichment and Birthday alerts could be implemented. The first one consists of gathering information about the contacts in the user's mobile phone from Facebook, and use that content to update the contact profiles in the user's smartphone. The second functionality consists in reminding the contact's birthday date inside the app.

Finally, the experience exchanged with the product teams could be emphasized. It was an enriched experience that made possible to understand, discover and discuss several ideas about the iOS environment, architecture design and UX guidelines. During the development time, the trainee has contributed for an enriched product, which the development main focus was to optimize the UI responsiveness on loading several types of content to a view, and to test the application UI/UX on different scenarios with different conditions.

6.3 Future work

Since all the defined goals were achieved, there is not much more left to do as a future work. Once the implemented features rely on third-party services, as a future work, the APIs will be monitored in order to understand the changes that may possible influence them.

Regarding the usability tests, they should be analysed within a short time period. With some results, we can conclude that there are some aspects that could be improved at the UI level, especially the features related to the chat and the contacts functionalities. However, a larger study with a larger set of use cases should be made in order to create more viable conclusions. A good example of this is the use case defined for the chat feature. Most users took more than the expected actions to perform the specific use case and for some of them it was necessary to provide a hint. This means that the components structure could be reformulated, re-thinking how this feature should be re-integrated within the application and repeats the usability tests.

6.4 Final Remarks

This section represents the trainee's view of the internship. So, as an exception it is written in the first person in order to express all the final thoughts and lessons learned.

It is been almost a year since I had started the internship. Everyday I had to face new challenges and I was improving my personal knowledge facing unexpected situations and learning from my own mistakes. One of the main reasons that kept me motivated was the fantastic environment the company provides to its employees. I felt quite respected and I always had the chance to express my opinion regarding all decisions.

The final results were slightly deviated from what was proposed, since not all the features were possible to implement. However, the motivation was kept with the addition of a new feature, as explained in this the report – Share on Facebook.

Before this internship, its scope was significantly unknown to me. To have some know-how about that, I needed to develop very important skills of self-learning and initiative. I spent some of my free time reading the GSMA specifications in order to become more familiar with what I was dealing with. Despite this lost but still invested time, I can say that, in the end, it was a huge experience, a year full of lessons about programming patterns, structures and also a deeper insight on the Objective-C programming language that was a completely new world for me, when I started. I also developed skills at technical level, more specifically, on C++ programming language, which I had to learn while reading the existing code, to understand the basic mechanisms and architecture of the communications library.

With these final words, I can say with conviction that I became a better software engineer, understanding what the business reality is about and the real meaning of teamwork. I can also consider my performance a success because, while my internship was barely finished, I was included in a product development team that is developing solutions for the telecommunications company Vodafone.

Bibliography

1 WIT Software, SA. Company Profile | WIT Software, S.A. [Internet]. [cited 2014 January]. Available from: http://www.wit-software.com/company/company-profile/.

2 WIT Software, S.A. WIT Mobile Communicator. [Internet]. [cited 2014 January]. Available from: http://www.wit-software.com/products/wit-mobile-communicator/.

3 mobilesquared. OTT Services Blow Up the Mobile Universe. Operators Must Act NOW! [Internet]. [cited 2014 January]. Available from: https://www.tyntec.com/fileadmin/tyntec.com/whitepapers/Whitepaper_Operator-survey-OTT-blows-up-mobile-universe.pdf.

4 GSM Association. Rich Communications | Future Communications. [Internet]. [cited 2013 October]. Available from: http://www.gsma.com/futurecommunications/rcs/.

5 Rich Communications | Network 2020. [Internet]. [cited 2014 March 31]. Available from: http://www.gsma.com/network2020/rcs/.

6 Sync.ME - Magically update your contacts. [Internet]. 2014 Available from: http://www.sync.me.

7 GSMA. [Internet]. [cited 2013 October]. Available from: http://www.gsma.com.

8 Hibberd M. OTT app use undermining SMS revenue. [Internet]. [cited 2013 October]. Available from: http://www.telecoms.com/197721/ott-app-use-undermining-sms-revenue/.

9 WhatsApp. [Internet]. [cited 2013 October]. Available from: http://www.whatsapp.com.

10 The Verge. WhatsApp now has 350 million monthly active users. [Internet]. [cited 2013 October]. Available from: http://www.theverge.com/2013/10/22/4865328/whatsapp-350-million-monthly-active-users.

11 WhatsApp Messenger on the App Store on iTunes. [Internet]. [cited 2013 October]. Available from: https://itunes.apple.com/us/app/whatsapp-messenger/id310633997?mt=8.

