Masters in Informatics Engineering
Dissertation/Internship 2013/2014
Final Report

Integration of an iOS App with Social Networks and Cloud Storages

Nuno Miguel Carvalho Caixeiro
caixeiro@student.dei.uc.pt

Supervisors:
Frederico Lopes
Hugo Gonçalo Oliveira

June 21th, 2014

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

Abstract

Since its origin, the human being had the need to communicate with those around him. Nowadays, society fully relies on telecommunications in order to communicate with the ones they care, whether they are in the same city or in another continent.

With the massification of smartphones and tablets, and the proliferation of an easy mobile access to the Internet, a new menace arose to the Mobile Network Operators (MNOs): the Over-the-Top applications. Taking advantage of the high and worldwide usage of these mobile devices that are always connected to the global network, they provide new and different features that improve the way people communicate. Together with having no associated cost in general, these applications led users to rely less on the MNOs for the communication processes, which consequently affected their revenues.

In order to provide the MNOs with a solution that could deliver innovative value to the consumers, the GSM Association created the Rich Communication Services platform. This initiative is revealed in a product branded joyn, which aims to provide other features beyond voice and SMS, namely enhanced calls, media content share and instant messaging. Outsourcing companies, like WIT Software, develop RCS solutions for the mobile operators so they can have their own joyn applications and thus provide new features to their customers.

The purpose of this internship is to provide added value to WIT Software's *joyn* client application, so it can stand out from its direct competitors and thus be the first option for operators. It can be accomplished by integrating WIT's solution with social networks and cloud storages. By integrating social networks, users will be able to share content received during an instant messaging session, and enrich their contacts' details with information from their respective social networks profiles. In the other hand, integration with cloud storages allows them to share any file that is in the cloud, store a received media file in order to make it accessible anywhere and anytime, and create a safe backup of their conversations.

Keywords: Social Networks, Cloud Storage, Ephemeral Messaging, Real-time Location Share, iOS, joyn, Rich Communication Services, Telecommunications

Acknowledgments

First and foremost, the present report represents the end of a five-year chapter as a student of Informatics Engineering at the University of Coimbra that had not been possible without the continuous effort and sacrifice of the person whom I owe everything I am and have today – my mother. A special and eternal thanks.

A big thanks to all my friends that have always been there for me, independently of the mood and situation I had.

I would like to express my gratitude to both my supervisors, Frederico Lopes and Hugo Gonçalo Oliveira for their support and time spent along the entire internship.

Finally, I would like to thank to WIT Software for creating this opportunity and allowing me to work with such a great iOS team that helped and taught me a lot.

Table of Contents

Index of Figures

Index of Tables

Glossary

API	An Application Programming Interface is a group of methods or access points that a software provides so that external parties can make use of their functionalities without any knowledge of the intrinsic processes.
Burndown Chart	A 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.
Contact Enrichment	Contact Enrichment, in the scope of this project, consists in gathering information about the contacts in the user's mobile phone from social networks, and use that content to update the contact profiles in the user's smartphone.
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.
joyn	joyn is the brand created by GSMA to identify the applications that comply with their specifications in order 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 Interface/User Experience of that feature in the current product.
OTT Application	Over-the-Top refers to applications that deliver multimedia content over the Internet, like audio and video calls between two end users.
PoC	A Proof of Concept is a demonstration of the feasibility of certain feature inside a product, which is obtained through its successful implementation.
Product Backlog	Product Backlog is an artifact 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.
Product Owner	In Scrum, the Product Owner is the person responsible for defining the User Stories that compose the Product Backlog and assign them a priority in order to conduct the Scrum Team's work. The Product Owner can change the priority of those User Stories in order to maximize the value of the work by the delivery date.
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 and task boards. This tool was used in the project because it is widely used inside WIT Software and supports the Scrum framework.
Scrum	Scrum is an iterative and incremental agile development framework. This process is task priority-oriented, meaning that it seeks to fulfill tasks with higher priority first.
Scrum Master	Scrum Master is the responsible to help both Product Owner and Team Members. The Scrum Master helps the Product Owner to manage the Product Backlog, and helps the Scrum Team to mark all the User Stories in a Sprint as done, by removing impediments and keeping them focused and productive.
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 to four weeks; in this project, each Sprint lasts two weeks.
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, which detail the steps the Team must take in order to fulfill a story.
Task	A Task is a smaller assignment, subset of a User Story, which states something that must be implemented as part of that feature. 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.
Team Member	The Development Team Members are responsible to create the incremental changes on the product that match the User Stories assigned to each development Sprint. Team Members forecast the User Stories that they can deliver at the end of the Sprint, and also decide how they will be implemented.
User Story	A User Story defines a possible user action with the product in order to state a project requirement in the everyday 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.
VCard	File format standard for electronic business cards. They can contain name and address information, phone numbers, and email addresses.

Acronyms

AES	Advanced Encryption Standard
API	Application Programming Interface
CMS	Central Message Storage
DoD	Definition of Done
GSM	Global System for Mobile Communication; originally Groupe Special Mobile
GSMA	GSM Association
IP	Internet Protocol
MMS	Multimedia Messaging Service
MNO	Mobile Network Operator
OS	Operating System
OTT	Over-the-Top
PC	Personal Computer
PoC	Proof of Concept
REST	Representational State Transfer
RCS	Rich Communications Services
SIM	Subscriber Identity Module
SMS	Short Message Service
SSL	Secure Sockets Layer
UI	User Interface
URL	Uniform Resource Locator
USA	United States of America
UX	User Experience
VoIP	Voice over IP
WMC	WIT Mobile Communicator
WWDC	Apple Worldwide Developers Conference

1. Introduction

The present document, together with all the appendices, is part of the work done at WIT Software during the one-year internship. The internship work was supervised by Hugo Gonçalo Oliveira, PhD professor at the Department of Informatics Engineering of the University of Coimbra, and Engineer Frederico Lopes, Project Manager at WIT Software.

This introductory chapter is organized in five sections. The first section presents the context of this work in the area of telecommunications and its current trends. The second section presents the grounds that gave origin to this internship and the reasons of its importance. The third section contextualizes the proposed work in WIT Software's roadmap. The goals of this work are defined in the fourth section. The fifth section describes the purpose of each chapter, describing as well all the documents that append this report.

1.1. Context

In 1983, Motorola released the first commercial portable cellular phone. This event revolutionized the way people talked, allowing them to communicate on the move. Over the following years, this kind of devices evolved: they got smaller, thinner, lighter, more powerful, and their autonomy increased. This technological evolution was accompanied by the release of new communication features provided by the Mobile Network Operators (MNOs). The appearance of the Short Message Service (SMS) allowed people to exchange short messages with their contacts, in a quicker and cheaper way than via the traditional voice call communications. The evolution of the SMS originated the Multimedia Messaging Service (MMS), allowing people to share multimedia contents (photos and videos). Both these services were quickly adopted all over the world, becoming an important source of revenue to the MNOs.

Smartphones, which are mobile phones built on a mobile operating system (OS), have emerged less than a decade ago. This equipment has more advanced computing power and connectivity than the typical cellular phone, providing a series of advanced features offered by third party applications running on the OS, just like in a computer. These devices revolutionized the way people communicate, allowing them to share easily any kind of information with an Internet connection. The success of these devices is revealed in Figure 1: the sales of smartphones are increasing exponentially, unlike the PCs that are slowly decreasing. In 2013, 1.81 billion mobile phones were sold, 55% of which are smartphones [1]. At the moment, Android and iOS devices are the market leaders, with 32.1% and 12.1% respectively.

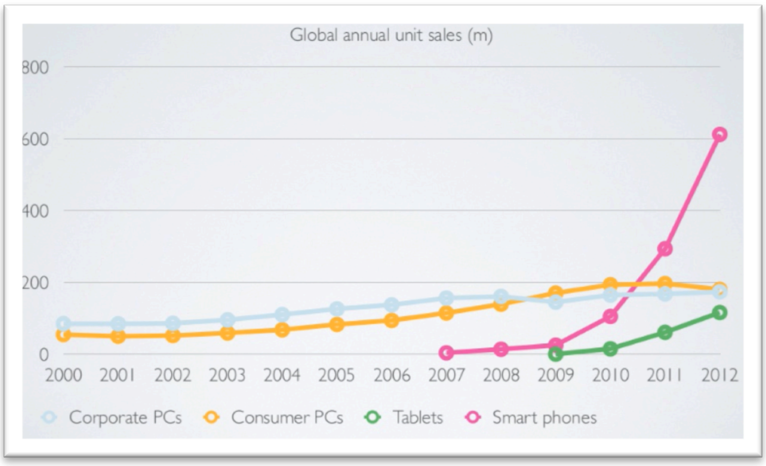


Figure 1 - PCs vs. mobile devices annual sales [2]

1.1.1. New ways of communication

The proliferation of smartphones and mobile Internet access all over the world granted the opportunity for the appearance of a new generation of mobile services. In the area of communication, new ways of talking and messaging have emerged, which are threatening the MNOs: the Over-The-Top applications (OTT). An OTT is "any app or service that provides a product over the Internet and bypasses traditional distribution. Services that come over the top are most typically related to media and communication and are generally, if not always, lower in cost than the traditional method of delivery" [3].

Why are they threatening the MNOs? Generally, they are freemium services that provide the basic communication features (voice and video calls, and chatting) free of charges. However, users can access extra features or contents by purchasing plans or subscriptions inside the application. Thus, they are the reason users make less traditional calls and send less SMS/MMS, resulting on huge losses of revenues to the operators. According to [4], SMS generated worldwide revenues of 15.3 million dollars per hour, against the only 2.6 million dollars per hour generated by the OTTs, in 2013. This high difference is justified: OTTs are only available to smartphones and not everybody has one, while everybody with a mobile phone can send SMS. However, this article predicts that OTT messaging traffic will increase exponentially in the next few years, reaching the 32 trillion messages exchanged annually against the 7.89 trillion SMS by the end of 2017.

The OTTs are a real menace and the MNOs must innovate in order not to lose more market share.

1.1.2. The Rich Communications project

Formed in 1995, the GSM Association (GSMA) "is an association of mobile operators and related companies devoted to supporting the standardizing, deployment and promotion of the GSM mobile telephone system" [5].

In order to provide a solution to the MNOs so they could deliver innovative value to the consumers and, consequently, generate more revenues, GSMA defined a "platform that enables the delivery of communication experiences beyond voice and SMS, providing consumers with instant messaging or chat, live video and file sharing – across devices, on any network" [6] – the Rich Communications Services (RCS). Following the adoption of

4G, "RCS marks the transition of messaging and voice capabilities from Circuit Switched technology to an all-IP world" [6]. In other words, RCS allows the communication via Internet, just like the OTTs, as well as via the typical technology used in the voice calls and SMS exchange.

This project is of great importance and interest for the operators because it enriches their services for the users, offering them new and competitive features, and ways of being connected. "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" [6]. Besides the improvement of the user experience, MNOs have a huge advantage comparing to the OTTs: while each of those communication services do customer segmentation once they only allow the communication between users of that service, the operators allow interoperability, meaning that their customers can communicate with any person in the world that has a SIM card.

As shown in Figure 2, this platform is strongly supported by the MNOs and device manufacturers, existing in 17 countries at the moment, and counting with commitment of 85 operators to launch it by 2015.

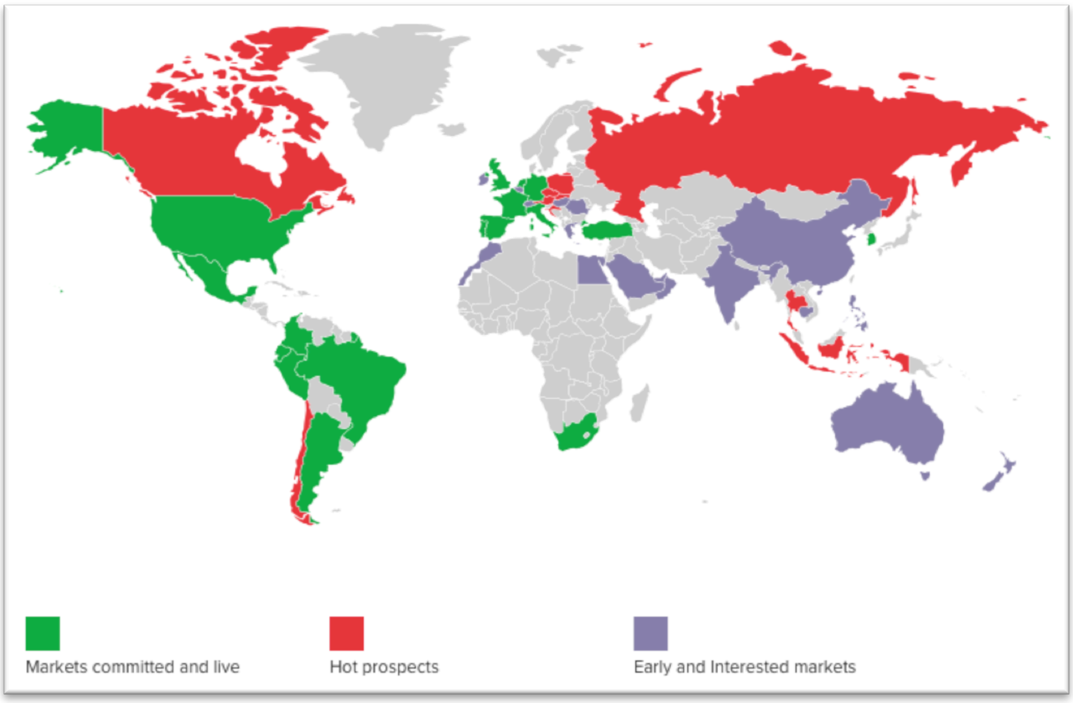


Figure 2 - RCS market status [6]

1.1.3. RCS Specifications and Product Definition

The presence of RCS services is identified through the joyn brand. The presence of this brand's logo informs that devices and applications have been accredited by GSMA.

In order for MNOs to implement the necessary infrastructures, and companies to develop RCS-based client applications, the Global Specification Group within the RCS project, composed by the leading operators, have design a set of specifications for the RCS platform. These specifications are regularly updated and their objective is to "provide a framework for discoverable and interoperable advanced communication services and functionality" [7]. The most recent set of specifications is contained in RCS 5.1 v4.0 document.

joyn branded applications must follow the Product Definition documents. These documents are directed to device manufacturers and application developers, with the following goals:

- Define the product and include some definition of the User Experience (UX), based on the latest RCS feature specifications;
- Provide guidance on how the features should be implemented.

The two product definition documents that indeed define how the joyn applications must be are joyn Hot Fixes and joyn Blackbird, the latest being the most recent release based on RCS 5.1 specifications. However, in the present year the Crane version will be publicly released, with new features for the joyn branded products.

1.1.4. Social Networks

In these days, social networks are a fundamental part of almost everyone's lives. Through these communities people are able to communicate more easily, to share their thoughts and feelings, and even to meet new people. They allow us to be more present in the important events of our friends, to be constantly up to date about the news, and even to find that friend from long time ago. Were they necessary ten or twenty years ago? Sure not! Although, in the present days people are often required to leave their home locations, and consequently their loved ones, for many reasons. This is where these communities play an important role by keeping people in touch.

As there is no specification regarding the integration with social networks on the latest joyn release – Blackbird -, and as social networks are of great importance to the users, WIT Mobile Communicator (WMC) for iPhone would be highly enriched with the implementation of this feature. Furthermore, WIT will have one more chance to suggest new specifications if this feature is revealed successful, increasing even more its success and worldwide recognition.

1.1.5. Cloud Storages

Mobility is one of the principles that rule the world nowadays, leading people to be always in movement. Due to that, people started to adopt cloud storage services as their main repository of information in order to access it anywhere in the world. Users won't have to worry anymore about having that pen drive with them all the time or buying a larger hard disk in order to save all their data. There are cloud storage services for everyone's needs.

As revealed in Figure 3, the market for cloud storage services has grown notoriously, reaching 625 million Internet users until the end of 2013. It's predicted that more users will start using this kind of services, surpassing the 1 billion users mark in 2016.

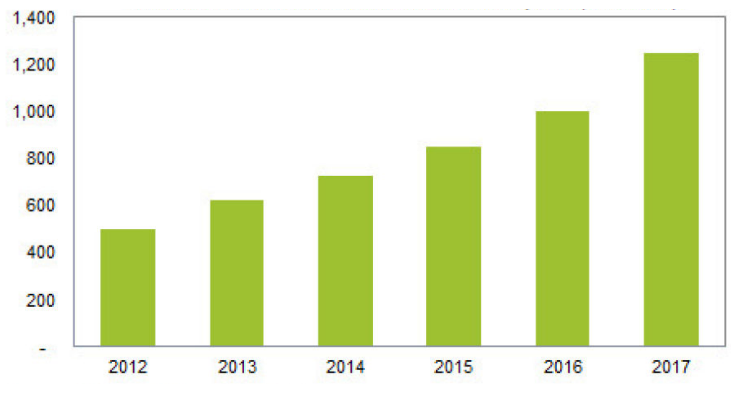


Figure 3 - Worldwide forecast for personal cloud subscriptions (in Millions) [8]

This way, the integration of a cloud storage service with the WMC for iPhone is of great importance once it will greatly improve the way a user shares contents: they will be able to send photos and videos that are not on their mobile devices (which is usual nowadays), and to store received contents on a cloud, making them accessible anytime and anywhere.

1.1.6. Communication trends

Until the previous decade, the communication was all about voice calls and SMS exchange. Nowadays, with all the capabilities provided by the smartphones, the communication process has evolved, wrapping a whole new set of possibilities. Presently, there are two communication features that are very popular among the smartphone users: the exchange of ephemeral messages, and the share of location in real-time location.

Ephemeral Messaging

One of the usages of the Internet is for communication purposes. Although it is extremely useful and versatile, any action we have on the Internet leaves a digital footprint that stays stored in third-party servers across the globe. This issue did arise concerns regarding the loss of privacy. This way, the concept of ephemeral messaging appeared in order to allow the Internet users to have privacy while communicating. It consists of the exchange of messages and media files that have a short life span and are not stored in the servers. After the validity of the message expires, it is erased from the device, leaving no trace of their existence.

Real-time Location Sharing

Today's smartphones are equipped with GPS chips, making it possible to determine the current position of the device. This capability allows the existence of location-based contents inside the smartphone, such as navigation, and sport and fitness applications.

"How long until the taxi arrives?", "I'm hungry! How long will that pizza take to be delivered?" are some of the stressful questions that the society faces nowadays. Both of them have in common the desire of knowing the location of something at a certain moment. This problem can be solved using a real-time location solution, which mainly relies on the smartphone's GPS chip. The device determines periodically its location in order to share it with someone and, consequently, enable the receiver to keep track of the sender's route.

The implementation of these features on WIT's RCS-based solution plays an important role on its innovative process once it will equip the application with new communication capabilities that are desired and applauded by the consumer market. Thus, this will provide more reasons for this solution to be attractive for the MNOs once it will be well received by their clients.

1.2. Motivation

As stated in the previous section, the OTTs are threatening the MNOs due to their innovative features. These features attract hundreds of millions users worldwide, providing them with other ways of communication, mostly free of costs. This way, users decreased their high dependency on the MNOs to communicate which led to lower revenues by the MNOs.

In order to fight this menace and provide a solution to the operators, GSMA created a service capable of providing similar features to those provided by the OTTs: the Rich Communication Services. This initiative is promoted under the brand "joyn" for both accredited applications and mobile devices, meaning that they fulfill all the defined specifications once they are approved by GSMA.

At the moment, some MNOs already provide joyn branded applications, which were produced by outsourcing companies specialized on the development of mobile applications and experienced in the Telco business. WIT Software is one of these companies, and this internship aims to contribute for the enrichment of its joyn application for iPhone: WIT Mobile Communicator. Besides enriching the product, this internship aims for the innovation its innovation as it focuses on features that are not present on any of the current specifications: integration with social networks and cloud storage and, more recently, ephemeral messaging and real-time location share. The implementation of these features will provide business advantage to WIT Software because none of its competitors has publicly revealed to provide such features.

All these features are believed to be a success and highly desired by the final consumers. On the one hand, hundreds of millions of people use social networks on a daily basis for the most varied purposes. We can say that social networks are becoming essential tools in people's lives. On the other hand, cloud storages are widely used as they represent a secure and centralized storage of information, ending with the common situation where people forget or lose their pen drives full of important documents. Regarding ephemeral messaging and the sharing of real-time location, both these features are considered to be successful because they provide great solutions for real-life problems that are shared by the population, namely the privacy issues while messaging and the impossibility of following someone's route, respectively.

1.3. Scope

The internship is taking place at WIT Software, a software company headquartered in Coimbra, Portugal, that is specialized in advanced solutions for mobile telecommunications companies. Since December 2011, WIT is a member of GSMA and, consequently, of the RCS project, where it helps to improve the specifications and develops joyn branded client applications.

1.4. Goals

The goals of this work can be divided into two major interconnected parts:

- Internship, which focus on the experience and knowledge gained by the intern;
- Project, which corresponds to the implementation of all the proposed features.

1.4.1. Internship

From the internship angle, the goals are to consolidate the knowledge about Software Engineering, learn and master the Objective-C language used in the development of iOS applications, and gain experience on developing software in a corporate environment where commitment and team work are essential to produce an excellent piece of software, which will be delivered to an actual client.

1.4.2. Project

Regarding the project, the main goal is to contribute for the enrichment of WIT's WMC for iPhone during the internship. To accomplish this, the following features must be fully implemented and tested to guarantee the existence of zero bugs in the product:

- Integration with Social Networks: allow the user to interact with a social network within WMC in order to share contents, and enrich its contacts' details with information from their respective social networks profiles;
- Integration with Cloud Storages: provide the user with access to cloud storage services, so that they can retrieve any hosted file and share it, store a received video in order to make it accessible anytime and anywhere, or do a safe backup of their conversations.
- Ephemeral Messaging: allows the user to send text messages and share images, videos and audio clips that are only visible for a small amount of time. After the message has been previewed, it is erased, leaving no trace of its existence.
- Real-time Location Share: provides a way of sharing the user's location in real-time with someone, allowing the receiver to track the sender's route on a map.

These two last features were added during the internship in order to embrace the current trends. The main goal for both these features was to be demonstrated at the world's largest exhibition for the mobile industry, the Mobile World Congress. This way, they were the first features to be designed and implemented by the trainees because they needed to be ready in time for that event.

Furthermore, it is required to do planning job before the implementation, namely the following tasks:

- Requirements analysis – identify, discuss and prioritize all the product requirements;
- Mockuping – create a visual design draft that represents the structure of information, visualizes the content and demonstrates the features in a static way. This way, it encourages the team to review the UI/UX side of the features.
- Architecture definition – define how the features will work internally; understand which components must be created or modified, and how they will communicate with each other within the application.

1.5. Document structure

Besides this introductory chapter, the document is divided according to the following chapters:

- 2. State of the Art: presents the analysis of both direct and indirect competitors, as well as the most popular social networks and cloud storages at the moment;
- 3. Approach: describes the approach followed in the internship, from the development methodology to the risks that may affect the project;
- 4. Architecture: presents the architecture of the proposed features, and provides an overview of some technical decisions made;
- 5. Implementation Challenges and Evaluation: analyzes the implementation stage, specifically the problems and challenges found, and the validation of all the work along the internship;
- 6. Conclusion: concludes this report by providing an overview of all the work done, suggesting the work that can be done in the near future, and presenting a personal insight about the internship.

The work described in this report is complemented with the following appendices, which contain more sensitive contents for the company, in terms of confidentiality:

- **Appendix A – State of the Art**: contains the complete study of the State of the Art, including all the studied features and the 24 OTTs analyzed;
- **Appendix B – Approach**: provides information about the methodology used in this project. Complements Chapter 3 by detailing some aspects that were only briefly address;
- **Appendix C – Architecture**: contains the complete study of the necessary Application Programming Interfaces (APIs) and definition of both high and low level architectures for the various implemented features;
- **Appendix D – Evaluation**: contains the complete and detailed set of different kind of tests performed to the implemented features.

2. State of the Art

This chapter is a summary of the complete study of the State of the Art, which is consisted by the following analyses:

- Competitors of the WIT Mobile Communicator by WIT Software, whether direct or indirect ones, in section 2.1;
- Most popular social networks, in section 2.2;
- Most popular cloud storage services, in section 2.3;
- Most popular ephemeral messaging applications, in section 2.4;
- Most popular real-time location share applications, in section 2.5.

For each of these analyses a brief description of the applications/services is provided, as well as some information about its users, main features and rating in the main mobile stores. Each of the following sections contains only the most important and remarkable services or applications in the corresponding area, sorted in alphabetical order. For the complete study, which even contains even the services and applications that are less known by the general public, refer to *Appendix A – State of the Art*.

It is important to clarify two aspects before proceeding to the next sections:

- Whenever possible, the Monthly active users are used as the main success metric, once they show how many users actively use the service per month. Although, when such information is not available, we relied on the Registered users metric. This one isn't so accurate since it only reveals the amount of accounts created, which considers the number of accounts used actively and those that are barely or never used;
- In order to analyze what users think about the services' mobile applications, both Apple's App Store USA and Google Play Store ratings were extracted from the official websites, once they are the most used mobile application stores. The chosen App Store was USA's because it is considered a reference worldwide. Although, there are some occasions where it is relevant to present App Stores from other countries, mainly when the application is directed to a specific country or region.

2.1. Competitors

WIT Software and its competitors provide features that hugely increase the user experience while communicating. Section 2.1.1 describes these new features.

Section 2.1.2 and 2.1.3 contain the analysis of WMC's direct and indirect competitors, respectively.

2.1.1. New and powerful features

Nowadays, the typical voice calls, SMSs and MMSs are no longer sufficient for the general population. The following listing contains some of the features that enrich the communication process and that are highly valued by the users. A complete listing of the analysed features can be found in section 1.1 of Appendix A.

- <u>Contact Discovery</u>: identification of contacts from the native address book that use the intended application.

- Network Address Book: capability of organizing and managing all the contacts through a single interface that makes them available in a wide range of services and devices.
- Chat: instant messaging session between two people.
- Group Chat: chat with more than two people simultaneously. In this type of messaging, all the users belonging to the conversation can see everything that is written there.
- Message State Notifications: show the state of the messages sent, that is, if they were sent, delivered or read.
- Typing Notifications: show when a contact is typing a message.
- File Transfer: exchange of multimedia files, either stored on the device or captured in the moment. This can be done either inside a chat or group chat.
- Location Sharing: share information about the user's location. This information is represented by latitude and longitude, which may have associated text data relating to the location's name.
- Contact Share: ability to share vCards stored in the device.
- VoIP Calls: voice calls between two people that run over the Internet instead of being conducted through the traditional copper lines. The only cost is the one associated with the Internet access.
- Video over IP Calls: calls with voice and video between two people. The shared video is from one of the device's cameras.
- Voice Messages: send audio clips during a chat session.
- Video Messages: send video clips during a chat session.
- Stickers: send and receive "illustrations or animations of characters" [9] during a chat session.
- Emoticons: send and receive emoticons during a chat session.
- Emojis: send and receive emojis during a chat session.
- Presence: information that describes the status of a certain contact. It can assume many values: "online", "busy", "offline", etc.
- Profile Picture: associate a photo to a contact profile.
- Themes/Skins: change the layout and appearance of the application.

The following features are the core of the internship. It is of extreme importance to define them and to verify if they are present in any of the applications/services under analysis.

- Integration with Social Networks: ability to authenticate, share contents, access timelines or enrich contacts using a social network.
- Integration with Cloud Storages: ability to interact with a user's cloud account in the application. This can be useful for sharing a photo stored in the cloud, save a received file, or even backup and restore existing conversations.

- Ephemeral Messaging: ability to exchange media with a short lifetime. After its expiration, the message are erased from the devices, being no longer accessible.

- Real-time Location Share: ability to share its geo-location position continuously with someone during a defined amount of time. The set of shared positions can be seen on map, which is updated in real-time.

2.1.2. Direct competitors

WIT Software is the only Portuguese company that develops RCS-based client applications. However, there are other international companies that do the same. What is their goal? To develop, in the least time possible, the most complete RCS client with, at least, all the features defined in the latest GSMA specification. This way, they are considered to be direct competitors because they compete directly against WIT Software. Those are:

The following list contains the companies that are considered as direct competitors.

- Jibe Mobile, Inc. – California, USA;
- Nable Communications, Inc. – South Korea;
- Neusoft Corporation – China;
- Summit Tech Communications – Montreal, Canada.

More information about these companies, as well as a comparison with all the studied features, are available in section 1.2 of Appendix A.

Comparative Analysis

We have compared each direct competitor in terms of provided features (Table 1) and supported operating systems (Table 2). This comparison is of great important once it allows to discover what are the strengths and weaknesses of WMC against its competitors, as well as to know if anyone already implements the proposed features of this internship.

Before starting the comparison, it is important to clarify the meaning of the symbols that appear on both tables:

- ✓ - feature is provided;
- - feature is not provided;
- **?** – there is no public information about this feature. This occurs when we did not managed to get access to some applications and we only know about the publicly documented features.

	WIT Software	Jibe Mobile	Nable Communications	Neusoft	Summit Tech
Contact Discovery	✓	✓	✓	✓	✓
Network Address Book	✓	?	?	?	✓
Chat	✓	✓	✓	✓	✓
Chat Message State Notifications	✓	✓	✓	✓	✓
Chat Typing Notifications	✓	✓	✓	✓	✓
Group Chat	✓	✓	✓	✓	✓
Group Chat Message State Notifications	✓	✓	✓	✓	✓
Group Chat Typing Notifications	✓	✓	✓	✓	✓
Chat File Transfer	✓	✓	✓	✓	✓
Group Chat File Transfer	✓	✓	✓	✓	✓
Location Sharing	✓	✓	✓	✓	✓
Contact Share	✓	✓	✓	✓	✓
VoIP Calls	✓	✓	✓	✓	✓
Video over IP Calls	✓	✓	✓	✓	✓
Voice Messages	✓	?	?	?	?
Video Messages	✓	?	?	?	?
Stickers	✓	✗	✗	✗	✗
Sticker Store	✗	✗	✗	✗	✗
Emoticons	✓	✓	✓	✓	✓
Emojis	✓	✓	✓	✓	✓

	WIT Software	Jibe Mobile	Nable Communications	Neusoft	Summit Tech
Presence	✗	✗	✓	✓	✓
Profile Picture	✓	✗	✓	✓	✓
Themes/Skins	✓	✗	✗	✗	✓
Integration with Social Networks	✓[1]	✗	✗	✗	✓[2]
Integration with Cloud Storages	✗	✗	✗	✗	✗
Ephemeral Messaging	✗	✗	✗	✗	✗
Real-time Location Share	✗	✗	✗	✗	✗

Table 1 - Direct competitors' provided features

	WIT Software	Jibe Mobile	Nable Communications	Neusoft	Summit Tech
Android	✓	✓	✓	✓	✓
iPhone	✓	✓	✓	✓	✓
iPad	✓	✗	✓	✓	✓
BlackBerry	✗	✗	✗	✓[3]	✗
Windows Phone	✗	✗	✗	✓[3]	✓
Symbian	✗	✗	✗	✓[3]	✗
Windows	✓	✗	✗	✓	✓
Mac	✗	✗	✗	✓	✓

Table 2 - Direct competitor's supported platforms

This analysis shows that WIT Software is very well placed in the production of RCS-based client applications. Besides providing many differentiating features, its application supports

[1] only posts an advertising message on Facebook

[2] has extensions that give social networks presence information

[3] produced only if requested by a client

the major mobile operating systems. However, it can't stop the development and enrichment of its RCS product once the competition, namely Summit Tech, is pretty powerful as well. This way, the internship plays a big role on WMC's enrichment process: all the features that compose it are an unexplored area in this kind of services, so it is of the most importance to implement these functionalities in a way that it highly privileges the user experience.

2.1.3. Indirect competitors

The current section presents the OTT applications. Why are they indirect competitors of WIT Software? Although they aren't RCS-based applications, their success "steals" clients from the MNOs. Once the *joyn* branded applications are developed to the operators, companies like WIT Software consider the OTTs as indirect competitors, having to implement more and better features in order to reach the users and bring them back to the mobile operators.

In order to keep it short, this section will only approach the 5 more popular applications. The analysis of the 24 studied OTTs can be read in *Appendix A – State of the Art: Indirect Competitors.*

Facebook Messenger

Facebook Messenger is the instant messaging mobile application from the most used social network in the world - Facebook. In order to make the chatting more friendly and easier on mobile devices, Facebook Messenger was initially released on August 9, 2011 for iOS and Android, having only chat functionalities. Currently, the support has been extended to many other mobile operating systems and VoIP calls are now possible between Facebook users. Moreover, in some countries, users can use their phone number and name in order to log in and send SMS [10]. This way, Facebook intends to compete with the mobile network operators and to "recruit" more people for its social network.

<u>Monthly Active Users on Facebook Mobile products</u>: 874 million [11].

<u>Monthly Active Users on Facebook Messenger</u>: 56.7 million [12].

<u>Main Features</u>: [13]
- Get to your messages without opening Facebook;
- Bring your conversations to life with stickers;
- Send photos privately;
- Have group conversations and make plans on the go;
- Text your phone contacts, even if you're not Facebook friends;
- Share your location so people know when you're nearby;
- Record voice messages;
- Make free calls, even to friends in other countries;
- Know when people have seen your messages;
- See who's available on Messenger and who's active on Facebook;

<u>App Store Rating (USA)</u>: 4.5/5 stars (136241 Ratings) [13]

<u>Google Play Store Rating</u>: 4.3/5 stars (1963758 Ratings) [14]

iMessage & FaceTime

Both iMessage and FaceTime are proprietary communication services by Apple, Inc. They allow Apple device owners to communicate with each other using their iCloud accounts.

FaceTime was announced by Steve Jobs during the WWDC 2010 [15]. Initially, it only allowed Video over IP calls and was only supported by iOS devices. Nowadays, it allows VoIP calls as well.

iMessage, Apple's instant messaging service, was announced one year later at the WWDC 2011 keynote [16] by Scott Forstall. It allows cost-free chatting using an Internet connection. Although, the message can be sent via SMS or MMS if the recipient isn't an iCloud user or doesn't have an Internet connection. As FaceTime, it is currently available for the entire Apple device family.

Both these services are available for any Apple equipment, either mobile device or Macintosh computer.

<u>Monthly Active Users</u>: 250 million [17].

<u>Main Features:</u>

- Contact Discovery;
- Video over IP calls;
- Single and Group chat;
- Typing and Message Delivery Notifications
- File, location and vCard share;
- Contact block.

Skype

Skype was created by a group of Estonian developers, the same that developed Kazaa. On August 2003, Skype had its first release for Windows desktop. This application was, and still is, a huge success once it allowed not only VoIP/Video over IP calls and conferences, but VoIP calls to mobile and landline numbers all over the world. This last functionality was a revolution on the telecommunication market because it made possible to call anywhere paying much less than doing a traditional call via MNOs.
All this success brought the attention of many multinational companies, leading Microsoft to buy Skype for 8.5 billion dollars, in 2011 [18]. Currently, Skype supports every major mobile OS, as well as Windows, Mac and Linux computers.

<u>Monthly Active Users</u>: 299 million [19].

<u>Main Features</u>: [20]

- "Calling": make free Skype to Skype calls or call mobiles and landlines home and abroad at low rates;
- "Video": catch up face to face or get a whole group together on a video call;
- "Messaging": you're always in the loop with instant messaging, voice messaging and sending texts;
- "Sharing": send photos, videos and files of any size.

<u>App Store Rating (USA)</u>: 3.5/5 stars (355038 Ratings) [21]

<u>Google Play Store Rating</u>: 4/5 stars (1942897 Ratings) [22]

Viber

On February 2010 Talmon Marco founded Viber Media, a company located in Cyprus. It is responsible for the creation and development of Viber, an instant messaging and VoIP application that aimed to compete with Skype. It was initially launched for iPhone on December 2, 2010, followed by a pre-release version for Android on May 2011. Nowadays, it supports all the major mobile operating systems, as well as PC running Mac OS or Windows.

Registered users: 200 million [23].

Main Features: [24]

- Free text, calling, photo messages and location-sharing with Viber users;
- No registration, alias or invitations required;
- Instantly integrates with your own contact list;
- Best-quality mobile calls using 3G or Wi-Fi.

App Store Rating (USA): 4.5/5 stars (95054 Ratings) [25]

Google Play Store Rating: 4.4/5 stars (2027710 Ratings) [26]

WhatsApp

Brian Acton and Jan Koum, both former Yahoo engineers, founded WhatsApp, Inc. in 2009. This California-based company developed the instant messaging application named WhatsApp. It has always been growing since the beginning, even though having fierce competitors based in Asia (KakaoTalk, WeChat). It provides support for all the major mobile operating systems. However, it fails on the iOS support once it only works on iPhone. Currently, the service is free for the first year, and $0.99 for each following year. This application is such a well succeed phenomenon that even Financial Times wrote about it: WhatsApp "has done to SMS on mobile phones what Skype did to international calling on landlines" [27].

Monthly Active Users: 400 million [28].

Main Features: [29]

- Group chat up to 50 participants;
- Send photos, videos, and audio messages;
- Share locations;
- Ads free.

App Store Rating (USA): 4.5/5 stars (120726 Ratings) [30]

Google Play Store Rating: 4.5/5 stars (6291278 Ratings) [31]

<u>Comparative Analysis</u>

After identifying and describing the main indirect competitors, follows a comparison between them in terms of provided features (Table 3) and supported operating systems (Table 4). This comparison is of extreme importance once it allows to conclude what are the features users like the most, and what must be implemented in WMC in order to get leverage against these popular services. Morevoer, this study allows checking the degree of integration these applications have with social networks and cloud storage services. This way, it is an opportunity to verify how important this internship is for the enrichment of WMC product, once it allows the implementation of features that may be differentiating.

Before proceeding to the comparisons, it's important to point two issues:

- Table 4 only considers the access via an official application, excluding any kind of web browser interaction;
- An asterisk (*****) appearing on the feature "Integration with Social Networks" refers to a partial integration (e.g. authenticate with Facebook account, publish an advertisement, etc.).

Contact Discovery	✓	✓	✓	✓	✓
Network Address Book	✓	✓	✓	✓	✓
Chat	✓	✓	✓	✓	✓
Chat Message State Notifications	✓	✓	✓	✓	✓
Chat Typing Notifications	✓	✓	✓	✓	✓
Group Chat	✓	✓	✓	✓	✓
Group Chat Message State Notifications	✓	✗	✓	✓	✓
Group Chat Typing Notifications	✓	✗	✗	✓	✗
Chat File Transfer	✓	✓	✓	✓	✓
Group Chat File Transfer	✓	✓	✓	✓	✓
Location Sharing	✓	✓	✗	✓	✓

Contact Share	✗	✓	✓	✓	✓
VoIP Calls	✓	✓	✓	✓	✗
Video over IP Calls	✗	✓	✓	✗	✗
Voice Messages	✓	✗	✗	✗	✓
Video Messages	✗	✗	✓	✓	✗
Stickers	✓	✗	✗	✓	✗
Sticker Store	✓	✗	✗	✗	✗
Emoticons	✓	✗	✓	✓	✓
Emojis	✓	✓	✓	✓	✓
Presence	✓	✗	✓	✓	✓
Profile Picture	✓	✓	✓	✓	✓
Themes/Skins	✗	✗	✗	✗	✓
Integration with Social Networks	✓	✗	✓*	✓*	✓*
Integration with Cloud Storages	✗	✓	✗	✗	✗

Table 3 - Indirect Competitors' features

Android	✓	✗	✓	✓	✓
iPhone	✓	✓	✓	✓	✓
iPad	✓	✓	✓	✓	✗
BlackBerry	✓	✗	✓	✓	✓
Windows Phone	✗	✗	✓	✓	✓

Symbian	✗	✗	✗	✓	✓
Windows	✗	✗	✓	✓	✗
Mac	✗	✓	✓	✓	✗

Table 4 - Indirect Competitors' supported platforms

It is easy to understand why these OTTs are a menace to the MNOs and rule the communications market: they provide features to the users that couldn't be found elsewhere before the existence of RCS-based applications. Users only needed the mobile operators to have a data plan in order to access these services anywhere. The texting, chatting and calling are done via these OTTs once most of them are cost free to the end user. By supporting all the major and, in some cases, minor mobile operating systems, these services have a very high rate of adoption. Although some are more complete than others, all these OTTs have a huge market share according to the amount active users.
In terms of the proposed features for the internship, there is already some offer:

- iMessage is integrated with Apple's cloud storage iCloud, allowing users to have all their messages and contacts synchronized between all their Apple devices;
- Facebook Messenger is naturally integrated with its social network. Users can seamlessly talk with their Facebook friends, share contents and view their profiles;
- Skype provides Facebook authentication, giving Facebook users the possibility to use its features without creating a Skype account. While logged in with a Facebook account, users can see the Facebook feed in Skype's main screen, as well as to invite and talk with their Facebook friends;
- Viber and WhatsApp have a poorer integration with Facebook: they only allow getting Facebook information in order to complete the user's profile of the application.

In addition to the different integration degrees of these OTTs with social networks, they all have something in common: the integration is done with Facebook. This shows that Facebook must be the first candidate social network to be integrated with WMC. However, a deeper analysis will be done in the next chapter in order to validate this conclusion.

Finally, besides iMessage, all these services lack integration with cloud storages. Most of them don't need it in order to have the messages synchronized between devices. Nevertheless, the presence of cloud storages in these services would be good so the users could access and share photos, videos and documents that are not on their mobile devices.

2.2. Social Networks

The popularity of the social networks varies from country to country. For example, a social network may be a huge success and have millions of active user in United States of America, but be considered a flop in an Asian country. Should this be seen as a problem? Usually no, it just means that the fashions are different according the world region, just like the cultures.

This section describes and analyzes the three most popular social networks in the world in order to check if they fit in the proposed internship, and to prioritize them. In order to read about the seven most used social networks, go to chapter 2 of Appendix A.

Facebook

Mark Zuckerberg, together with some of his colleagues, while studying in Harvard University, created Facemash: a web site that compiled photos from students of the Harvard campus and, placing two photos side by side, asked users about which was 'hotter'. This website was such a success that it started to include more universities and high schools. In February 4, 2004, it was launched as "Thefacebook". Nowadays, known as Facebook, this social network is a huge success worldwide. Besides being
Accessible through any web browser, it is also accessible via its official social application for iOS and Android.

Monthly active users: 1.19 billion [11].

Main Features: [32]

- "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.

App Store Rating (USA): 4/5 stars (2755659 Ratings) [33]

Google Play Store Rating: 3.8/5 stars (10462196 Ratings) [34]

Google+

Google's social network, named Google Circles while in development, was publicly release on September 20, 2011. Its launch follows the retirement of many failed Google social network platforms: Google Buzz, Google Friend Connect and Orkut (which only operates in Brazil). It is considered by many, including The New York Times [35], as Google's biggest attempt to compete against Facebook. Although it has way less users than Facebook, Google+'s growth has been notorious.
Nowadays, its official app is available for both iOS and Android mobile devices.

Monthly active users: 300 million [36].

Main Features: [37]

- "Profile": helps you control how you appear within Google+ and all across Google;
- "Circles": Easily manage whose updates you see and choose who you'd like to share

with;

- "Communities": Join public communities around shared interests, or create private communities to get together with just the right people;
- "Photos": Keep your photos safely backed up online, where they're accessible from any device;
- "Hangouts": brings conversations to life with photos, emoji, and even group video calls for free.

App Store Rating (USA): 4/5 stars (53729 Ratings) [38]

Google Play Store Rating: 4.1/5 stars (627669 Ratings) [39]

Twitter

Jack Dorsey, Evan Williams, Biz Stone and Noah Glass created in March 2006 a social network called Twitter. It is mainly characterized by the transaction of short text messages, limited to 140 characters, called "tweets". Twitter is highly adopted worldwide mainly because most of celebrities and other public figures use it, making the common citizens to join it in order to keep in touch with their everyday life moments and events. Besides being accessible using a web browser, it provides an official application for all the major mobile operating systems.

Monthly active users: 232 million [40].

Main Features:

- "Tweet": send and receive short text messages up to 140 characters;
- "Follow": subscribe to other users in order to see their tweets;
- "Retweet": share someone's tweet;
- "Hashtag": associate your tweet to a category or keyword by prefixing it with #;
- "Direct Messages": send messages and share media with users that follow you".

App Store Rating (USA): 3.5/5 stars (257477 Ratings) [41]

Google Play Store Rating: 4/5 stars (1536008 Ratings) [42]

Summary

With more than 1 billion users, it is clear that Facebook is the main and most popular social network at the moment. Together with a powerful set of APIs that allow integration with websites and iOS and Android mobile applications, this analysis confirms the conclusion made in the previous chapter: Facebook must be the first social network to be integrated with WMC.

However, Google+ is growing very quickly and getting an important position in the social networks market. Along with a very good and complete set of APIs, Google's social network was assigned the second place in WMC's integration with social networks roadmap.

Twitter is considered as a low priority social network. It is more limited in terms of features, and the provided APIs are poorer and badly documented.

2.3. Cloud Storage services

By these days, there are lots of offer in terms of free cloud storage size. Although, these services have to provide more than that to be successful and popular: security, multi-platform support and availability are some important variables in this equation.