12 WhatsApp Messenger on Google Play. [Internet]. [cited 2013 October]. Available from: https://play.google.com/store/apps/details?id=com.whatsapp.

13 The Next Web. Mobile messaging service WhatsApp now has 350 million active users each month. [Internet]. [cited 2013 October]. Available from: http://thenextweb.com/apps/2013/10/22/messaging-service-whatsapp-now-350-million-active-users-month/#!rdHVz.

14 WhatsApp. "WhatsApp users share 325 million photos per day! Awesome!". [Internet]. [cited 2013 October]. Available from: https://www.facebook.com/JoinWhatsApp/posts/675219532496843.

15 WhatsApp. "Why we don't sell ads". [Internet]. [cited 2013 October]. Available from: http://blog.whatsapp.com/index.php/2012/06/why-we-dont-sell-ads/.

16 Skype. Skype - Free Internet calls and online cheap calls to phones and mobiles. [Internet]. [cited 2013 October]. Available from: http://www.skype.com/en/.

17 Tech Radar. Microsoft highlights 299M Skype users, 1.5B Halo games played. [Internet]. [cited 2013 October]. Available from: http://www.techradar.com/news/software/operating-systems/xbox-live-upgrade-includes-300-000-servers-600-times-more-than-its-debut-1161749.

18 Skype for iPhone on the App Store on iTunes. [Internet]. [cited 2013 October]. Available from: https://itunes.apple.com/us/app/skype-for-iphone/id304878510?mt=8.

19 Skype on Google Play. [Internet]. [cited 2013 October]. Available from: https://play.google.com/store/apps/details?id=com.skype.raider&hl.

20 Wired. Microsoft Buys Skype for $8.5 Billion. Why, Exactly? [Internet]. [cited 2013 October]. Available from: http://www.wired.com/business/2011/05/microsoft-buys-skype-2/.

21 Skype Numerology. [Internet]. [cited 2013 October]. Available from: http://skypenumerology.blogspot.pt.

22 Skype Calling Rates - Cheap international calls & low-cost calling plans. [Internet]. [cited 2013 October]. Available from: http://www.skype.com/en/rates/.

23 Viber - Free calls, Free text messages, photo and location sharing. [Internet]. [cited 2013 October]. Available from: http://www.viber.com.

24 Tech Crunch. Viber Releases Update To BlackBerry App To Catch Up With iOS And Android. [Internet]. [cited 2013 October]. Available from: http://techcrunch.com/2013/10/23/viber-releases-update-to-blackberry-app-to-catch-up-with-ios-and-android/.

25 Viber on the App Store on iTunes. [Internet]. [cited 2013 October]. Available from: https://itunes.apple.com/us/app/viber/id382617920?mt=8.

26 Viber on Google Play. [Internet]. [cited 2013 October]. Available from: https://play.google.com/store/apps/details?id=com.viber.voip.

27 Viber Desktop. [Internet]. [cited 2013 October]. Available from: http://www.viber.com/products/mac/.

28 Facebook. Messenger. [Internet]. [cited 2013 October]. Available from: https://www.facebook.com/mobile/messenger.

29 Tech Crunch. Facebook Mobile User Counts Revealed: 192M Android, 147M iPhone, 48M iPad, 56M Messenger. [Internet]. [cited 2013 October]. Available from: http://techcrunch.com/2013/01/04/how-many-mobile-users-does-facebook-have/.

30 Facebook Messenger on the App Store on iTunes. [Internet]. [cited 2013 October]. Available from: https://itunes.apple.com/us/app/facebook-messenger/id454638411?mt=8.

31 Facebook Messenger on Google Play. [Internet]. [cited 2013 October]. Available from: https://play.google.com/store/apps/details?id=com.facebook.katana&hl=pt_BR.

32 Forbes. Facebook Messenger Takes On SMS, With No Account Needed. [Internet]. [cited 2013 October]. Available from: http://www.forbes.com/sites/tomiogeron/2012/12/04/facebook-messenger-takes-on-sms-with-no-account-needed/.

33 The Washington Post. Facebook Messenger app change allows free calls via WiFi. [Internet]. Available from: http://www.washingtonpost.com/business/technology/facebook-messenger-app-change-allows-free-calls-via-wifi/2013/01/17/4169e368-60a6-11e2-9940-6fc488f3fecd_story.html.

34 Apple. Messages. [Internet]. [cited 2013 October]. Available from: https://www.apple.com/ios/messages/.

35 Pocket-lint. Apple's popular iMessage and FaceTime services are suffering downtime. [Internet]. [cited 2013 October]. Available from: http://www.pocket-lint.com/news/120411-apple-imessage-down-april.