This subtopic will do a brief description of the most popular cloud storage services that use a freemium business model, being then compared in terms of user adoption, features and API quality. Chapter 3 of Appendix A contains the full study regarding the most popular cloud storage services.

With this analysis we intend to draw conclusions on which is the most popular and used cloud storage service, in order to choose the one that must be implemented in first place.

Dropbox

This worldwide famous service was designed when one of the founders, while studying in the MIT, kept forgetting his flash drive. This way, Drew Houston and Arash Ferdowsi, with the seed funding from Y Combinator, founded Dropbox, Inc. in 2007. They developed a file hosting service called Dropbox, which offers cloud storage and file syncing through a client application. Besides being accessible using any recent web browser, it provides an application for all the major desktop and mobile operating systems.

Registered users: 200 million [43].

Main Features:

- Starting at 2GB, free accounts can go up to 16GB by inviting friends;
- Premium solutions for single users (Dropbox Pro) and for enterprises (Dropbox Enterprise);
- Files always available from the secure Dropbox website;
- Only transfers the changed parts of a file while synchronizing;
- Share viewable photo galleries;
- Uses Secure Sockets Layer (SSL) and AES-256 encryption;
- Keeps one-month history, allowing changes to be undone and files to be undeleted;
- Public API that allows the integration with third-party apps.

App Store Rating (USA): 5/5 stars (42040 Ratings) [44]

Google Play Store Rating: 4.6/5 stars (411667 Ratings) [45]

Google Drive

On April 24, 2012, Google released its own file storage and syncing service, providing cloud storage, file sharing and document editing by multiple users simultaneously. This last feature is possible by the presence of their office productivity suite, Google Docs, allowing the creation and editing of text documents, spreadsheets and presentations. The Google Drive syncing client is available for Windows, Mac OS, iOS and Android.

Monthly Active Users: 120 million [46].

Main Features:

- Free accounts have 15GB of storage space;

- There is a great variety of paid plans that increase the storage limits;
- Fast search by recognizing objects in images and text in scanned documents;
- View over 30 file types in the browser, even if the program isn't installed on the computer;
- Edit documents even while there's no network connection;
- Chat with other users that are editing the same file;
- Revision History that lasts for 30days, allowing to restore previous versions of files.

App Store Rating (USA): 3/5 stars (7102 Ratings) [47]

Google Play Store Rating: 4.4/5 stars (187911 Ratings) [48]

SkyDrive

Skydrive is the file sharing service from Microsoft. It was released in a beta version in 2007, with the name of Windows Live Folders, being available at the time for a small group of people to test it. Since then it was progressively expanded to the worldwide audience, having its name changed to the one it has nowadays. If it was difficulty to use and had a bad UX in the past, it has been fully revamped with the native integration with Windows 8 and Office 2013 Suite, having lots of new features and better User Interface (UI). At the moment, the client application supports Windows and Mac OS desktop platforms, as well as Windows Phone, iOS and Android.

Registered users: 250 million [49].

Main Features:

- Work seamlessly with Microsoft Office across Windows, Mac and web;
- Edit online documents simultaneously with other people;
- File version tracking;
- Post SkyDrive content on Facebook and Twitter;
- Share files with non-SkyDrive users.

App Store Rating (USA): 3.5/5 stars (2890 Ratings) [50]

Google Play Store Rating: 4.2/5 stars (25877 Ratings) [51]

Summary

As shown in the previous overview, Dropbox and Google Drive are the most powerful cloud storage services in terms of functionalities and user experience. The high number of users that actively use them reveals the huge success of these services. The existence of powerful and documented APIs lead developers to integrate their applications with these services.
SkyDrive has become an impressive player in the last years as well, having more registered user than Dropbox and GoogleDrive. Although, as claimed in [52], this official value is kind of suspicious because it raises the question "How many of these users are active?". Nowadays, Microsoft provides many ways for users to sign up for SkyDrive accounts (e.g., the first sign in in Office 365 suite), but most of those accounts are not actively used.
The conclusion taken from this analysis is that the main focus must be Dropbox due to its popularity, powerful features and great APIs. Google Drive is considered as the second implementation option due to be less popular than Dropbox but still having a high quantity of active users. SkyDrive is low priority due to be way less actively used and having a poorer API.

2.4. Ephemeral Messaging applications

The society highly depends on the communication process to interact within itself. Phone calls, business meetings, and chats over coffee and meals are some of the situations we communicate on a daily basis. On any of these situations, people say what they think; they are honest and genuine because they are having off-the-record conversations. What about conversations we have online, such as SMS and Internet chats? All these conversations heavily rely on web servers, most of which store the conversation records. The storage of these digital footprints is translated in loss of privacy.

In order to solve this issue and allow the users to say digitally what they're willing to say in person, the concept of ephemeral messages arose and has been well embraced by the modern digital community. Online off-the-record conversations are now possible because of the messages are exchanged with end-to-end encryption and they are not permanently stored in servers. Furthermore, these messages have associated a short lifespan, after which they eliminated from the devices of both sender and receiver.

This subtopic presents a brief analysis of the two most popular ephemeral messaging applications. It is important to understand how these services work and what are their features in order to implement a feature that can compete with them. Chapter 4 of Appendix A contains the full study regarding the most popular cloud storage services.

Confide

In 2014, John Brod, a former AOL executive, and Howard Lerman, CEO of Yext, developed and released Confide. Their ambition was to create a free ephemeral messaging application that could be used by professional business users. In other words, "What LinkedIn is to Facebook, Confide aspires to be to Snapchat — a grown-up, professional alternative for business users" [53]. One hundred days after its release for iOS, they launched an Android client application in order to make Confide a cross-platform messenger and, consequently, increase its amount of users.

User: N/A.

Main Features: [54]

- "Go off the record with self-destructing messages": Messages you send are private and encrypted, and they disappear forever once they're read;
- "Screenshot protection": Swiping prevents screenshots;
- "Works with any email or phone number": Instantly send messages to any email address or phone number;
- "Locations": Share where you are and find friends nearby;
- "Read receipts": Read receipts tell you when your message is read.

App Store Rating (USA): 4/5 stars (96 Ratings) [55]

Google Play Store Rating: 3.8/5 stars (2818 Ratings) [56]

Snapchat

In September 2011, Evan Spiegel and Robert Murphy, two Stanford students at the time, developed a photo and video messaging application called Snapchat. There were already many applications and social networks where people could share their moments, but this one had a different concept: those moments, named Snaps, were only visible for a limited amount of time. "Snapchat isn't about capturing the traditional Kodak moment." It is "a photo app that doesn't conform to unrealistic notions of beauty or perfection but rather creates a space to be funny and honest (…) at the moment you take and share a Snap" [57].
This concept is a huge success and that is confirmed by the amount of messages transacted every day: 400 million [58]. This success bothered the big players, such as Facebook who developed the Poke application in order to "kill" Snapchat [59]. Although, that was a failed attempt once users hated it. Following, Facebook and Google tried to buy Snapchat for 3 and 4 billion dollars respectively, which were both refused [60]. At the moment, its application is only officially available for iOS and Android.

Monthly active users: 30 million [58].

Main Features:

- Send image and video snaps visible up to 10 seconds;
- Add short text descriptions to your snaps;
- "Snapchat Stories": snap that is visible to all your friends for 24 hours;
- "Replay": view one of your incoming snaps again, at time of your choosing;
- "Image filters": customize your photos applying the available filters.

App Store Rating (USA): 3/5 stars (44334 Ratings) [61]

Google Play Store Rating: 4/5 stars (303814 Ratings) [62]

2.5. Real-time Location Share applications

One of the characteristics of today's society is its mobile lifestyle. The applications that share location in real-time appeared in order to ease the process of locating the ones we are meeting. There is no need to ask "Where are you?" via call or text, and try to provide directions anymore. These applications reveal a user's position in a dynamic map that updates his whereabouts in real-time.

The current subtopic approaches the two applications that provide this kind of feature at the moment. As in the previous section, the study of these applications has a big impact on the way the Real-time Location Share feature will be implemented on the WIT's product.

Find My Friends

Apple released its proprietary real-time location share application on October 12, 2011, which is available, for free, for any Apple mobile device with iOS5 or later. Using the device's built-in GPS chip, the location is obtained and shared to whom the user desires, either permanently or temporarily.

Users: N/A. Although, any of the more than 700 million [63] iOS devices sold may use this application.

Main Features: [64]

- Easily locate friends and family;
- Temporary sharing option;
- Location-based notifications;
- Simple privacy controls;
- Parental restrictions;

App Store Rating (USA): 3/5 stars (7245 Ratings) [64]

Glympse

In 2008, two former Microsoft employees that worked together for more than fifteen years grouped up and founded Glympse. Besides sharing the location of a user in real-time, they managed to develop a concept different than any other in the market: there is no need to create an account to exchange Glympses, and they can be sent to any contact in the user's contact list. The receiver will receive the invite, via a notification if he has the application, or via SMS if he doesn't. This last feature, together with such reliable results, made possible for Glympse to become the leader in location-sharing. Glympse is available for iOS, Android, and Windows Phone users in order to share the location. Although, the receiver can see the location shared in any web browser, besides the official applications.

Monthly active users: 10 million [65].

Main Features: [66]

- "Easy": no sign-up required or friend lists to manage;
- "Safe": your Glympse automatically expires
- "Live": share location in real-time
- "Open": share with anyone – no app required to view
- "Passive": runs in the background
- "Global": works anywhere you have GPS and a data connection
- "Connected": with support for Ford AppLink, BMW Apps and Mini Connected systems

App Store Rating (USA): 4/5 stars (4079 Ratings) [66]

Google Play Store Rating: 4.5/5 stars (45445 Ratings) [67]

3. Approach

As stated earlier, this project aims at the enrichment of WIT's joyn application by implementing various features: integration with social networks and cloud storages, ephemeral messaging, and real-time location share. An approach was followed in order to accomplish this goal, which consists on a development methodology that is used during the whole project, namely in the planning phase. As in any project, there are risks that may affect its outcome. This way, it is important to identify them and create mitigation plans in order to minimize or eliminate them.

This chapter presents the approach followed in the internship in four sections. The first section identifies and describes the product development methodology used in the internship. The second section presents the planning process and explains all its stages and concepts. The third describes all the process followed in order to learn the required programming languages used in the product. Finally, the last one presents the risks identified during the internship, as well as some mitigation plans.

It is recommended the reading of *Appendix B – Approach* in order to complement the information provided in this chapter.

3.1. Methodology

The current section provides an overview on the methodology used for the product development, Scrum.

What is Scrum?

According to Figure 4, there are three well-defined kinds of projects, in terms of complexity:

- Simple: projects with low complexity, where there is 100% certainty about the requirements and technologies for the solution;
- Complicated: projects with low-medium complexity. There is still some good degree of certainty regarding the solution's requirements and technologies to be used;
- Anarchy: projects where there is no consensus about the necessary requirements and a complete uncertainty about the technologies to be used. With all that uncertainty, they are hardly named projects.

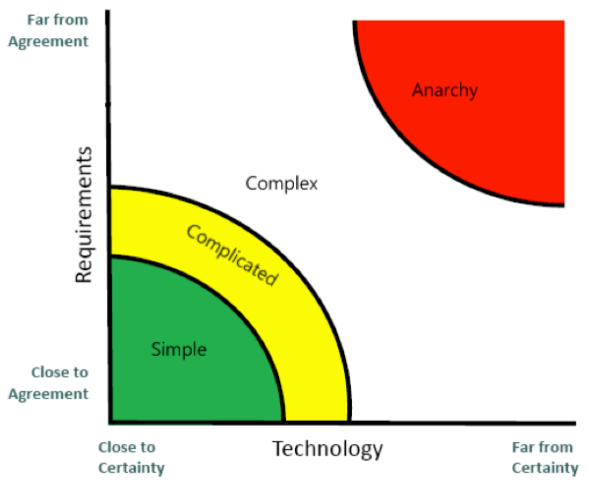


Figure 4 - Projects Complexity

Besides these three areas illustrated in Figure 4, there is a big empty space that corresponds to the complex projects. This kind of projects occurs when the intended solution has a great number of different components involved, there are many possible requirements that demand a very thorough study in order to get them minimally defined, and there is a huge variety of complex technologies that may be used. In short, these projects are feasible but, due to their complexity, they do not allow a fixed planning with all the requirements and technologies well defined. They require flexibility along their development though.

Scrum makes the difference in projects of the last type. "Scrum is a framework for developing and sustaining complex products. A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value" [68]. Among other things, Scrum provides the necessary agility for projects like the one this internship is focused on: projects, directed for products of big dimensions (like WMC), where the features can have a wide spectrum of possible requirements, and the technologies that are used are very complex.

Scrum roles

One component that is essential to Scrum's success is the work in teams. Although this internship is intended to be a one-man job, it correlates with another internship that goes in the very same area. While this internship focuses on the iOS application, the other one focus on the Android application. Both of them target the very same goal: to have both applications with the same features and appearance. This way, besides working for different operating systems, Paulo Santos, another MSc student, and me are working together as a team on the project planning, requirements definition, architecture, and UI designs.

The following roles are implemented in this project, each serving a specific purpose:

- Product Owner: "person responsible for maximizing the value of the product and the work of the Development Team" [68]. He is responsible for ordering the items in the Product Backlog, by prioritizing them in the best way to achieve the goals, and ensuring that the Scrum Team understands all the necessary Product Backlog items. The product owner can have the Scrum Team do his work. However, he still has responsibility;
- Scrum Team: set of professionals of different areas who do the work of delivering a potentially releasable increment of the product at the end of each Sprint. The Design, Development and Testing teams compose the Scrum Team.
- Scrum Master: supports the Scrum Team by eliminating any impediments to their progress, facilitating Scrum events whenever necessary, and easing the communication with the Product Owner. At the same time, he evaluates the Scrum Team's performance.

Supervisor Frederico Lopes has the Product Owner role, being responsible for the prioritization of the requirements. Since he considers important for the trainees to learn properly the Scrum methodology, he delegates some of its work to the Scrum Team, namely the definition of the requirements.

Paulo and I constitute the Scrum Team. Although we are developing for different platforms, the project is the same. This way, most of the tasks are done side-by-side in order to keep an equitable progress on both platforms. It is important to notice that we are the Scrum Team and not only the Development Team once we do more than developing: designing the UIs and testing the developed product are some of the other tasks that we are responsible for.

At the same, these two trainees also play the Scrum Master role in this project. One example of our activity as Scrum Master is regularly create meetings to check the progress of the project, and solve impediments. However, only one person is responsible for this role in a pure Scrum approach.

3.2. Planning

The project planning on Scrum, like in any other development methodology, starts with the definition of the requirements. However, a requirement in this methodology has different name and motivations.

3.2.1. User stories

A user story (Figure 5) is one or more sentences, in the everyday language of the end user, that describes the 'who', 'what', and 'why' of a requirement in a simple and concise way. It is a quick way of handling the product requirements without creating formal requirement documents or performing administrative maintenance tasks. In short, the goal of a user story is "to be able to respond faster and with less overhead to rapidly changing real-world requirements" [69], without wasting precious time in the creation or maintenance of the typical formal requirement documents.

WHO	**WHAT**	**WHY**
As a user	I want to select a received movie	in order to upload it to my Dropbox account

Figure 5 - Components of a user story

Typically, the user stories are created and managed by the Product Owner. Yet, for the reasons explained in the previous section, and together with the importance of learning this methodology, the Scrum Team does that task in this project.

It is usual to assign a user story to a certain category, mainly if there are many different features to implement in the product. This eases both development and management entities once it is easier to find the user stories regarding a certain feature. For the categories used in the project, refer to Chapter 1 of Appendix B.

3.2.2. Product Backlog

Each of the created user stories is an entry of an ordered list of everything that would be great to be in the product: the Product Backlog. This is the single source of requirements for the product and is never complete, once it is updated as the product and the environment in which it will be used evolves. In other words, its dynamism makes it constantly changing in order to identify the product needs at a specific moment. The Product Owner has an important role here: he prioritizes the user stories in the Product Backlog according to the product needs. He can also insert new user stories, for example, in case the product must incorporate another feature.

Table 5 contains a sample of the Product Backlog of this project. For the full Product Backlog, refer to Chapter 3 of Appendix.

Category	Sub-category	User Story
Prototyping	Facebook	As a developer I want to see the profile information of a contact through his number or email
Prototyping	Facebook	As a developer I want to merge my native contacts with my Facebook friends

Category	Sub-category	User Story
Documentation	Sharing Mockup	As a developer I want to create a mockup of the sharing screen in order to create a sharing UI proposal
Documentation	Timeline Mockup	As a developer I want to create a mockup of the timeline screen in order to create a timeline UI proposal
Authentication	Dropbox	As a user I want to authenticate with my Dropbox account
Cloud Storage	Dropbox	As a user I want to access my Dropbox file listing in order to select a file to be sent

Table 5 - Partial Product Backlog of the project

3.2.3. Sprint

After the project is minimally defined, it is time for the Scrum Team to work on the product. This will be done in Sprints, which are short duration milestones that allow the team to develop a small portion of the project and produce a demonstrable product increment, the deliverable. In this project, the defined sprint duration is 2 weeks because it is a period of time that allows a visible increment on the project, as well as regular sprint meetings in order to evaluate the work done and reflect about what must be improved.

The work to be performed in a Sprint is listed in the Sprint Backlog. The Sprint Backlog is the set of Product Backlog items selected for the Sprint. This way, it is a forecast by the Scrum Team about what functionalities should be done in the end of the Sprint, in order to present a deliverable. A user story is only considered "Done" when it adheres to the team's Definition of Done (DoD), meaning that all the team members recognize it as finished. For details about the definition of "Done" followed in this project, refer to Chapter 2 of Appendix B.

Chapter 4 of Appendix B contains an analysis of each completed sprint, providing a deeper context of what has been done during the first semester.

3.2.3. Burndown Chart

It is extremely important to monitor the progress of the sprint. There is a useful tool for that: the Burndown chart. The Burndown chart is the number one reason for Scrum's popularity, and one of the best project visibility tools to ensure a project is progressing smoothly. This chart proves a day-by-day measure of the amount of work that remains in a given sprint.

Using the Burndown chart, the team can quickly calculate their Burndown Velocity: average rate of productivity for each day. Knowing that, it's possible to calculate an estimated completion date for the sprint based on the amount of work remaining and, consequently, know if the work is late or on time.

3.2.4. Sprint Meeting

In the last day of the Sprint, the Scrum Team and the Scrum Master gather in order to discuss some aspects:

- Sprint Review: the Scrum Teams shows what has been "Done" in the current Sprint, and what is the status of the stories that are not finished. The provided Sprint deliverable is evaluated in order to provide feedback to the team;

- Sprint Retrospective: together with the team, the Scrum Master analyzes the Burndown Chart in order to verify how the Sprint went. This way he can identify the major positive items and the potential improvements.
- Sprint Planning: it defines which user stories are going on the next Sprint Backlog. It is possible that they are re-prioritized in order to reflect the current product needs.

As Scrum is a flexile agile development methodology, it's important to do this reflection frequently in order to continuously improve the Scrum team and the way they work.

Regarding to this project, both trainees and supervisor Frederico gather every Tuesday in order to do a Sprint Meeting with all the described stages.

3.3. Learning

This internship approaches various programming languages that were unexplored to me. In first place, as can be seen by the name of the internship, the core programming language that will be used is Objective-C, language that must be used in order to use the public APIs provided within Apple's framework. Secondly, it is necessary to acquire knowledge of C++ in order to develop parts of some features that will "live" inside WIT's communications stack – COMLib.

Objective-C
In order to properly use the language, it is necessary to understand all its fundamentals. This way, the reading of [70] and [71] were crucial for having a complete first theoretical contact with this language.

After understanding the theoretical components, I read [72] in order to understand and try practical examples where the concepts acquired previously were applied.

Acquired these basis, I felt ready to start developing for the internship. However, it is completely usual that some doubts and questions appear along the development. This way, there are two websites that have always been by my side:

- iOS Developer Library [73], which is Apple's official documentation. It is a must in order to understand how the operating system mechanisms work and how the public APIs interact with it. It is also useful in order to understand which methods can be called for each type of property, as well as if they are instance or class methods;
- Stack Overflow [74], which is a collaborative question and answer site design for the developers. In other words, it is a website where developers help each other by providing answers and suggestions regarding someone's problems.

C++
In order to understand C++ rules and structure, I relied on "The C++ Programming Language " [75], one of the books written by the creator of this language, Bjarne Stroustrup. This book provides a comprehensive coverage of all C++ language features and standard library components, from namespaces to the I/O streams.

Once *Boost C++ Libraries* are extremely used in the project, their official website [76] was also consulted many times in order to understand some of the libraries that were required.

3.4. Risks

"A project risk is an uncertain event that, if occurs, has a positive or negative effect on the prospects of achieving project objectives" [77]. It is important to anticipate such uncertain events that may affect the project in order to be able to prevent or overcome them. This can be achieved by a periodical risk analysis.

The following listing contains the risks identified during the internship, which may affect this project, as well as some mitigation plans:

Offline APIs
Once the proposed features rely on third party APIs, it is not possible to guarantee that those services will be available all the time.

- This risk cannot be eliminated.
- Mitigation plan: the application will monitor the state of the APIs. In case one of them is offline, the dependent feature will be disabled in order not to occur any kind of application crash or malfunction.

Modifications on the APIs
Since the features depend from external APIs, there is a risk of those APIs being modified/updated/deprecated without further notice.

- This risk cannot be eliminated.
- Mitigation plan: the application will analyze all the responses provided by those APIs before processing them. In case an error code is returned, the application will disable the correspondent feature. In case some API is deprecated, the feature must be disabled permanently until some alternative solution is found.

No Internet connection
An Internet connection is required to make requests to the APIs for the feature to work properly.

- This risk cannot be eliminated.
- Mitigation plan: the application periodically checks for the Internet connection status. If it is down, the application will temporarily disable the features until the connection is up again.

New RCS specifications
These proposed features are innovative and are not defined in any RCS specification at the moment. But they can be hereafter defined while the development of the features is on course.

- This risk can be eliminated.
- Mitigation plan: since WIT Software is a member of GSMA, it helps on the definition of the next specifications. This way, the features must be developed according the current suggestions of these new future RCS features, seeking for a future accreditation without any refactoring.

4. Architecture

The architecture is an essential component of any software project because it defines the structure and behavior of the different components of the system, while being a guideline for its implementation. It is important that this information is well detailed and represented in order to allow other team and company members to easily understand the system's workflow, whether they want to know its specifications or need to extend a certain module.