36 WWDC 2011 Keynote - iOS 5 iMessage and iOS 5 Summary. [Internet]. [cited 2013 October]. Available from: http://www.youtube.com/watch?v=y1qtY2_AP74.

37 Apple Insider. US DEA upset it can't break Apple's iMessage encryption. [Internet]. [cited 2013 October]. Available from: http://appleinsider.com/articles/13/04/04/us-dea-upset-it-cant-break-apples-imessage-encryption.

38 Welcome to Facebook - Log In, Sign Up or Learn More. [Internet]. [cited 2013 December]. Available from: https://www.facebook.com.

39 The Next Web, Inc. Facebook Now Has 1.19 Billion Monthly Active Users. [Internet]. [cited 2013 December]. Available from: http://thenextweb.com/facebook/2013/10/30/facebook-passes-1-19-billion-monthly-active-users-874-million-mobile-users-728-million-daily-users/#!qUTTn.

40 Facebook on the App Store on iTunes. [Internet]. [cited 2013 December]. Available from: https://itunes.apple.com/us/app/facebook/id284882215?mt=8.

41 Facebook on Google Play. [Internet]. [cited 2013 December]. Available from: https://play.google.com/store/apps/details?id=com.facebook.katana.

42 Guardian News and Media Limited. Facebook's value climbs above $100bn for first time. [Internet]. [cited 2013 December]. Available from: http://www.theguardian.com/technology/2013/aug/27/facebook-stock-price-record-high.

43 Facebook Help Center | Facebook. [Internet]. [cited 2013 December]. Available from: https://www.facebook.com/help/393277774048285.

44 Growing Social Media. Social Media Statistics and Facts of 2013 [INFOGRAPHIC]. [Internet]. [cited 2013 December]. Available from: http://growingsocialmedia.com/social-media-statistics-and-facts-of-2013-infographic/.

45 Google+. [Internet]. [cited 2013 December]. Available from: https://plus.google.com/.

46 Google+ on the App Store on iTunes. [Internet]. [cited 2013 December]. Available from: https://itunes.apple.com/app/google+/id447119634.

47 Google+ on Google Play. [Internet]. [cited 2013 December]. Available from: https://play.google.com/store/apps/details?id=com.google.android.apps.plus.

48 The Christian Science Monitor. Looking for a Google+ invite? Either get comfortable - or get crafty. [Internet]. [cited 2013 December]. Available from: http://www.csmonitor.com/Innovation/Horizons/2011/0630/Looking-for-a-Google-invite-Either-get-comfortable-or-get-crafty.

49 The Wall Street Journal. Google+ Pulls In 20 Million in 3 Weeks. [Internet]. [cited 2013 December]. Available from: http://online.wsj.com/news/articles/SB10001424053111904233404576460394032418286.

50 Surefire. KEY FEATURES & BENEFITS OF GOOGLE+. [Internet]. [cited 2013 December]. Available from: http://www.surefiremedia.co.uk/blog/benefits_of_google_plus/.

51 Twitter. [Internet]. [cited 2013 December]. Available from: https://twitter.com.

52 Business Insider. Twitter Total Registered Users V. Monthly Active Users. [Internet]. [cited 2013 December]. Available from: http://www.businessinsider.com/twitter-total-registered-users-v-monthly-active-users-2013-11.

53 Twitter on the App Store on iTunes. [Internet]. [cited 2013 December]. Available from: https://itunes.apple.com/us/app/twitter/id333903271?mt=8.

54 Twitter on Google Play. [Internet]. [cited 2013 December]. Available from: https://play.google.com/store/apps/details?id=com.twitter.android.

55 Twopcharts. Twitter activity monitor. [Internet]. [cited 2013 December]. Available from: http://twopcharts.com/twitteractivitymonitor.

56 Wikipedia, the free encyclopedia. Use of Twitter by public figures. [Internet]. [cited 2013 December]. Available from: http://en.wikipedia.org/wiki/Use_of_Twitter_by_public_figures.

57 Grandstaff P. Twitter Tutorial Two: 7 Important Twitter Features Explained. [Internet]. [cited 2013 December]. Available from:

http://www.petergrandstaff.com/marketing/twitter-tutorial-2.

58 Form S-1. [Internet]. [cited 2013 December]. Available from: http://www.sec.gov/Archives/edgar/data/1418091/000119312513390321/d564001ds1.htm.

59 Jeffbullas's Blog. 12 Awesome Social Media Facts and Statistics for 2013. [Internet]. [cited 2013 December]. Available from: http://www.jeffbullas.com/2013/09/20/12-awesome-social-media-facts-and-statistics-for-2013/.

60 The Agile Mindset. Scrum Roles. [Internet]. [cited 2014 January]. Available from: http://theagilemindset.wordpress.com/scrum-roles/.

61 Maxxor. Software Development Process. [Internet]. [cited 2014 January]. Available from: http://www.maxxor.com/software-development-process.

62 Scrum em 10 minutos. [Internet]. [cited 2014]. Available from: http://www.diariodarede.com.br/2013/11/25/scrum-em-10-minutos/.

63 XMPP Standards Foundation. RFCs – The XMPP Standards Foundation. [Internet]. [cited 2014 January]. Available from: http://xmpp.org/xmpp-protocols/rfcs/.

64 XMPP Standards Foundation. XMPP Extensions. [Internet]. [cited 2014 January]. Available from: http://xmpp.org/xmpp-protocols/xmpp-extensions/.

65 iOS App Programming Guide: App States and Multitasking. [Internet]. [cited 2014 May]. Available from: https://developer.apple.com/library/ios/documentation/iphone/conceptual/iphoneosprogrammingguide/ManagingYourApplicationsFlow/ManagingYourApplicationsFlow.html.

66 Relations EDP. An Average Facebook User Has 229 Friends. [Internet]. [cited 2014 June]. Available from: http://embracedisruption.com/2013/01/08/an-average-facebook-user-has-229-friends-100-social-stats-from-2012/.

67 Mac Rumors. iOS 8 to Include Battery Usage Per App and Much More. [Internet]. [cited 2014 June]. Available from: http://www.macrumors.com/2014/06/02/ios-8-tidbits/.

68 2014 P©. Wit Software leva produtos de comunicações integradas ao GSMA Mobile World Congress. [Internet]. [cited 2014]. Available from: http://tek.sapo.pt/noticias/telecomunicacoes/wit_software_leva_produtos_de_comunicacoes_in_882615.html.

69 Facebook. iOS - Facebook Developers. [Internet]. [cited 2014 January]. Available from: https://developers.facebook.com/docs/ios/.

70 RFCs – The XMPP Standards Foundation. [Internet]. [cited 2014 January]. Available from: http://xmpp.org/xmpp-protocols/rfcs/.

71 Facebook. Getting Started - Facebook Developers. [Internet]. [cited 2014 January]. Available from: https://developers.facebook.com/docs/ios/getting-started/.

72 Facebook. Login for iOS - Facebook Developers. [Internet]. [cited 2014 January]. Available from: https://developers.facebook.com/docs/ios/login-tutorial/.

73 Facebook. Understand Sessions. Login for iOS - Facebook Developers. [Internet]. Available from: https://developers.facebook.com/docs/ios/login-tutorial/#session-launch.

74 Understand Sessions. Login for iOS - Facebook Developers. [Internet]. Available from: https://developers.facebook.com/docs/ios/login-tutorial/#session-login.

75 Understanding Sessions. Login for iOS - Facebook Developers. [Internet]. Available from: https://developers.facebook.com/docs/ios/login-tutorial/#sessions.

76 XEP-0085: Chat State Notifications. [Internet]. [cited 2014 January]. Available from: http://xmpp.org/extensions/xep-0085.html.

77 XEP-0085: Chat State Notifications. [Internet]. Available from: http://xmpp.org/extensions/xep-0085.html.

78 XEP-0054: vcard-temp. [Internet]. [cited 2014 January]. Available from: http://xmpp.org/extensions/xep-0054.html.

79 Authentication X0NS. [Internet]. [cited 2014 January]. Available from: http://xmpp.org/extensions/xep-0078.html.

80 TechTarget. XMPP (Extensible Messaging and Presence Protocol). [Internet]. [cited 2014 January]. Available from: http://searchdomino.techtarget.com/definition/XMPP.

81 Mobile World Congress. Who Attends MWC. [Internet]. [cited 2014 January]. Available from: http://www.mobileworldcongress.com/who-attends/.

82 Peter Saint-Andre KSaRT. XMPP: The Definitive Guide. O'Reilly Media, Inc.; 2009.

83 SDWebImage. [Internet]. Available from: https://github.com/rs/SDWebImage.

84 Dropbox. Core API - endpoint reference. [Internet]. [cited 2013 December 30]. Available from: https://www.dropbox.com/developers/core/docs.

85 Apple. iOS App Programming Guide/ App States and Multitasking. [Internet]. [cited 2014 Jun 19]. Available from: https://developer.apple.com/library/ios/documentation/iphone/conceptual/iphoneosprogrammingguide/ManagingYourApplicationsFlow/ManagingYourApplicationsFlow.html.

86 WIT Software. WIT Software | RCS Suite. [Internet]. [cited 2014 Jun 19]. Available from: https://www.wit-software.com/products/rcs-suite/.