The current chapter is divided in the following nine sections:

- **iOS Fundamentals**: provides an overview the most important architectonical aspects of iOS that must be considered upon the architecture designing and implementation of each feature;
- **WMC Global Architecture**: gives an overview about the WIT's RCS application, namely its role within the network infrastructure, and its internal architecture;
- **Ephemeral Messaging**: provides a succinct architecture of Ephemeral Messaging feature;
- **Real-time Location Share**: presents a concise architecture about the Real-time Location Share feature;
- **Authentication**: contains a summary architecture about the authentication with both Dropbox and Facebook services within the application;
- **Save and Share from Dropbox**: provides a concise architecture about the "Save to Dropbox" and "Share from Dropbox" features;
- **Backup and Restore of conversations**: presents a succinct architecture about the "Backup and Restore of conversations" feature;
- **Contact Enrichment and Contact Timelines**: contains a summary architecture about the "Contact Enrichment" and "Contact Timelines" features.

As stated previously, this chapter only provides an overview of the features' high-level architectonical aspects. Once the project code is proprietary, all the information regarding to it, including the low-level architecture details, cannot be published. This way, it is highly recommended to read the *Appendix C – Architecture*, which provides the **complete version** of the defined architectures, as well as some details about the APIs that are used.

4.1. iOS Fundamentals

Before defining the architecture of the proposed features, it is important to study and understand various architectonical aspects of iOS. They are of great importance once they are the key to develop an application that is designed to fully operate with the system in all situations.

4.1.1. Model-View-Controller

The project's source code is written in Objective-C, with some small C++ chunks that allow the connection to the COMLib. The UI part of the project follows the MVC pattern, which assigns one of three roles to each object of the application: model, view, or controller. Figure 6 illustrates the interactions between these three components.

- **Model** – objects that encapsulate the data specific to the application and define the logic that manipulates and processes that data.
- **View** – object in an application that is seen by the user. Its main purpose is to display to the user data from the application's model objects.

- **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.

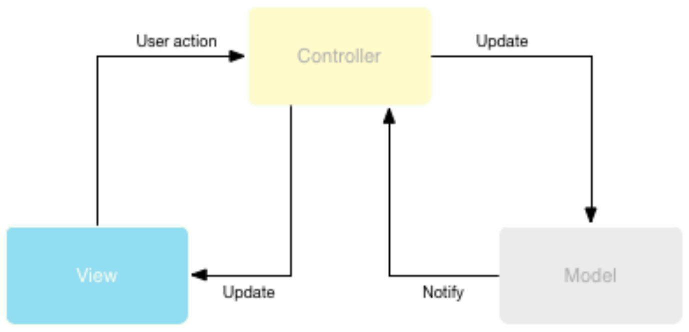


Figure 6 - MVC communication flow inside an iOS application

Using the MVC pattern in an application contributes for the reusability of its objects, while their interfaces tend to be better defined.

4.1.2. Application States

One of the extremely important while developing for iOS is the states change of an application. Since iOS devices have more limited system resources than a PC, this operating system limits the application differently according if it is in background or in foreground. These limitations improve battery life and the user's experience with the foreground application.

According to [78], there are five different application states:

- **Not running** – The application has not been launched or was running but was terminated by the system;
- **Inactive** – The application is running in the foreground but is currently not receiving events. An application usually stays in this state only briefly as it transitions to a different state;
- **Active** – The application is running in the foreground and is receiving events. This is the normal mode for foreground applications;
- **Background** – The application is in the background and executing code. Most applications enter this state briefly on their way to being suspended. However, an application that requests extra execution time may remain in this state for a period of time;
- **Suspended** – The application is in the background but is not executing code. The system moves applications to this state automatically and does not notify them before doing so. While suspended, an application remains in memory but does not execute any code.

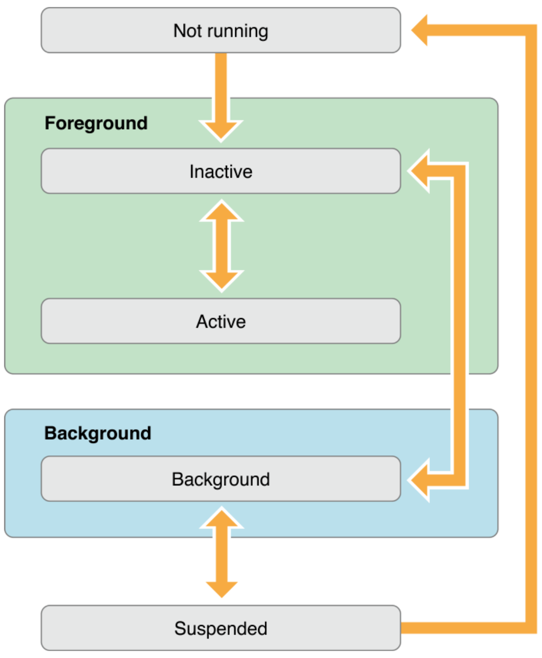


Figure 7 - State changes in an iOS application [78]

In general, these state transitions are accompanied by a corresponding method call inside the application. These methods give the opportunity to run chunks of code that respond properly to those state changes. The following list briefly describes each of the methods that may be triggered upon a state change:

- `application:willFinishLaunchingWithOptions:` This method is the application's first chance to execute code at launch time;
- `application:didFinishLaunchingWithOptions:` This method allows the developer to perform any final initialization before the application is displayed to the user;
- `applicationDidBecomeActive:` This method informs the application that it is about to become the foreground application. This should be used for any last minute preparation;
- `applicationWillResignActive:` This method informs the application that it is transitioning away from being the foreground application. This should be used to put the application into a quiescent state;
- `applicationDidEnterBackground:` This method informs the application that it is now running in the background and may be suspended at any time;
- `applicationWillEnterForeground:` This method informs the application that it is moving out of the background and back into the foreground, but that it is not yet active;
- `applicationWillTerminate:` This method informs the application that it is being terminated. This method is not called if the application is suspended.

4.1.3. Grand Central Dispatch

iOS applications are known for offering good user experience (UX). To achieve this, it is necessary to make all the API calls **asynchronous** in order to give the user a better UX. This way, an **asynchronous** request is made and when the response arrives an event is fired in order to notify the interested party. All the operations related to this request are running in secondary thread, which are called **queues**.

However, if we are talking about an **UI operation** (i.e. modifying a visible value from certain table cell, updating a contact details view), the operation has necessarily to be performed in the main thread in order to the modifications to be displayed. Dispatching a block of code to the main queue is usually done from a background queue, which signal that some background processing has finished (i.e. asynchronous API call). By convention, the main thread is responsible for updating the UI. If the UI update is not dispatched for the main thread, the UI update will never be visible to the user.

In short, all the API calls are done in a secondary thread, so that the main thread doesn't get locked and, consequently, the UI stays fluid. When a response arrives, the processing is switched to the main thread in order to update the user view.

All these dispatching operations are possible using the **Grand Central Dispatch** (GCD): Apple's proprietary technology that optimizes application support for systems with multi-core processors and other symmetric multiprocessing systems.

4.2. WMC Global Architecture

Before proceeding to the architecture design of each of the proposed features, it is important to both situate the application in its real-life scenario, and illustrate and explain all the layers that constitute this complex project. These layers will be subject of modifications and additions in order to successfully implement the features that compose this internship.

Figure 8 illustrates an IMS network from a certain Mobile Network Operator (MNO). In short, this is an architectural framework for delivering IP multimedia services. WIT Mobile Communicator for iPhone connects to this kind of networks to provide such content.

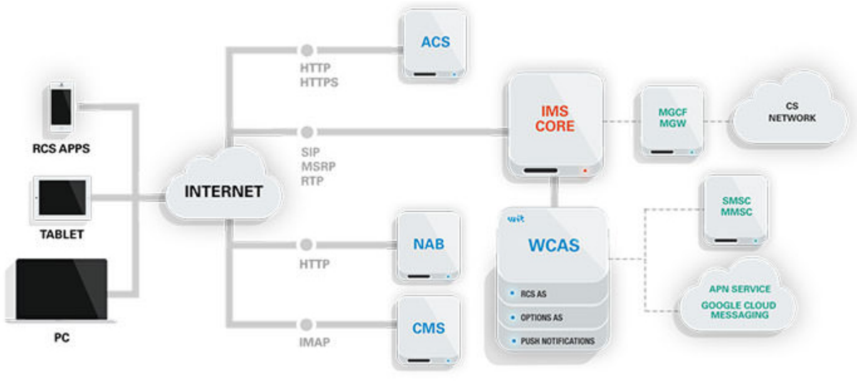


Figure 8 - WIT's RCS application in an IMS infrastructure [79]

WIT's RCS application lives upon the operating system of the mobile device. As shown in Figure 9, it is composed by two layers: the COMLib and UI/UX layers.

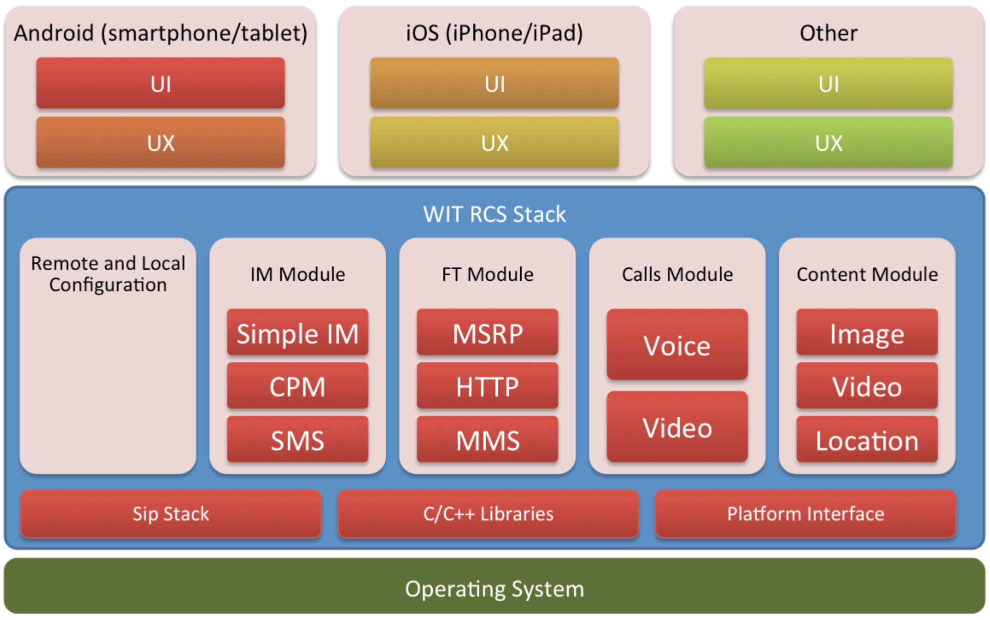


Figure 9 - WIT RCS Stack

4.2.1. COMLib layer

The application makes use of a proprietary communications library – designated COMLib – whose main purpose is to provide all the communications the application needs, such as messaging, voice and video calls, and content share. Additionally, it includes many public libraries that are required to process media or to provide secure communications.

In short, COMLib provides all the core services of the application in a Service-Oriented Architecture (SOA). This architectonical choice is justified by two main reasons:

In first place, it eases the way to enable and disable library functionalities by configuring the services that may run or not, depending on simple configuration files. Furthermore, it allows the application to keep its UI always responsive by providing services that schedule jobs to run on independent threads. This application layer is implemented in C++, meaning that one single version of the code can run on totally different platforms. For example, it can run on iOS because that system accepts any kind of C language. But it can also run on Android by using a Java Native Interface (JNI) layer that exposes the native code to the UI layer.

Secondly, this RCS application requires a strong interaction with tools that vary from platform to platform, such as the access to the Internet connection state, location sensors, or even radio signal strength. In order to keep the UI side cleaner, the access to platform utilities is executed at the very same level of the communications, and then a specific implementation is applied according the platform.

To enable the communication between the UI layer and the communication stack, COMLIb has a sub-layer – named COMLib API – that provides a set of APIs that enable the access to different modules of the application, as well as to register callbacks necessary to receive information after the completion of internal jobs.

4.2.2. UI/UX layer

This layer stands on top of the communications library, representing all the views and components visible to user, the controllers of those views, and a set of managers that execute tasks that may involve interaction with the COMLib.

The high-level package diagram illustrated in Figure 10 provides an overview of the internal architecture of the UI/UX layer. For the sake of simplicity, it only displays the packages that are most relevant for the internship, which will be modified in order to implement the proposed features.

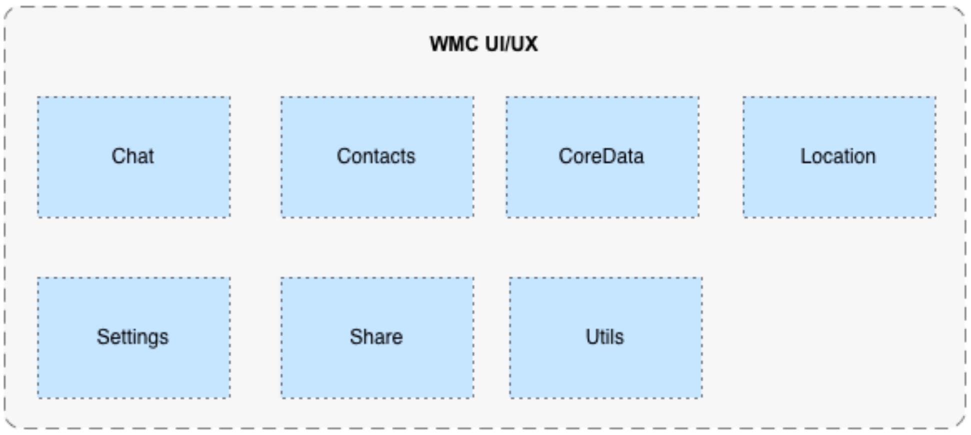


Figure 10 - UI/UX high-level package diagram

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 – they are singletons – in order to be accessed by "normal" controllers via their shared instance. This way it avoids 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** module contains all the views regarding the chat windows and chat list. These views are commanded by controllers, which will contact with the manager in situations like sending a message or a file transfer, and even deleting a chat entry. All these operations are directed to the manager that will interact with the COMLib in order to start the respective communication process.

The **Contacts** module is responsible for all the management of the contacts, which come from both native and network address books. The models retain the contact information, controllers command the views to display the necessary information from the models, and the managers are responsible for any operation regarding contacts.

The **CoreData** module provides generalized and automated solutions to common tasks associated with object life-cycle and object graph management, including persistence. A database model, a set of models that represent the objects that can be manipulated and persisted, and managers that implement all the logic necessary for the retrieval and persistence of those objects in the database compose this module.

The **Location** module is responsible for all the tasks regarding location services. By implementing the CLLocationManagerDelegate protocol, the manager is able to access the location services, such as start updating the device's location, and set the desired GPS accuracy. The controllers process the places and location models in order to tell the map views what data and when it should be displayed.

The **Settings** module is composed by a set of models that contain information about each setting of the application. The controllers will check the configuration of the application in order to tell the views which setting options shall be displayed by the views. When there's a modification of a setting value (e.g, turning off auto-accept of file transfer), the manager is responsible for the persistence of the new user preference for the specific setting.

The **Share** module refers to all the accesses to external files, such as Gallery photos, and their respective preview inside the application.

Finally, the **Utils** module contains a set of public libraries and frameworks that are required for some features to work on the UI/UX layer. One of these libraries is SSZipArchive, which is a utility class for zipping and unzipping files on iOS and Mac.

Provided this overview about the RCS application, it is now time to proceed to the next chapters, which will approach the architecture for each of the features this internship is focused.

4.3. Ephemeral Messaging

The ephemeral messaging feature was introduced in the intership's scope on the end of the first semester. This feature requires only implementation on the UI/UX layer of the application.

This feature consists of the transaction of messages and file transfers that have an associated lifetime. After the expiration of its lifetime, the corresponding message will be deleted automatically.

Figure 11 illustrates the communication process used by the ephemeral messaging feature.

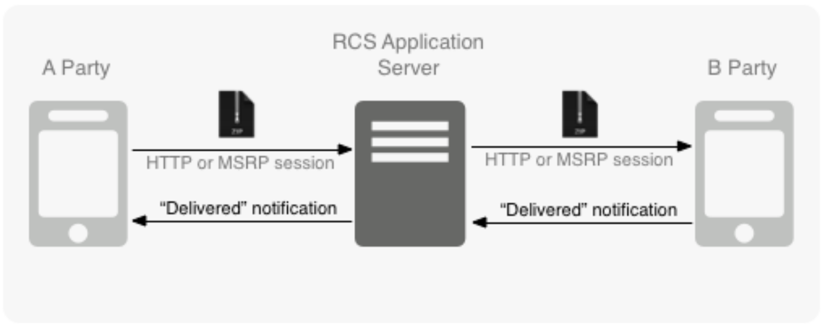


Figure 11 - Ephemeral Messaging communication flow

In order to send and ephemeral message (Figure 12), the A Party enables the "Ephemeral Messaging Mode" and defines a short lifetime for them. After setting up, the sender is now ready to select any media file: image, video, or audio asset. The selected media file, together with a file that contains the asset's lifetime, is then compressed in a .zip file in order to be sent to the B Party. This exchange of media is done via a typical file transfer via HTTP or

MSRP, depending on the technology that is available in both parties. When the B Party receives the file transfer, the file is unzipped and, consequently, a new generic balloon (that doesn't reveal if the received file transfer is an image, video or audio) is displayed on the respective chat windows. Afterwards, it still replies with a "Delivered" notification so the A Party knows it has been successfully delivered and, consequently, deletes both the respective media and entry from the device.

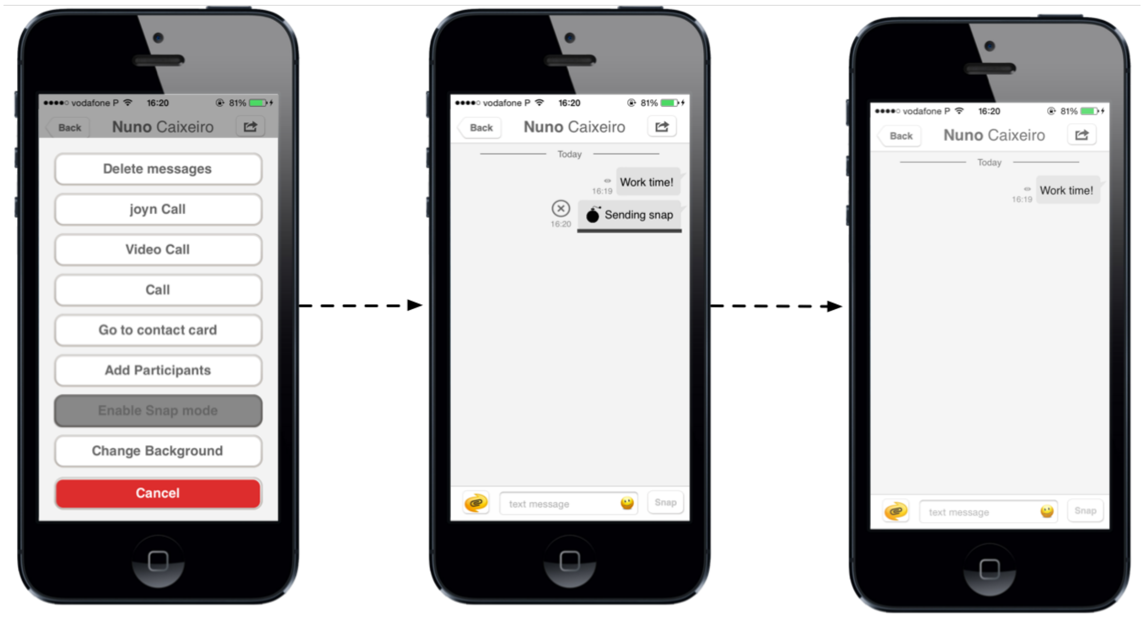


Figure 12 - Send Ephemeral Message: enable mode, send content, and automatically erase records

In order for the B Party to preview the content of the file (Figure 13), it is necessary to press the respective generic balloon. This single tap gesture triggers a mechanism responsible for the preview of the resource, which makes a whole new screen appear that shows the received media. This also starts a countdown timer that decreases after each second. After it reaches zero, meaning that the media lifetime has expired, the B Party is automatically redirected to the chat windows, and that entry, together with the respective media file, are erased from the device.

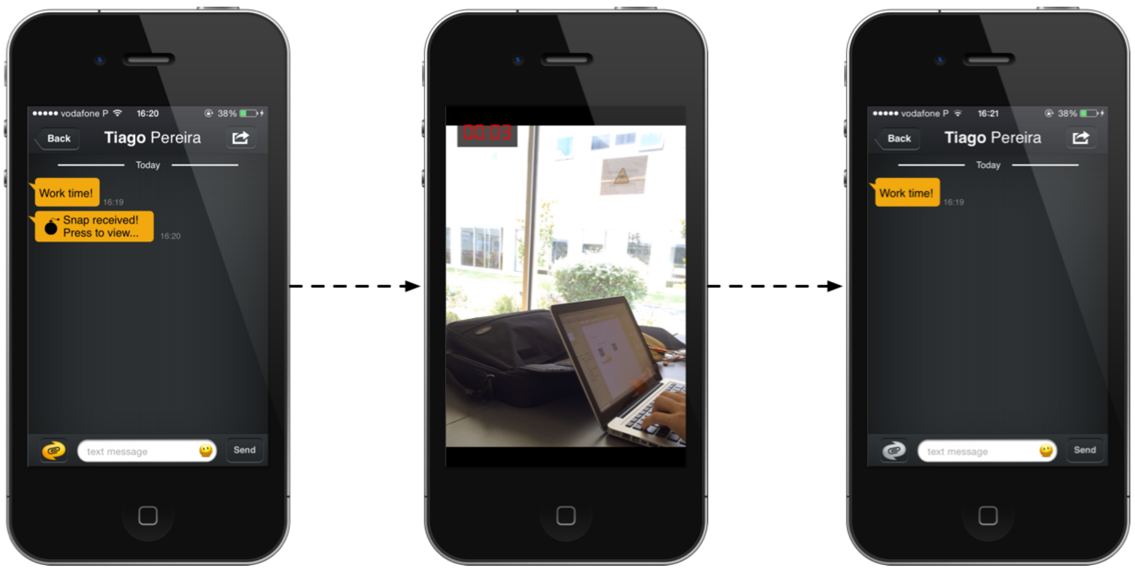


Figure 13 - Receive Ephemeral Message: press balloon, preview content, and automatically erase the record

Figure 14 shows a high level architecture of this feature. For a detailed diagram that shows and explains all the involved classes, as well as all the additions and modifications, refer to *Appendix C – Architecture: Ephemeral Messaging*.

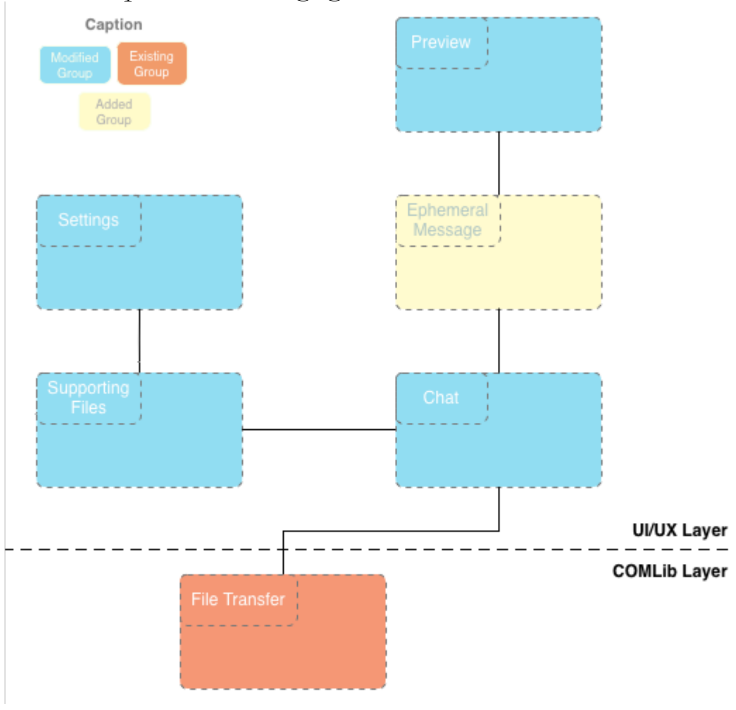


Figure 14 - Ephemeral messaging high-level architecture

A new settings section was added for the user to define the lifetime of each ephemeral message. In order to do that, a new identifier was created and added to the XML file used to set the active settings. Furthermore, this identifier was assigned to a **list cell** model for the user to be able to select one of the three lifetimes available. Each time the user defines the duration of the ephemeral messages sent, that new value will be persisted inside a supporting file.

The Chat module was modified for the user to be able to enable/disable the Ephemeral Mode within the chat window; for the chat window to display a new balloon type that represents an ephemeral message; and for the application to send a compressed zip file with both the file and its lifetime.

The added Ephemeral Message group is responsible for the new cell types that are displayed in the chat window.

Finally, the Preview group was modified in order for the preview screens to display a countdown timer that shows the remaining lifetime of the received ephemeral message.

4.4. Real-time Location Share

As the name says, this features allows the sharing of A Party's location with a B Party, in real-time. This location is shown on a map that keeps track of the most recent position of the sender, presenting as well all the previous positions received in that session which constitutes the shared route.

Figure 15 illustrates the communication process used by the ephemeral messaging feature. For the sake of simplicity, this feature will be referred by its acronym: RTL (Real-time Location).

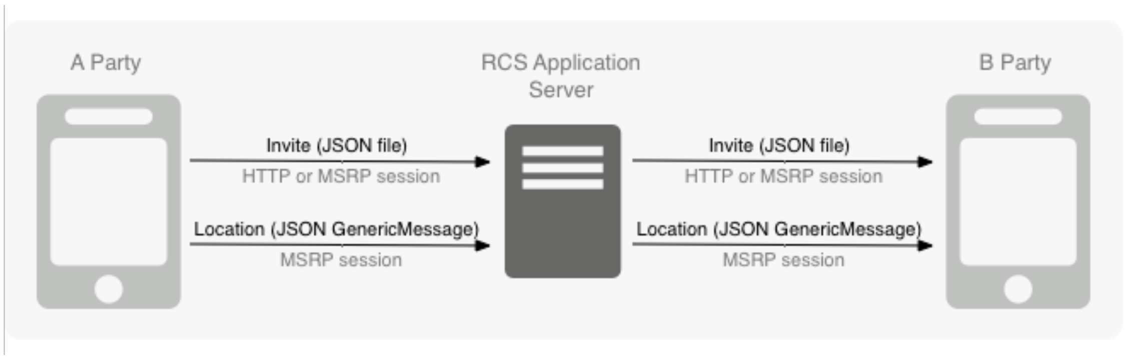
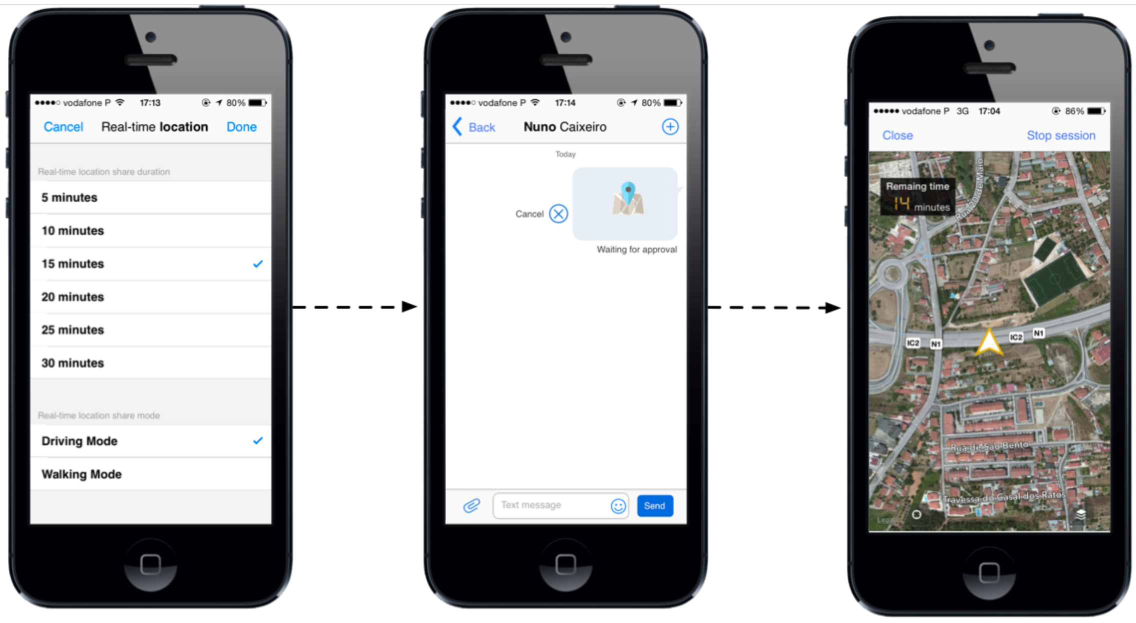


Figure 15 - Real-time Location Share communication flow

In order to share real-time location (Figure 16), the A Party selects "Share location" and then "Real-time location". This entry point will lead to a modal view that asks the user to select one of the available durations for the share session, as well as its mode (Walking or Driving). As will be seen later, the session mode influences how the routes are drawn in the map. After setting up these preferences, the sender may now send the invite to the B-party. This invite consists of a JSON file with the timestamp of the invite, the user's latitude and longitude, and the session state (which for default is IDLE). This file will be sent to the B-party using file transfer over HTTP or MSRP, depending on the available technology. This file is of huge importance for the persistence of current state and date of the session. For example, the JSON file will be updated every time a new position is received or the RTL session state changes.


Cancel Real-time location Done

Real-time location share duration

5 minutes	
10 minutes	
15 minutes	✓
20 minutes	
25 minutes	
30 minutes	

Real-time location share mode

Driving Mode	✓
Walking Mode	

Figure 16 - Send Real-time Location: define settings, wait for a response to the invite, and share location

When the invite is received, the session is established and all the communications will be done via Generic Message: text messages, just like chat messages, but that have different media type in order not to be interpreted as usual chat messages. These messages are a way of powering this feature and are not intended to be directly displayed in the chat window.

At this point, the B-party must answer the invite (Figure 17): either accepts it and the sender starts sharing its locations periodically, or rejects and the session is terminated. In order to explain the remaining flow, lets assume the B-party accepts the invite. This acceptation will be translated in the sending of a generic message to the A-party saying that the invite has been accepted. Upon the reception, the A-party will start sending its location periodically via generic message, which will be stored on both sender and receiver in order to be displayed on the map.

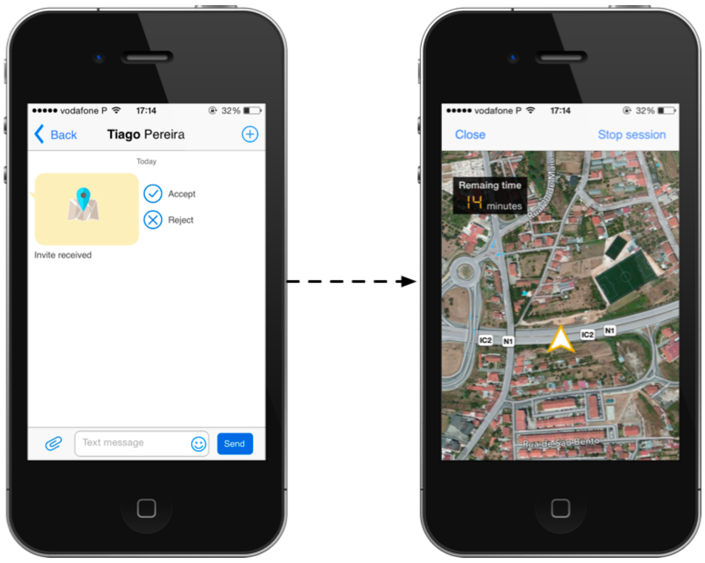


Figure 17 - Receive Real-time Location: receive invite, accept the real-time location session, and receive the shared location

The map view is the way the user can see the sharer's current position and route. In order to display the route, the logic behind the map's controller will read all the stored points and draw a line over the map. However, that line's aspect will vary according the session mode:

- Driving mode: as the name suggests, this mode is intended for sharing location while driving on roads. For the routes to precisely match the roads in the map, requests with the received points must be made to the Apple's Map API in order to receive the corresponding map positions.
- Walking mode: this mode is intended as a free-style and off-road mode. In other words, there is no need for the routes to exactly match the roads in the map. It is suitable for walks in the woods or traveling inside tiny roads within a city, for example. This way, the locations coordinates are used to draw a line directly in the map, without requesting any directions to external APIs. However, this process is not straightforward. In order to present smooth lines in the map, Bezier curves [80] are calculated and then displayed in the map.

Every time a new location arrives, the route line will be updated in order to present the most up-to-date state of the share session, as shown in Figure 18. This process will terminate when the session is terminated by one of the users, or its duration expires.

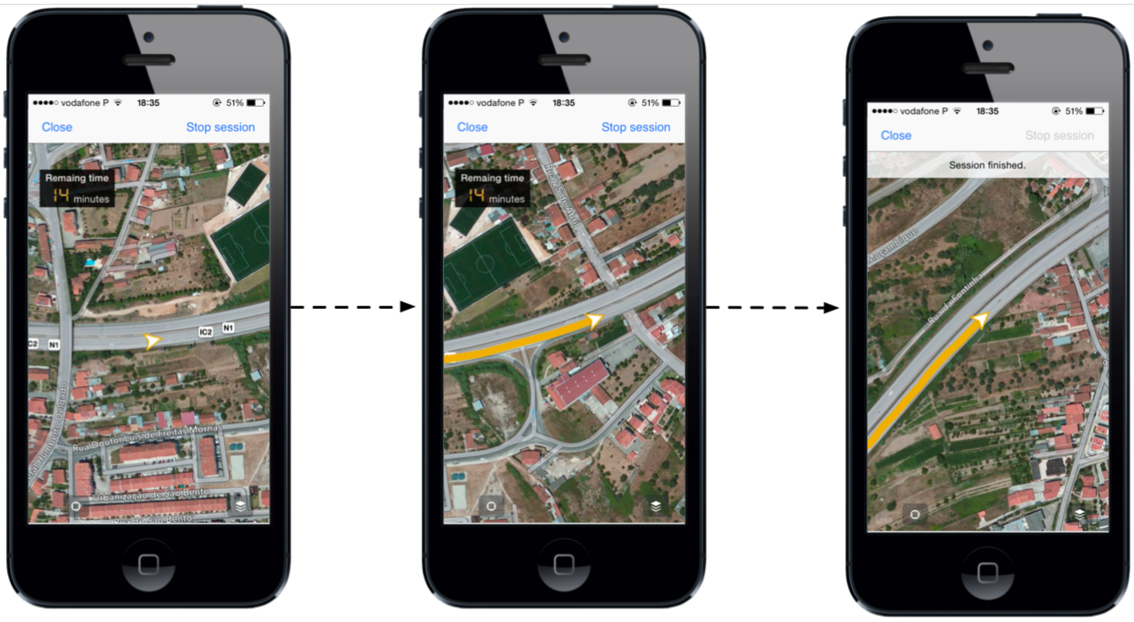


Figure 18 - Evolution of the shared route until finishing the session

Figure 19 shows a high level architecture of this feature. For a detailed diagram that shows and explains all the involved classes, as well as all the additions and modifications, refer to *Appendix C – Architecture: Real-time Location Share.*

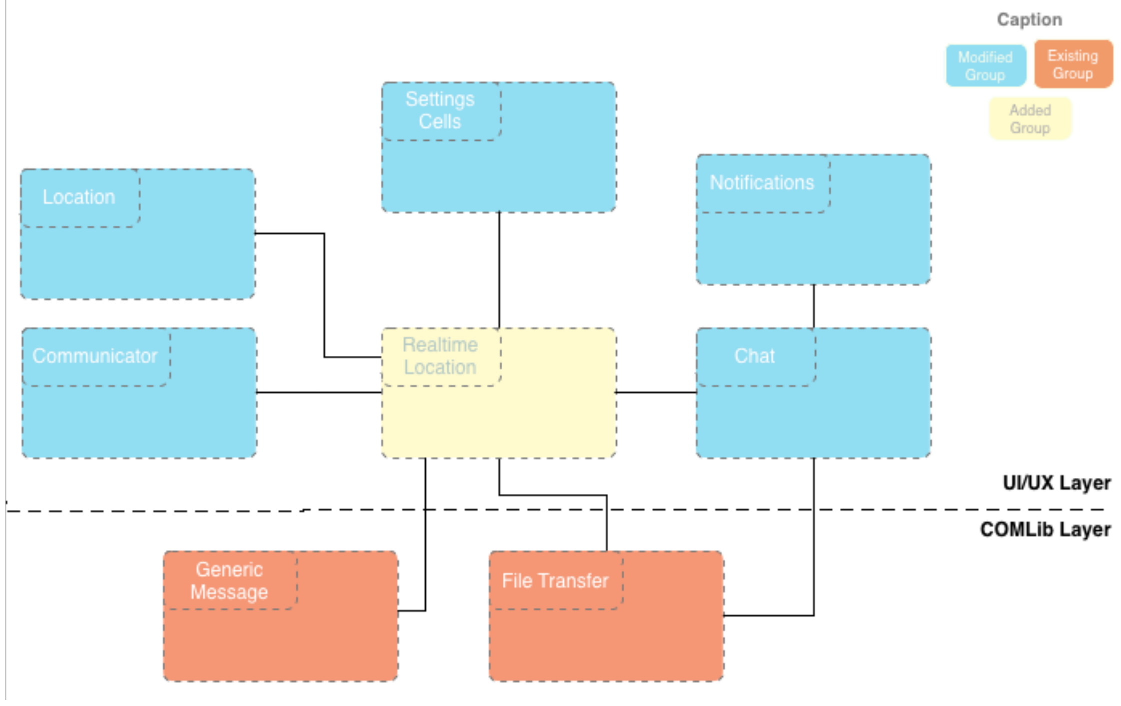


Figure 19 - Real-time Location Share high-level architecture

In order to implement this feature is necessary to create the Real-time Location group, which contains all the logic that makes it work, such as creating the session invites and send them via the COMLib's File Transfer module; sending the locations periodically via the Generic Message module; or even to process all the shared points according to the session mode and present the resulting route on a map. However, many groups of the UI layer must be modified as well.

The Communicator group must initialize the Real-time Location module upon the initialization of the RCS application.

Besides fetching more accurate positions, the Location module must include the necessary logic to empower the Real-time Location group.

The Chat module must now present new conversation balloons that are specific for RTL sessions, and trigger specific notifications that warn about the reception of new RTL invites.

Finally, the Setting Cells group must suffer some modifications in order for some of its models and views to be reused by the Real-time Location module, such as the RTL invite setup menu.

4.5. Authentication

In order to deploy features that depend on third-party services, it is necessary to implement the respective authentication mechanisms, which will grant access to the respective user's account. Regarding the proposed features, it is necessary to implement authentication mechanism 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 current chapter will merge both services in one only 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 20. Once social networks and cloud storages are different kind of services, two different menus were created in order to wrap them accordingly.

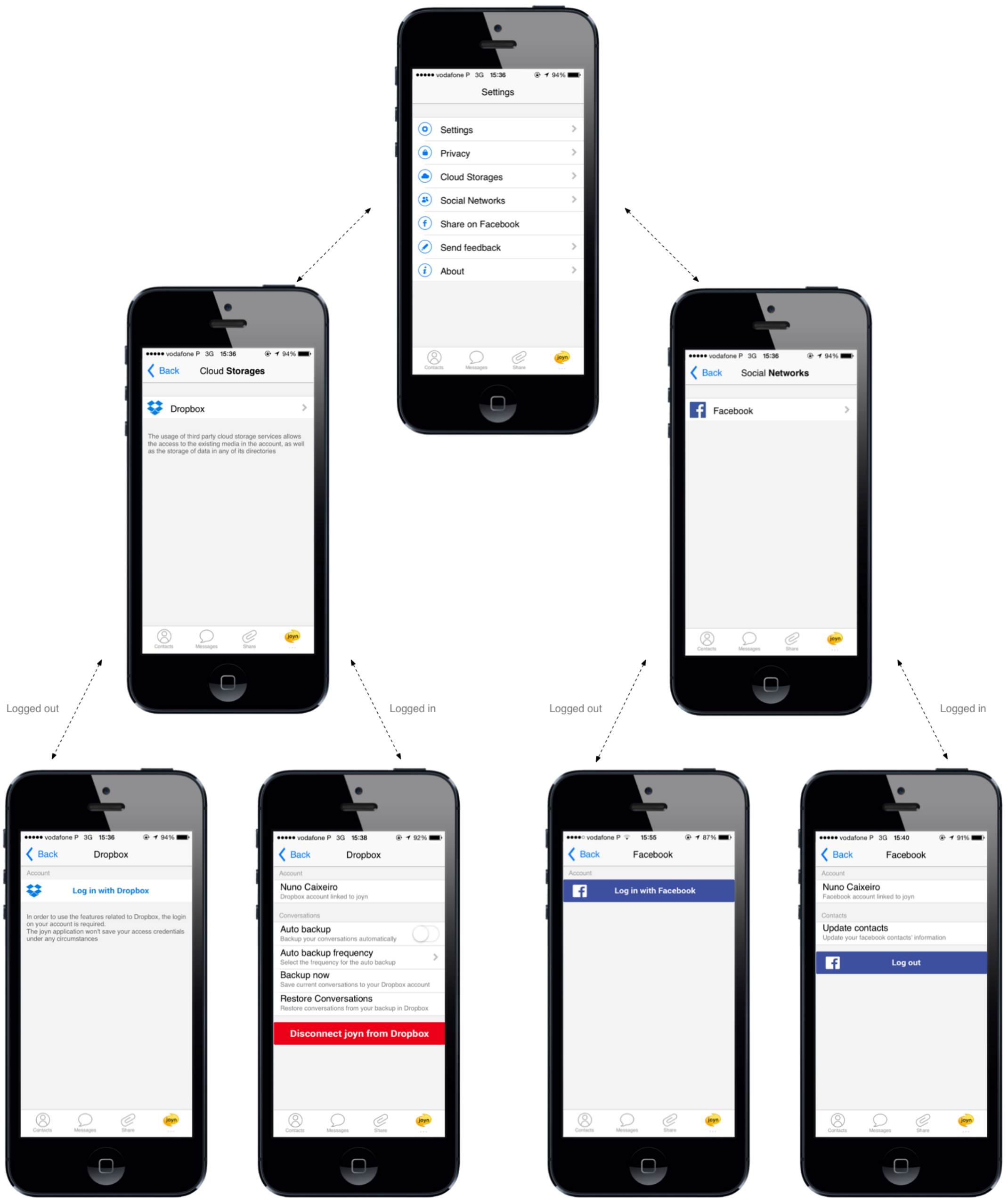

Settings
Settings
Privacy
Cloud Storages
Social Networks
Share on Facebook
Send feedback
About
Back
Cloud Storages
Dropbox
The usage of third party cloud storage services allows the access to the existing media in the account, as well as the storage of data in any of its directories
Back
Social Networks
Facebook
Logged out
Logged in
Logged out
Logged in
Back
Dropbox
Account
Log in with Dropbox
In order to use the features related to Dropbox, the login on your account is required.
The joyn application won't save your access credentials under any circumstances
Back
Dropbox
Account
Nuno Caixeiro
Dropbox account linked to joyn
Conversations
Auto backup
Backup your conversations automatically
Auto backup frequency
Select the frequency for the auto backup
Backup now
Save current conversations to your Dropbox account
Restore Conversations
Restore conversations from your backup in Dropbox
Disconnect joyn from Dropbox
Back
Facebook
Account
Log in with Facebook
Back
Facebook
Account
Nuno Caixeiro
Facebook account linked to joyn
Contacts
Update contacts
Update your facebook contacts' information
Log out
Contacts
Messages
Share
joyn

Figure 20 - Authentication flow for both Dropbox and Facebook

The application settings are populated with base on 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 behavior. The application already has set of setting models that have its specific behavior and design.

Figure 21 represents a high-level architecture for the authentication with both these services. For a detailed diagram that shows and explains all the involved classes, as well as all the additions and modifications, refer to *Appendix C – Architecture: Authentication*.

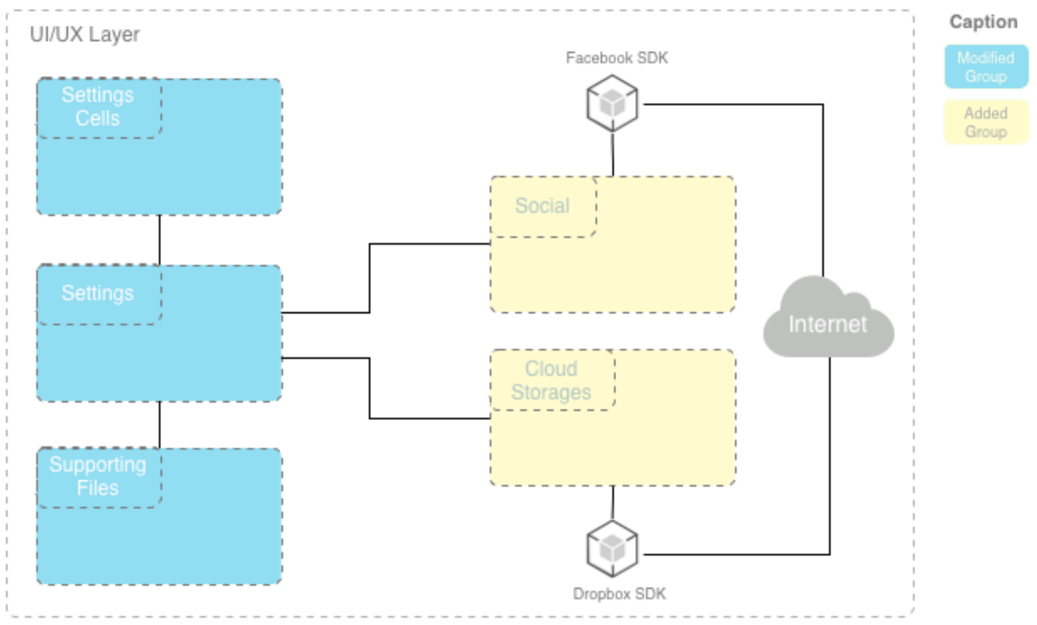


Figure 21 - Dropbox and Facebook authentication high-level architecture

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.

4.6. Save and Share from Dropbox

The current topic is responsible for defining the architecture for the features "Save to Dropbox" and "Share from Dropbox". Both these features rely on the previously defined authentication architecture, once they will only be available if the user is logged in to its Dropbox account.

Both these features have many similarities in terms of classes, and they only rely on Dropbox Core API. Their architecture will then be designed together in order not to overflow this topic and to provide their shared context. However, each of the flows is explained separately in order to fully understand each of the features.

Save to Dropbox

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

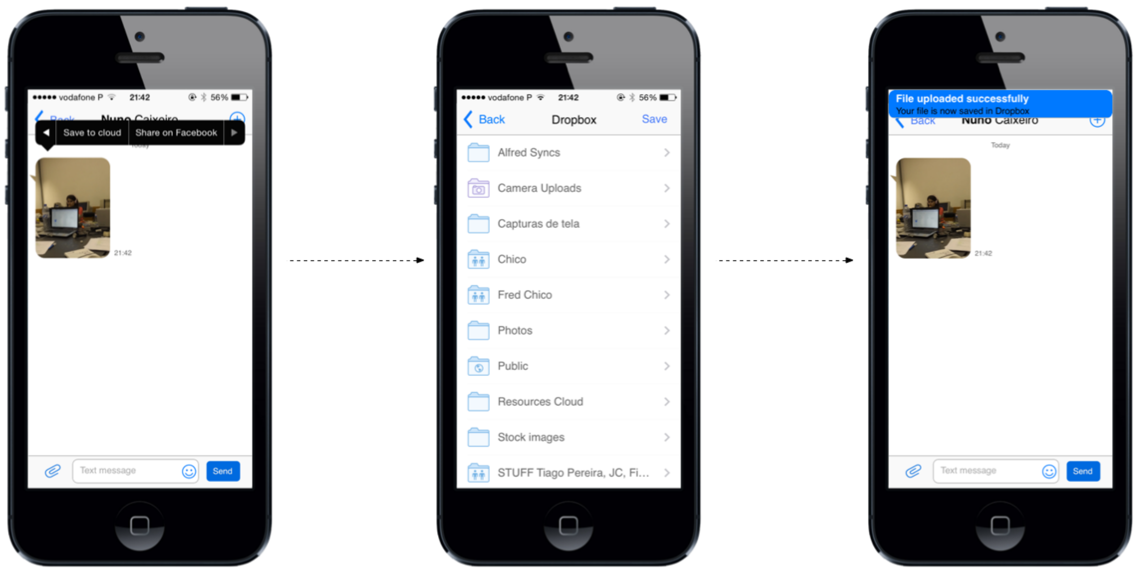


Figure 22 - Save to Dropbox flow

Figure 22 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 for 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 he can see a progress bar below the balloon he selected. After the upload is completed successfully, a local notification will be 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 to be saved to their cloud account.

Share from Dropbox

A user can share a file from their Dropbox account via two entry points within the application: a chat window or the share menu within the application's main screen. Figure 23 displays both these entry points.

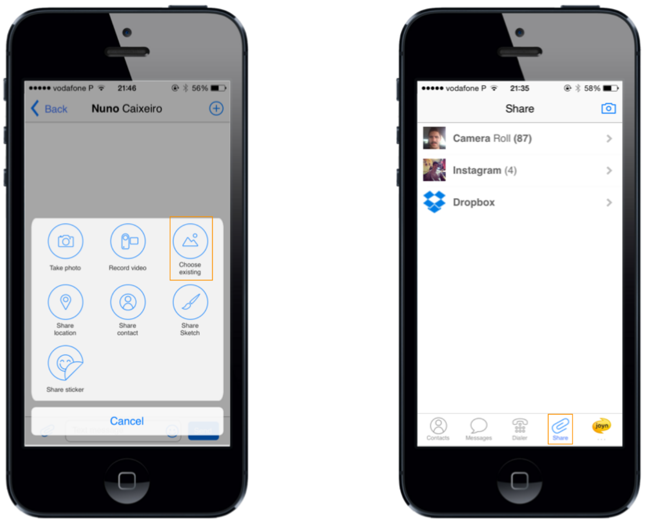


Figure 23 - Share from Dropbox entry points

Both the available entry point will start the feature's flow illustrated in Figure 24. A list with all the media sources pops up, which includes the clouds the user is logged within the application. Selecting the Dropbox entry will once again present him a browser where he can navigate through his account's hierarchy. However, unlike the previous feature, the browser displays all the available content, both files and 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.

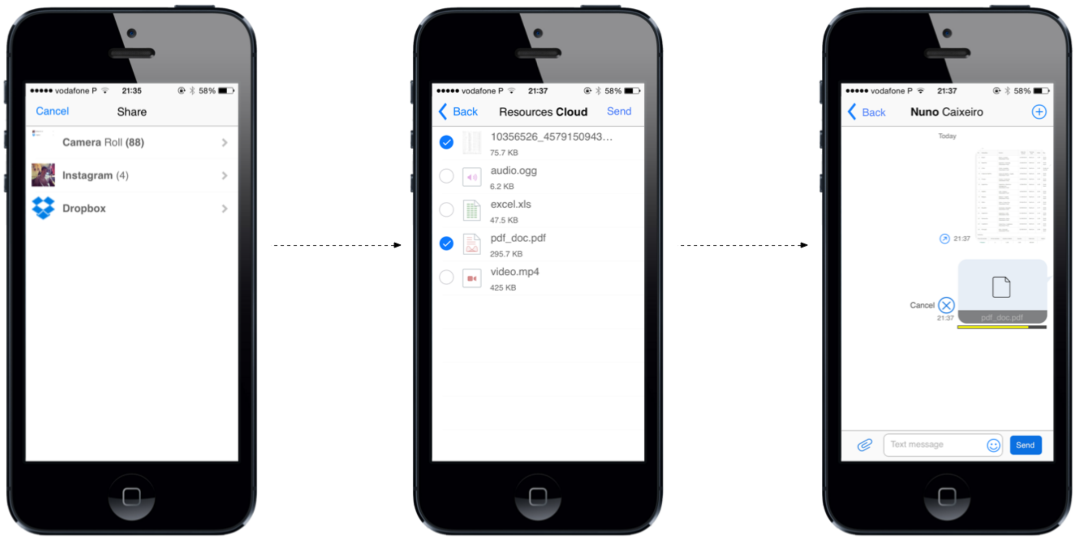


Figure 24 - Share from Dropbox flow

Figure 25 represents a high-level architecture for this feature. For a detailed diagram that shows and explains all the involved classes, as well as all the additions and modifications, refer to *Appendix C – Architecture: Save and Share from Dropbox.*

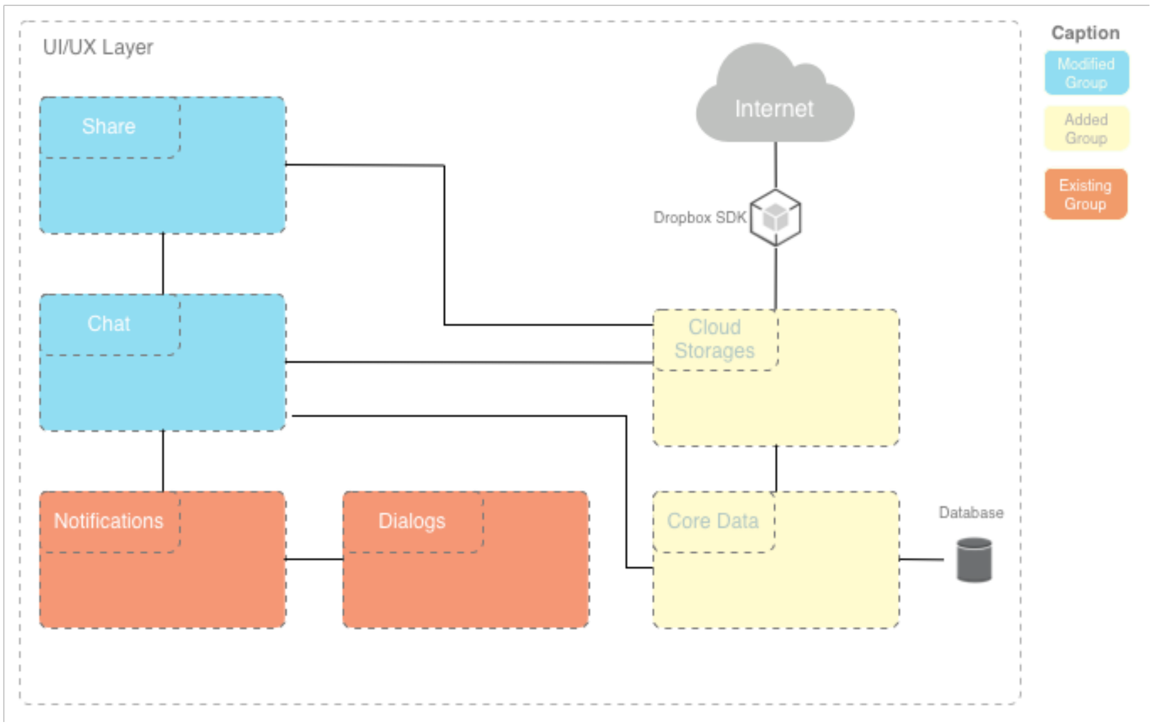


Figure 25 - Save and Share to/from Dropbox high-level architecture

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 services.

The Share group contains all the logic regarding the listing of the media sources upon a share, such as native gallery albums. This way, it had to be modified in order to include the cloud storages that have a valid session within the application.

The Cloud Storages group is responsible for all the methods that interact with the Dropbox SDK in order to empower both features.

- Regarding the "Save to Dropbox" feature, it contains all the methods responsible for the file uploading, including one to fetch the current upload progress in order to be displayed to the user.
- In terms of the "Share from Dropbox", it contains the methods responsible for obtaining the public links. Public links are downloaded instead of the files in order to reduce the amount of mobile data plan consumed during the File Transfer process. This way, the sender won't need to download the file to the device, and then upload it in order to be sent to the B-party.
- Common to both features are all the methods regarding the metadata fetching and session validity control. This is crucial for the file explorer mechanism since it provides the hierarchy of each folder, as well as information of each of its files.

The Core Data group allows the caching of the files thumbnails used in the file explorer, and the persistence of the upload progress of each file in order to be displayed in the corresponding balloon.

Finally, each time the upload process finishes, whether with success or not, a notification will be displayed to the user informing him of the final status of the upload.

4.7. Backup and Restore of conversations

The "Backup and Restore of conversations" feature intends to allow a user to do a safe backup of all its conversations within the RCS application to Dropbox, as well as to restore any stored conversation back to the application.

This section approaches the "Backup and Restore of conversations" feature and is divided in two subsections: the first approaches the data models that will be used in order to store the information of the messages, while the second is responsible for briefly talking about the modifications and additions that must be done on the COMLib level.

The COMLib is the foundation of this feature, a layer where most of the implementation must take place. Once this communications stack is the most important component of WIT's RCS product, it is extremely important not to give details about its internal structure in a public document. This way, this topic will be very vague regarding the architecture of this feature. Thus, it is highly recommended to read *Appendix C – Architecture: Backup and Restore of conversations*, where everything regarding the COMLib is deeply explained, approaching in a very detailed way both the architectures before and after the implementation of this feature.

4.7.1. Data models

The backup and restore of conversations is achieved by recurring to the Dropbox Datastore API. This is possible by maintaining a local cache that contains the changes performed by the application in the datastore. When the application wants to *commit* these local changes to the remote datastores, it performs a sync call. Fortunately, the Dropbox Datastore SDK for iOS handles that sync process internally, without any possible interference by the developer.

However, there are some issues that may arise when saving the RCS application messages in the datastores. These problems are related with the limits applied by Dropbox.

1) If there are more than 10 MB of raw data in these messages, it won't be possible to save all in the same datastore. Fortunately, there is not a limitation on the maximum number of datastores that can be created by an application.
   <u>Solution</u>: When one datastore is full, create another one and save the remaining messages there.

2) The maximum record size allowed is very small to store the media files from a file transfer. To address this situation, the Dropbox Sync API will be used in conjunction with the Datastore API.

The Sync API allows for an application to use a folder created under its name inside the *Apps* folder. That folder can then be accessed and modified freely by the application, and the API provides the access to a mechanism similar to the Datastore API's one that synchronizes folders across several devices: a local cache for the file system modifications is kept; when the files are synchronized, the changes are committed to the remote folder. That said, the strategy is to save the file transfer's media file inside this folder, and keeping its path inside a record that represents that history entry.

As the entries inside the RCS application are ordered by type, it makes perfect sense to keep that simplicity in the remote datastore. This way, each datastore will contain entries of a single type within a single table that is identified by the kind of records it contains: File Transfers, Locations, or Chat Messages. When a datastore is about to get full, another one of the same type is created in order to store the next entries. This concept is illustrated in Figure 26.

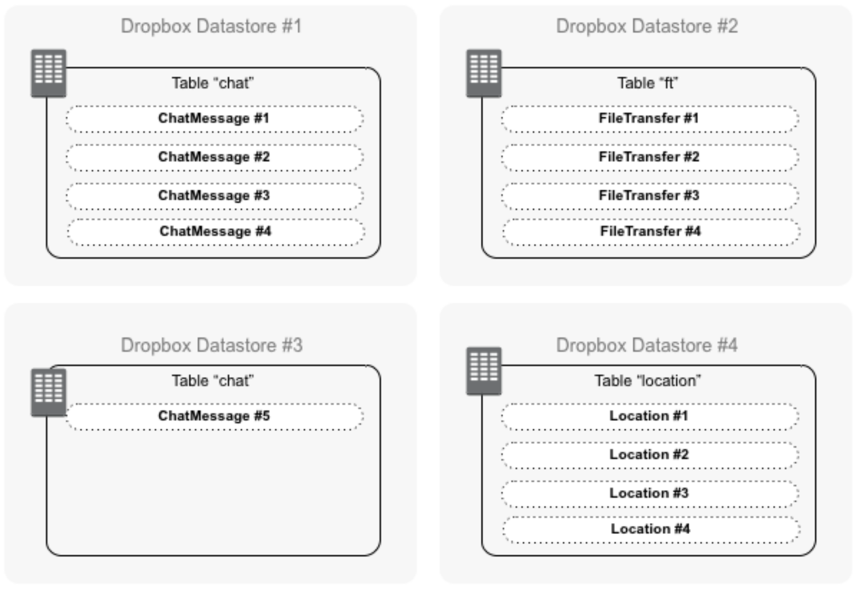


Figure 26 - Datastore structure within the RCS application

When the user's account is linked for the first time within the application, it contains no local datastores. In order to have it accessible locally, the SDK automatically fetches the remote information. However, there is a datastore that is created automatically for any application – the default datastore. In the scope of this feature, this datastore will be used in order to store relevant information about the current backup in the user's Dropbox, as illustrated in Figure 27. For example, it will allow the application to keep track of the datastores that are being used and which kind of entries they store, or the remaining size and records that any of the current datastores can still hold. In short, besides the control that it provides to the application, this information allows a performance improvement on the datastores accesses.

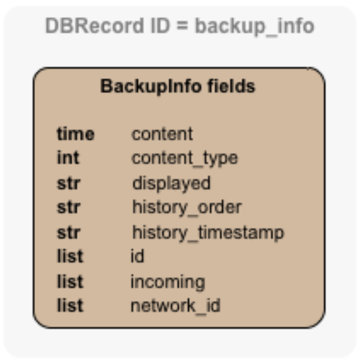


Figure 27 - Information stored by the default datastore

4.7.2. The Sync Module

After introducting all the aspects regarding the organization of the datastores and respective data, it is now time to address some architectonical aspects. As explained in the beginning of this chapter, this topic is going to be, to some extent, vague and brief about the feature's architecture. This way, it is important to once again advise the reading of Chapters 8.2 and 8.3 of the *Appendix C – Architecture: Backup and Restore of conversation.*

This feature is built on top of an already existing COMLib's module: the Sync module. This module constantly synchronizes the RCS application messages with the native messaging application, and vice-versa, in order to always keep the device's messaging history synchronized. Until the implementation of this feature, it was only useful for Android once it is the only platform that grants access to the native messaging application.

Figure 28 illustrates a high-level architecture of this feature in the COMLIb's context. Besides being mainly developed in the COMLib, there is some interaction with the UI/UX layer. This layer calls the methods in the API in order to start the backup or restore process, or even to enable or disable the automatic backup mode. It also receives event status updates via callbacks, in order to inform the layer of the conclusion on the backup process. This is useful in order to trigger the necessary mechanisms in order to display notifications to the user, which inform them of the status of each process.

Besides the APIs, the COMLib layer has many components that were implemented in order to make this feature work properly. A new service was created – the Remote Sync – in order to power this feature. This service contains multiple implementations (iOS and Android) in order to be available on different operating systems.

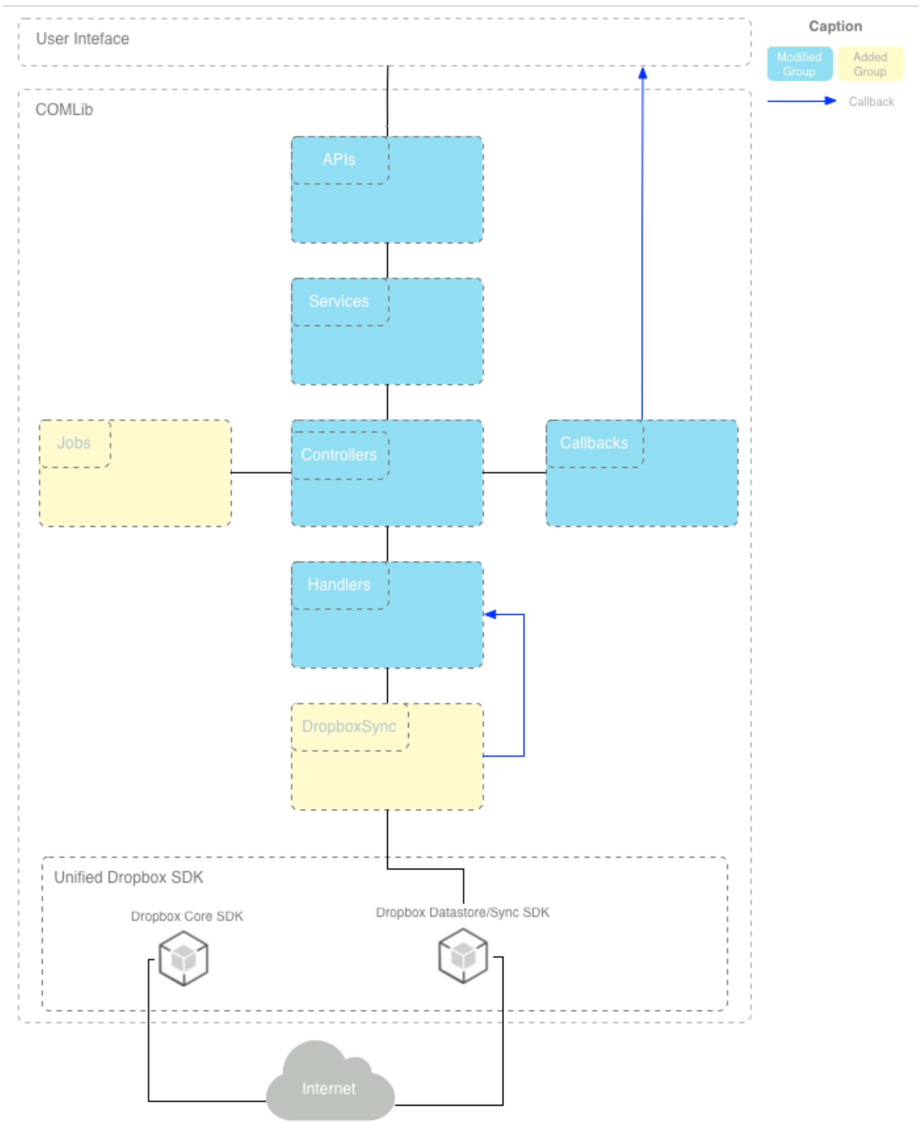


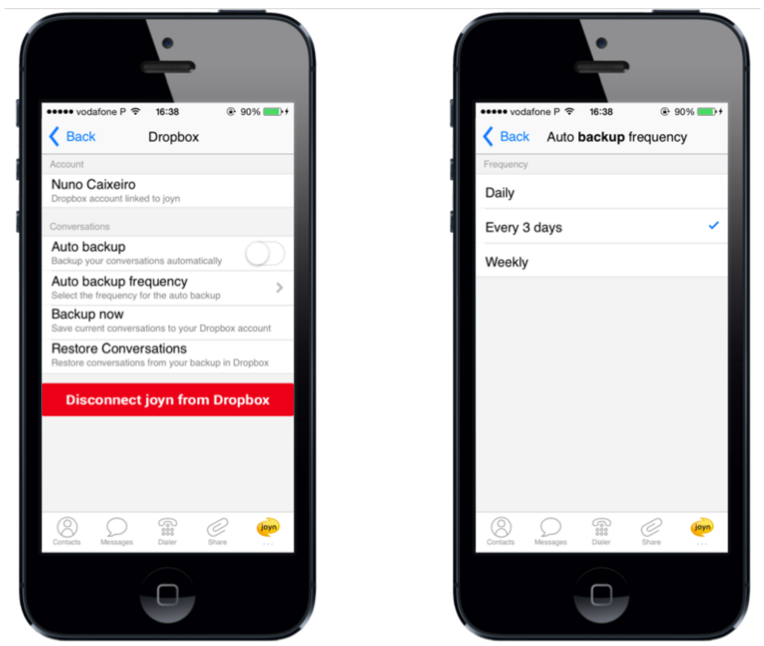
Figure 28 - Backup and Restore of conversations high-level architecture

This service is connected to the Remote Sync Controller, which processes both existing and new messages in order to be backed up. These messages are then redirected to the handlers, one for each service used for remote synchronization. At the moment, only one exists which is associated with the Dropbox cloud service. On the other hand, each handler redirects those messages for its specific logic for the corresponding platform. In this case, the DropboxSync contains specific logic for both iOS and Android platforms. For the application I developed, the iOS-specific logic will be called in order to properly process this data.

This logic is responsible for converting the received messages and adding them to the local datastores, starting the backup process, and triggering the restore mechanism that retrieves all the messages that exist remotely. In order to do this, there is a continuous interaction with the Datastore/Sync SDK.

As one can see on Figure 28, there is a module named "Unified Dropbox SDK", which contains both Core and Datastore/Sync SDKs. Because of this feature, the Core SDK had to be moved from UI/UX to the COMLib layer and join the Datastore/sync one. However, they can't be used together in the same project once they generate various conflicts, such as duplicate symbols. This way, a unified SDK had to be created in order to manage to keep using these two SDKs simultaneously in the same application. All the process discovered and implemented by me in order to generate the unified SDK is described in the following chapter of this document.

Regarding the UI layer, it uses the settings architecture presented section 4.5. Authentication for the "Save to Dropbox" and "Share from Dropbox" features. The entry points that trigger this feature are in the Dropbox settings screen, as illustrated in Figure 29. Notice that the user must be logged in in order to access those options. That screen allows the user to manually start the backup or restore process. Furthermore, they can activate the auto-backup option, which will periodically do an automatic backup of their conversations to Dropbox. In order to set the frequency of the automatic backup, the user needs to select the "Auto Backup Frequency" option so they can select one of the multiple frequencies available.


••••• vodafone P 16:38 90%

Back Dropbox

Account

Nuno Caixeiro
Dropbox account linked to joyn

Conversations

Auto backup
Backup your conversations automatically

Auto backup frequency
Select the frequency for the auto backup

Backup now
Save current conversations to your Dropbox account

Restore Conversations
Restore conversations from your backup in Dropbox

Disconnect joyn from Dropbox

••••• vodafone P 16:38 90%

Back Auto **backup** frequency

Frequency

Daily

Every 3 days ✓

Weekly

Figure 29 - Backup and Restore entry points (on the left) and Auto backup frequency options (on the right)

Besides the creation of the entry points, it is necessary to provide the feature with mechanisms that inform the user of the progress of the backup or restore task. This way, whether the backup or restore ends successfully or with an error, a local notification is always displayed to the user in order to inform them of the status of the operation. The restore process contains more than a local notification on the UI layer, as proved by Figure 30.

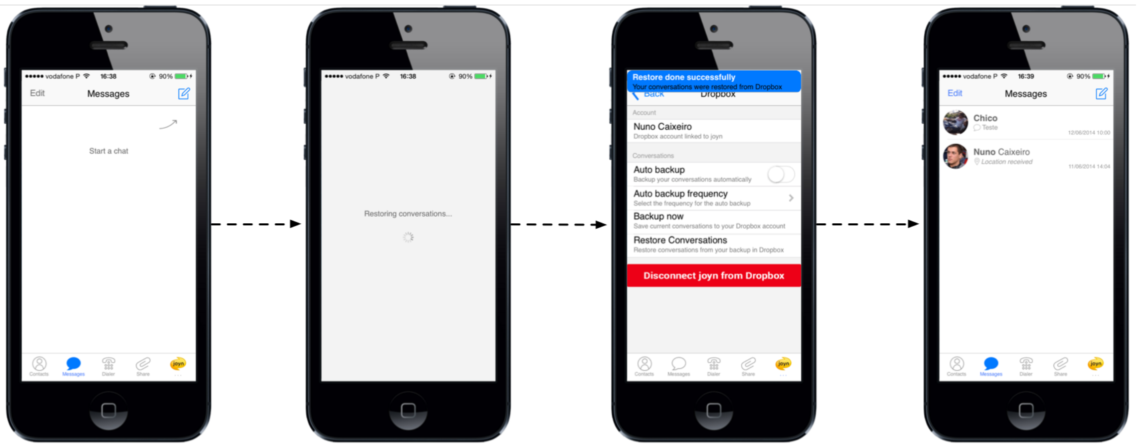


Figure 30 - Restore of conversations flow

When the user triggers the restore process by clicking on "Restore Conversations" option, a modal view appears in order not to allow them to take any action. After the process is completed, a local notification is displayed informing about the final status of the restore. After that, all the messages restored from an existing backup are in the application's database and displayed in the "Messages" tab menu.

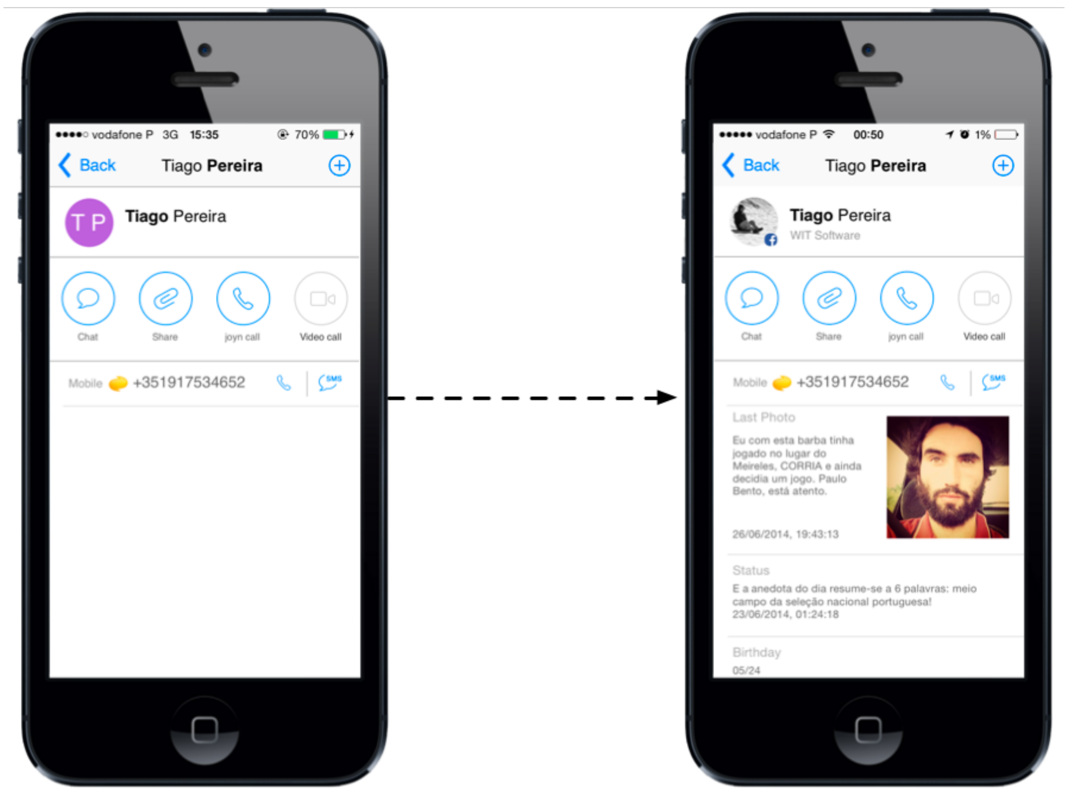
4.8. Contact Enrichment and Contact Timelines

"Contact Enrichment" and "Contact Timelines" are the features that integrate with Facebook. They have different purposes but their architecture is very similar, as they share most of the added and modified components. Therefore, this section approaches these two features together, using the very same approach used in section 4.6. Save and Share from Dropbox. However, both of the features are explained separately in order to provide an overview of their purpose, as well as their intended flow inside the RCS application.

Contact Enrichment

Most of the contacts in our mobile phone's address book contain only its name and phone number. If we send emails frequently, it is common to have some vCards with the emails on its respective field. However, all the other fields are empty because people just don't want to spend time filling them, or they do not know which is the updated information. For example, users discard fields like birthday or birthplace once they represent accessory information. On the other hand, people highly rely on social networks these days. They have many details about themselves on their profiles, like birthday date, current job, or even the city they live in.

This situation originated the perfect context for this feature within the RCS application: enrich the contacts' information by extracting the necessary information from their respective Facebook profiles. This way, it is possible to have a contact detail with many information that otherwise would not be there, as illustrated in Figure 31. Furthermore, this new information provides the necessary means for a whole new set of alerts, such as birthday notifications during a call or chat session.



Eu com esta barba tinha jogado no lugar do Meireles, CORRIA e ainda decidia um jogo. Paulo Bento, está atento.

26/06/2014, 19:43:13

E a anedota do dia resume-se a 6 palavras: meio campo da seleção nacional portuguesa!
23/06/2014, 01:24:18

05/24

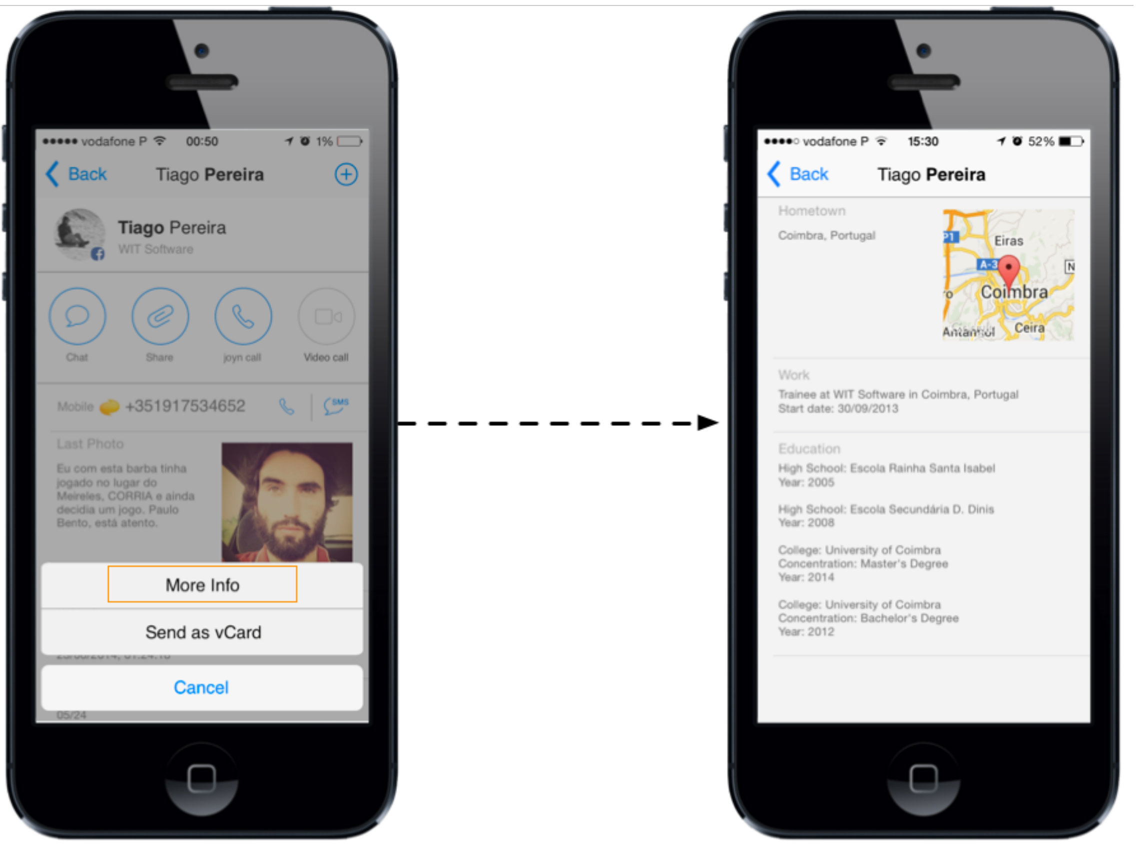
Figure 31 - Contact detail before and after the Contact Enrichment process

Two major steps compose this feature. First of all, it is necessary to extract the complete friends list of the user, and match each of them with the user's native contacts. This matching is done using phone number and/or email. After those matches have been correctly made, it is then time to gather information from the Facebook profiles that were matched and update each vCard.

Contact Timelines

The "Contact Timelines" feature works upon the previous feature once it requires the matching between Facebook profiles and native contacts in order to work. This feature provides additional information inside the contact details page in the RCS application. In other words, it is an easy and fast way to check all the updated Facebook information about a certain contact without leaving the application.

While the "Contact Enrichment" feature matches the contacts with Facebook profiles and obtains only the necessary information in order to fill or update each of vCards fields, the "Contact Timelines" provides an easy way of accessing all the contact's Facebook information within the contact detail page in the RCS application. It is useful once the user does not need to leave the application in order to see the Facebook profile information of a certain contact. Figure 32**Error! Reference source not found.**illustrates this feature and its entry point.



Tiago Pereira
WIT Software

Mobile +351917534652

Last Photo
Eu com esta barba tinha jogado no lugar do Meireles, CORRIA e ainda decidia um jogo. Paulo Bento, está atento.

Hometown
Coimbra, Portugal

Work
Trainee at WIT Software in Coimbra, Portugal
Start date: 30/09/2013

Education
High School: Escola Rainha Santa Isabel
Year: 2005

High School: Escola Secundária D. Dinis
Year: 2008

College: University of Coimbra
Concentration: Master's Degree
Year: 2014

College: University of Coimbra
Concentration: Bachelor's Degree
Year: 2012

Figure 32 - Entry point (on the left) and Contact Timeline screen (on the right)

Figure 33 represents a high-level architecture for both these features. For a detailed diagram that shows and explains all the involved classes, as well as all the additions and modifications, refer to *Appendix C – Architecture: Contact Enrichment and Contact Timelines.*

The Social group contains all the logic necessary to query Facebook in order to obtain both the user's Facebook friends and their respective information. This information is then passed to the Contacts group, where it is processed in order to find a match between the user's Facebook friends and the contacts that were loaded from the native address book. If there is a match, the necessary fields are created or modified inside that contact in order to persist that new information in the native address book.

In order for the user to see details about a certain person, the loaded contacts are passed to the Contact Detail group where all the information is displayed, including that obtained from Facebook. If the user wants to access that contact's timeline, the Contact Detail must gather updated Facebook information in order to be displayed. This way, the Social group will call the necessary APIs in order to obtain all the information. This is then redirected back to the Contact Detail group, which will then present it in a totally new view.

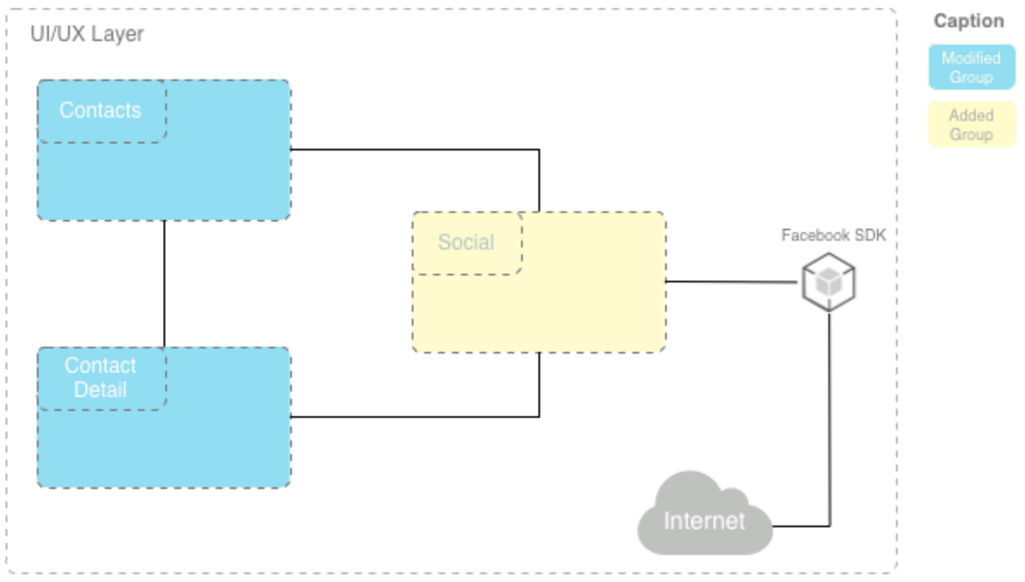


Figure 33 - Contact Enrichment and Contact Timelines high-level architecture

In order for these features to work, it is required to grant permissions for the application to get the necessary information about the Facebook users. In the login process, a set of permissions must be defined in order to obtain the necessary information about the users in a future phase. This way, the permissions required to load the Facebook friends and their information are:

- "friends_photos": permission required to get the users' published photos;
- "friends_status": permission required to get the latest status that the user ponted on Facebook;
- "friends_work_history": permission required to get the friends' work history indicating the current company, role, and time period;
- "friends_education_history": permission required to get the friends' educational path indicating the institute, concentration, and finishing year;
- "friends_birthday": permission required to get the users' birthday data;
- "basic_info": permission required to get the user's name, birthplace, and current location.

In order to obtain the previously described fields, a multi-query is used to get the information and, since the operation is **asynchronous,** it should be performed in a secondary thread (background) in order to maintain the UI fluid. When the response arrives an event is fired from the manager and the provider is notified. After retrieving the information, it is parsed into WMCContacts and inserted in a cache, avoiding constant API calls. The photo's cache mechanism is provided by a third-party library named as SDWebImage [81]. This library provides an asynchronous image downloader with cache support that allows a better abstraction to the developers.

When a user selects a contact from the contact list, its Facebook details are displayed – Contact Enrichment. This view shows the contact name in the social network and a set of basic information, including the current city, their birthday, the last posted status, and their last posted photo. These last two fields are followed by the respective publication date.

However, Facebook provides much more information than indicated above. This information can also be displayed in the contact details. If a user intends to see more information about the contact, they should press the “more options” button and select the option “more info”, in order to access that contact’s timeline. A new view is displayed with less relevant information. This includes their biography, work history, educational path, religion, political view, and relationship status.

5. Implementation Challenges and Evaluation

This chapter addresses the two stages encompasses by the development process: the implementation and validation each feature.

The first section presents some implementation details about each feature, mainly challenges and problems that were faced.

The second section contains all the validation steps performed regarding each feature, in order to evaluate the final solution of the product that integrates all the implemented features.

5.1. Challenges and Problem Solving

Some problems were faced during the development stage. Most of them were predicted during the designing of the architecture, being the features designed in a way that solved those problems. However, some other problems appeared during the development of the features. Some of them were minor, requiring only some small changes, while others were critical enough to require a reformulation of some parts of the feature.

This section explains the most important details of the implementation phase for each feature, while focusing on the main challenges that they brought to the project, along with the alternatives found in order to overcome them.

5.1.1. Ephemeral Messaging

The ephemeral messaging feature was the first feature implemented in the internship. This implementation contributed highly for the trainees acknowledgement of the internal structure of the application, as well as all the communication processes used. This feature relies on the File Transfer module of the existing communication stack. In order to implement it, various different developments were required.

In first place, a new kind of file transfer was created. This file transfer type had a new media type, which allowed the RCS application to detect if a received file transfer corresponds to an ephemeral message in order to apply the necessary correct logic.

Secondly, an ephemeral message has not the same composition as a typical file transfer, once it is not straightly composed by its corresponding media. Once an ephemeral message has an assigned lifetime, it is necessary to send that information together with the respective media. The solution found was to insert the lifetime value in a JSON formatted file, which was then wrapped together with the respective media in a zip file. After the creation of this file, the file transfer mechanism could be executed in order to send the ephemeral message to the B-Party.

One of the requirements of this feature was the deletion of the message's content from the device, as well as its corresponding entry from the chat window. In the sender's side, the content needed to be deleted as soon as the message was sent. This way, the application listened to the states of the corresponding file transfer in order to detect when it was sent to the B-party. When that happened, the communication stack was commanded to delete that message's media from the device. Furthermore, the corresponding entry was also deleted in order to provide the user with an experience of deletion, which in fact happened internally.

Finally, the last challenge was to implement a decrementing timer on the view responsible for the preview of a ephemeral content. The `NSTimer` native class had an important role here, once it provided a timer-like behavior. When it reached zero, an action could be executed. In this features case, it was intended to delete the media from the device, as well as its respective entry, and return to the chat window. This was relatively easy to do. However, there was a non-deterministic bug: sometimes the timer on the preview view was not decrementing, revealing always the same value. After the defined time passed, the view was dismissed. After reading much documentation, the reason of that problem was found: "`currentLoop`: Returns the `NSRunLoop` object for the current thread" [82]. In other words, the timer was being dispatched for one of the available threads, which isn't necessarily the main queue. When it was dispatched for a thread other than the main one, the timer action was set and triggered but the UI was never redrawn. In order to solve this problem, the `mainLoop` was used.

5.1.2. Real-time Location Share

The Real-time Location feature was introduced in the beginning of the current year and it was aiming to be ready in time for the Mobile World Congress 2014, in Barcelona. Fortunately, the developments were ready in order the demonstrate it there. However, this simple-looking feature turned out to be very complex in many aspects. The algorithm behind the calculation of routes for driving mode; protect the feature against network-loss when in driving mode; and selecting the valid points read by the GPS are some of the areas that turn out to be more time consuming than initially imagined.

For this feature, two session modes are provided: the Driving and Walking modes, which correspond to a road-based session or not, respectively. These modes have a huge impact on the way the shared routes are displayed in the map.

Regarding the Walking mode, the successive positions shared are connected using a Bezier curve, which will then be displayed in the map. The main idea behind this choice is that walks and runs are usually not on a straight line, but more likely to contain zones with curves. By calculating the intermediate points for a Bezier curve, a more realist path is created once there are no sudden changes of directions, but rather smooth curves. This concept is illustrated in Figure 34

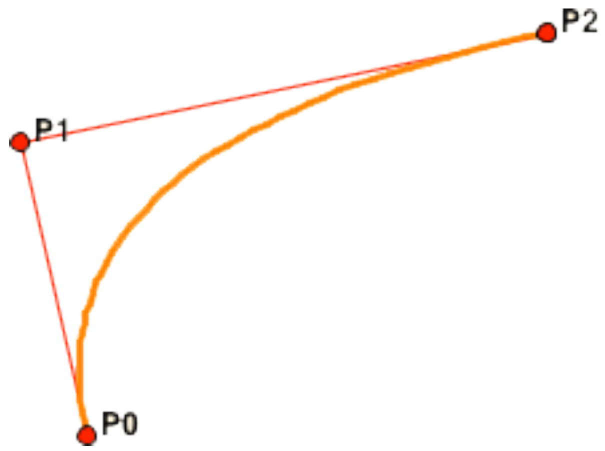


Figure 34 - Straight line (in red) vs Bezier curve (in orange)

For the Driving mode, it is intended that the route matches the existing roads. In order to guarantee that both more and less accurate points match a road, the Apple Maps API is used for requesting driving directions between two consecutive positions. This way, it managed to draw routes that exactly match with roads in the map.

Driving Mode

This mode requires a specific section once many challenges were faced in order to produce a feature that could correctly display routes that corresponded to the reality and that could be used by anyone.

First of all, the mobile data plan from the user is used in order to request directions once they are simple HTTP requests. In order to mitigate this problem, all the returned directions are saved in the corresponding JSON file. This way, when the user opens the map in order to view a previously shared route, there is no need to do duplicated requests once the necessary route information is inside the corresponding local file.

"What happens if there is a network loss when the feature is about to request a direction between two points?". That was one of the questions we faced in order to create a mechanism that controlled the network-loss in this feature. If there is no Internet connection, that request is saved in a queue along all the others in the same state. When the Internet connection is recovered, all these requests are process in a First-In First-Out (FIFO) order so that the results are obtained in the correct order and, consequently, that the route is correctly drawn in the map. In case new positions arrive when the requests queue is being processed, those points are added to the end of that queue in order to guarantee the necessary correct order. The atomic access (synchronized) to this queue makes it thread-safe and thus no crash will occur when a new request is added to the queue at the same another is being retrieved for processing.

The last problem regarding the Driving mode is related with the accuracy of the positions read by built-in GPS chip. Even a good reading from the iPhone's GPS sensor has an error radius of four to six meters. These errors have a huge influence when matching the points with roads. A slight error can point to a zone of the road that totally disrupts the calculation of the correct zone. One situation that is highly affected by these errors is when calculating routes near roundabouts, as illustrated in Figure 35.

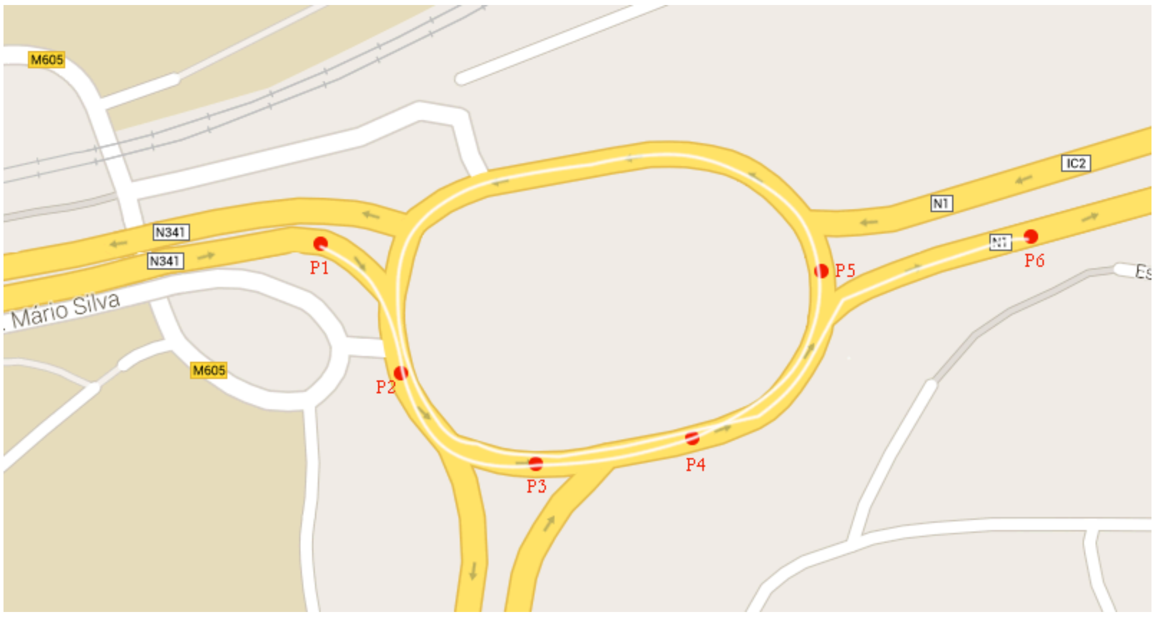


Figure 35 - Wrongly determined route caused by the P5 position

The P5 point contains a slight error as the figure shows. When requesting routes from P5 to P6, the result obtained will say to circumvent all the workaround until reaching the P6 point. However, that is not intended once it produces a route that was not traveled by the user. This way, in order to ignore such situations, the speed associated with GPS fetched point is also used. The average speed between two readings is calculated by the average speed on the start and end points. By multiplying that average speed for the time elapse between readings, a rough estimate of distance traveled by the user is calculated. In addition, a small factor is added to the obtained value in order to establish the maximum length that a route obtained from the Maps API can have. This way, when an obtained route is too long, it is ignored, as well as the last obtained position. When the next location update arrives, the same process is repeated in order to guarantee a precise and correct route.

5.1.3. Save and Share from Dropbox

Unlike the Android platform, the iOS operating system does not allow any communication between third-party applications. In other words, the RCS application cannot invoke the Dropbox application in order to, for example, display its file explorer, select the intended files, generate the respective public links, and finally receive them so they can be sent to another contact. This way, the most challenging part of both these features was the creation of a generic file explorer that allowed browsing through the Dropbox folder hierarchy, just like in the Dropbox application. I say generic because it must work with any cloud storage other than Dropbox, in order to be reused if other integration of that kind takes place.

Both these features rely on the File Explorer for the user to browse the cloud's folder hierarchy and select the intended media. 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 display the files and folders from Dropbox, a HTTP request is made for the metadata from a certain directory. The response contains all the information regarding 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 regarding the feature:

- <u>Share from Dropbox</u>: the user selects the files he wants to send. After he is done, each of the selected files are sent using COMLib's File Transfer API. Downloading a file from Dropbox and then uploading the file in order to be sent to the B-party represents a relatively high usage of mobile data. However, this process was not good enough and was optimized in order to reduce the amount of data exchanged. Instead of downloading the files from Dropbox, the public links of each of them was retrieved and then sent to the user. In order to provide a great UX, the necessary logic was created to display to both sender and receiver a balloon with the file's thumbnail instead of a text with the link.
- <u>Save to Dropbox</u>: the user browses the folder they want to save the file to. When they are done, that directory name is retrieved and a method from Dropbox Core SDK is called in order to upload the intended file to Dropbox. While the file is being uploaded, a progress bar shows the progress of the uploading process for a better UX.

As said previously, when in "Share from Dropbox" mode, the file explorer displays all the files inside a certain folder. All the image and video files have a thumbnail associated, which is useful in order to provide a quick overview to the user when seeing the files in a list. This way, it is a must to download these files' thumbnails in order to provide a great UX. However, the download of these small images have a downside: they consume the user's mobile data plan. Thus, it is necessary to optimize this in order to consume as less data as possible. A mechanism was implemented in order to solve this downside using Core Data – a way of persisting data on iOS which increases the responsiveness of applications. In short, for each file in the listing, it is verified if they have an available thumbnail. If so, it is downloaded and persisted in a Core Data entity. After that, each time the user returns to that same directory screen, the thumbnail is directly loaded from the database instead of being downloaded once again. This process brings two major advantages to the feature: it reduces the consuming of mobile data and increases the responsiveness of the file explorer 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.

5.1.4. Backup and Restore of conversations

As shown in 4.7. Backup and Restore of conversations and *Appendix C – Architecture: Backup and Restore of conversations*, the UI side of this feature is minimalist. However, it is the entry point to all the logic that empowers the backup and restore feature.

In order to implement this feature, I faced multiple challenges and difficulties. Some of those were already know since the beginning, such as developing in WIT's communication stack, but others were unpredicted until the implementation of the feature, like the incompatibility of both Core and Datastore/Sync SDKs when working together. This way, the current section will be divided in multiple topics in order to individually detail each of the faced challenges.

Unified SDK

There are three features that are possible due to the integration with Dropbox: "Save to Dropbox", "Share from Dropbox", and "Backup and restore of conversations". The first two require the Core SDK, while the last uses the Datastore/Sync SDK to interact with Dropbox Servers. It is intended that all these features may be simultaneously available inside the same application. This way, both Core and Datastore/Sync SDK frameworks must be inside the application at the same time.

These SDKs are not designed to interact simultaneously within the same application. First of all, the framework files for each SDK have the same name, meaning that one is replaced when adding the other one to the same project. Once Dropbox provides the source code of Core SDK, adding the Datastore/Sync SDK framework and the Core SDK source code can solve this problem. However, there are many issues that were discovered when compiling the code:

- The Core SDK source code cannot be compiled using ARC – a compiler feature that provides automatic memory management of Objective-C objects;
- There are several duplicate symbols because both the SDKs have some common parts that use the very same variables and constants;
- Both SDKs have classes with the same but with different logic.

In order to find a way of working with both SDKs simultaneously, the Dropbox Forums [83] were the first place to be consulted. Someone had already had the same problem as me, but the posts were not the answer (at least not completely). Guided by the compiler errors and by some answers already provided, I managed to integrate both the SDKs inside the same application. However, not everything was fully working. There was an error ("[ERROR] unable to verify link request.") occurring during the authentication process. After questioning around the forums and analyzing the Core SDK's class used on the authentication process, the problem was found: after authenticating, the SDK removes a value from the database file. Once both the SDKs are now using the same database, that value is removed after the authentication of one of the SDKs, not being available for the other and thus failing. The solution found involved editing that Core SDK's class in order not to erase that value from the file. This way, the Core SDK must be authenticated first and only then the Datastore/Sync can run its authentication process. When the authentication request is made, the value is created and added to the database. Then, the Core SDK does its authentication process without deleting that value. Finally, the Datastore SDK does that same process but erases the value from the database. This way, everything works fine and the final state of that internal database is the intended one.

Once there was no guide with all the steps required for this unification in the Internet, I produced a list of steps and shared it online in order to ease the job of anyone else that needs to do same task I did. This guide was published on Stack Overflow by answering to two questions regarding a related issue.

In order not to extend even more this section, the unification guide won't be detailed. However it can be consulted on the first answer for the questions [84] and [85].

Approach

The chosen approach for the backup mechanism was active synchronization, meaning that the only change that occurs to the application's database (e.g. new message) is promptly persisted in the local datastore. This guarantees the best performance because there is no need to collect all the data on a very big chunk when the user wants to perform the backup, this way avoiding a period where the device's resources are largely consumed. This would make the application slower and thus decrease the UX.

There was an easier way of implementing this feature: when the user wants to backup, all the data is collected from the database, zipped into a single file, and sent to their Dropbox account. However, this would shutdown the changes of creating a real-time synchronization mechanism between multiple devices.

A real-time synchronization mechanism is easily implemented by doing slight modifications to the current active synchronization mechanism. In first place, the history changes had to be synchronized to the remote datastore when they occurred, which is not happening currently. Finally, the synchronization manager had to be listening for remote changes on all the datastores in order to download them and add them on the device.

Backup of file transfers

The "Backup and Restore of conversations" was designed in order to manage the backup of restore of chat messages, file transfer, and locations. As stated in *Appendix C – Architecture: Application Programming Interfaces*, the maximum size of a datastore record is 100KB, being extremely small to store the media file. This way, it was necessary to use the Sync SDK in order to synchronize the files, just like the Datastore SDK synchronizes the datastores.

All the structure is ready for that, once the SDKs are now unified and the authentication process is working properly. However, an unexpected problem related to the Dropbox permissions arose. The RCS application is registered in Dropbox with full permissions mode because of the "Share from Dropbox" feature, once it is necessary to access any directory and any kind of file inside the user's Dropbox. However, this mode is only compatible with the Core API, meaning that no other API may have this amount of permissions. In other words, the Sync API does not work with the amount of permissions that are required, which makes it impossible to synchronize the media file and, thus, the complete file transfers. Once the "Share from Dropbox" is a feature that is highly appreciated by operators, the only solution was to implement the "Backup and Restore of conversation" for chat messages and locations only.

5.2. Evaluation

The validation of the developed solution is a fundamental step on any software project. It is desired that the application is delivered to the user with zero bugs, and with the best UI/UX possible in order to make them enjoy and, consequently, use it.

Various tests were designed and executed in order to validate the implemented features. All the types of testing that took place in this internship are described in the following sub-sections. The reading of *Appendix D – Evaluation* is highly recommended once it contains the complete set of functional and non-functional tests that were designed and performed.

5.2.1. Functional Testing

Functional testing is a quality assurance process. In this type of testing, the use cases are tested by giving an input and analyzing its output. In the current context, the user interacts in some way with the application, and it must produce the desired result in order for the feature to work properly and the user to have the intended user experience. Once most of the features that compose this internship are visual, meaning that they present visual information to the user upon the intern logic, it is of the most importance to guarantee that all the developed use cases produce the output that was planned.

A total of ninety functional tests were designed and executed in order to check the behavior of all the implemented features within the RCS application. Forty-two of those tests apply to the "Real-time Location Share" feature because it is the feature that provides more interaction with the user and, thus, a various set of outputs are expected.

The testing process has been continuous, meaning that it went along all the implementation. This way, it is natural that the tests were executed multiple times until all the bugs were corrected. Table 6 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
Ephemeral Messaging	2	14	0	16
Real-time Location Share	5	37	0	42
Save and Share from Dropbox	1	19	0	20
Backup and Restore of conversations	2	10	0	12

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

As it can be seen, the testing process allowed the discovery of bugs and problems in the features, which were iteratively solved until the last run, where none remained.

The functional tests that were designed and executed for each feature are described in *Appendix D – Evaluation: Functional Testing.*

5.2.2. Networking

All the implemented features rely on an internet connection in order to operate correctly. This way, it is imperative that these features properly handle network losses during the execution of any of its use cases. Disabling the feature or delaying the execution of its operations are some of the mechanisms that are used in situations like this.

Regarding network-based testing, twelve tests were executed in order to validate the implemented features, which can be seen in Table 7.

	First run		**Final run**	
Feature	Failed	Passed	Failed	Passed
Ephemeral Messaging	0	1	0	1
Real-time Location Share	1	4	0	5
Save and Share from Dropbox	0	3	0	3
Backup and Restore of conversations	0	3	0	3

Table 7 - Network testing results per feature for both first and last runs

As the results show, the only feature that had a network-related problem was the Real-time Location Share. The issue was related to the test case N.4.3, in *Appendix D – Evaluation: Networking*, where there were some minor problems with the algorithm regarding the fetching of directions when the connection was restored.

Once again, it is highly recommended to read *Appendix D – Evaluation: Networking* in order to better understand the executed network tests.

5.2.3. Usability

The User Experience is one of the most important components in an application nowadays. An application can provide many features to the user, but nobody will use it if it has a bad UX. This way, it is important to provide this application to different kinds of people in order to collect feedback about their sensation when interacting with the implemented features.

After ending the development phase, usability tests were performed to the implemented features. To achieve this, a total of 17 people from WIT Software exectued these tests. Unfortunately, it was not possible to collect feedback from people outside the company once these features are not yet officially presented by the company and must remain as private as possible.

The sample used to execute these tests was chosen randomly, containing users from several mobile operating systems. However, the most important factor is if they had already interacted with the RCS application for iOS. Unfortunately, eleven people from this testing sample had previously had some sort of contact with the application.

The following topics provide some comments about the usability tests executed to each developed feature, which are listed, described, and analyzed in *Appendix D – Evaluation: Usability*.

Ephemeral Messaging

In order to tests this feature, the trial set was challenged to change the lifetime of outgoing ephemeral messages to a specific value, and then to send an existing image in ephemeral mode to a specified contact. This task could be executed with thirteen actions.

The obtained results show that the UX of this feature needs to be improved, once only 29% of the users managed to execute it in the minimum required actions. By analyzing the interactions of each user upon the execution of this task, three major problems were identified:

- It is not clear which setting options must be chosen in order to define the ephemeral messages lifetime;
- The options button is some times confused with the "Share" button inside the chat window;
- Some users choose to take a photo instead of picking one from the gallery.

The execution of this kind of test allowed the identification of multiple UX aspects that must be further improved in order for this feature to be accepted by the final users. This will constitute part of the future work.

Real-time Location Share

Regarding this feature, the trial set was challenged to share real-time location in driving mode for five minutes with a specified contact. This task can be executed with only six actions.

The results from this test are quite pleasant, once 65% of the users performed this task with the minimum amount of required action. The average number of steps taken by the trial set was seven, which is very near the desired amount.

Save and Share from Dropbox

A task related to the "Share to Dropbox" feature was selected to this usability test. Once the flow of both features converge in a part, there is no need to test both of them. This time, the trial set was challenged to send a file from the root folder of a specified Dropbox account to a certain contact.

The results from this test showed that this is the feature with best UX, once eleven users managed to execute this with only six actions. The remaining six already did all the correct flow. However, they selected files with a size that displayed a warning, requiring their action to dismiss it.

Backup and Restore of conversations

In order to test this last feature, the trial set was challenged to log in to a specified Dropbox account, and then restore the conversations from an existing backup. In order to perform this task, only seven actions were necessary.

The results obtained are very satisfactory because the average of steps done by the trial set was only eight. Once again, watching the users executing the task helped to identify the reason of taking more actions. In this case, the users did all the required flow in the minimum amount of steps, but some misclicked the keyboard's keys and entered the wrong credentials. It is assumed that this had to do with those users being accustomed to different keyboards than the iOS's one.

6. Conclusion

This final chapter marks the end of this report and the end of the one-year internship at WIT Software.

The first section presents an overview of all the work done during the internship.

The second section provides explains the work that can still be done after the end of the internship, underlining the existing opportunities to improve certain features.

Finally, the third section presents some personal thoughts about the internship.

6.1. Contributions Overview

Since the beginning of the internship, the goal was to enrich WIT's RCS application for iPhone with features that could differentiate it from its competitors. With this goal in mind, the initial proposal was composed by features that required the integration with social networks and cloud storages. In order to embrace the new trends, both "Ephemeral Messaging" and "Real-time Location Share" were added to the scope of the internship. These additions, together with features suggested by the trainees (such as the "Backup and Restore of conversations") made this internship even more complete, for both the company and me.

The current section provides an overview of all the implemented features. Now that the work is finished, it is time to reflect and analyze each of the features.

Ephemeral Messaging

This feature was introduced at request of WIT's CEO, PhD Professor Luís Silva, at the end of the year of 2013. This was the first challenge I faced once I did not know much about Objective-C nor the complex internal structure of the product. This way, this was an important stage on my growth as an iOS developer that let me deeply increase my knowledge.

The concept behind this feature seems straightforward. However, details such as presenting a decreasing timer or erasing the corresponding entry from the chat window are important to provide a good UX to the user. The concept of this feature is fully functional, but there are some issues regarding the UX that need to be improved, as concluded in the usability tests reported in section 5.2.3. Usability.

Real-time Location Share

This feature was introduced in the beginning of the current year in order to be developed in time for the Mobile World Congress 2014. What we thought to be a relatively easy feature to develop, turned out to be a very complex one, from all the aspects related to the UI, to all the logic behind the calculation of the correct route for Driving mode. The degree of complexity of this feature can be seen by the time that was spent developing it, and the number of iterations spent in order to obtain the desired final result.

This feature was a huge success on that conference, which is revealed the number of operators that showed interested on it to compose their RCS applications.

Save and Share from Dropbox

This feature consisted of the initial internship proposal. Its development is not just an integration with Dropbox. Once all the components were designed to be completely generic, any other cloud storage can be added effortlessly to the application with just a small amount of specific logic.

Personally, this feature was extremely enriching once it taught me the importance of the reutilization of components, between many other aspects.

Backup and Restore of conversations

This feature, suggested by my colleague Paulo Santos and I, was probably the most complex feature once it required descending to a lower level of abstraction and a whole new level of complexity. The COMLib is with no doubt a beautiful piece of multi-platform software that contains a various set of services. Although only developing on a single module, the complete refactor of the Sync Module was a totally enriching task. Exploring and understanding a module that was not made by yourself, and trying to change it in order to adapt it to what you need is really hard. However, it was successfully managed to refactor that whole module in order to contain the new remote sync sub module, but still maintaining all of its original functionalities.

This feature subjected me to a very hard growth and learning process, but with no doubt made me a better software engineer.

6.2. Future Work

The WIT's RCS solution is an evolving application, which means that this project does not end here.

Firstly, the "Contact Enrichment" and "Contact Timelines" features were not yet implemented. They were part of the initial internship proposal but, given the introduction of the "Ephemeral Messaging" and "Real-time Location Share" features, there was no time to conclude them on schedule. However, their architecture and UI proposals were done during the internship, which eases and speeds up the future implementation process.

In second place, there are some features that need or may be improved. The "Ephemeral Messaging" feature needs to be highly improved at the UI level in order to produce a better UX. The "Backup and Restore of conversations" feature may evolve to a multiple device synchronization mechanism with some few changes.

Finally, iOS 8 was recently presented [86] and contains multiple new features that give developers new possibilities during the implementation of their applications. There are two that constitute great opportunity for the RCS application: "App Extensions" – binary that extends custom functionality and content beyond a certain application, making some of its features available to users while using other apps – and "CloudKit" - framework that provides interfaces for moving data between applications and iCloud.

App Extensions can allow, for example, to request files directly to Dropbox applications instead of using a custom browser. This will allow the application to be even more fluid and natively integrated.

CloudKit finally provides an easy way to use iCloud to synchronize content, including both application data, like Dropbox datastores, and media files, like any usual cloud service. This opens doors to two new possibilities:

- Save and Share files from iCloud, just like it is done with Dropbox;
- Easily synchronize the existing conversations across all the iOS devices. This is useful, for example, to synchronize a certain user's iPhone with its iPad, allowing them to have both applications always with the most recent chat history.

6.3. Final Remarks

To conclude this report, it is now time to present some personal thoughts and experiences about the internship.

This one-year internship has been a whole new experience, full of challenges and lessons learned. Fortunately, I was given the freedom to suggest and contribute to the improvement of WIT's RCS application for iPhone. Having the possibility to implement features that I suggested – the Backup and Restore of conversations – reveals that the company listens to everybody who wants it to grow, including the interns.

The initial plan for the internship contained multiple suggestions for integration with social networks and cloud storage service. Some of them are not completed by now, as stated previously. However, I think the balance could not be better: the implementation of two completely different features – "Ephemeral Messaging" and "Real-time Location Share" – allowed me to explore totally different areas of Objective-C programming. For example, the "Real-time Location Share" feature required me to learn multiple different aspects: how to interact with the Apple Maps API, how to fetch and process geo-location positions from the built-in GPS chip, and even to build an algorithm that had to be iteratively improved in order to produce the desired and acceptable results. All of these requirements are not simple but are necessary steps to produce a great feature that arouses the interest of MNOs. That is quite rewarding!

In the end, this was a great enriching experience and I would not go back if I had the chance. Was it tough? Yes it was! But it allowed me to grow both personally and professionally in a way I never thought to be possible.

References

1 Lunden I. Gartner: 456M Phones Sold In Q3, 55% Of Them Smartphones; Android At 82% Share, Samsung A Flat Leader. [Internet]. 2013 [cited 2014 January 19]. Available from: http://techcrunch.com/2013/11/14/gartner-456m-phones-sold-in-q3-55-smartphone-android-at-82-share-samsung-a-flat-leader/.

2 Evans B. Mobile is eating the world. [Internet]. 2013 [cited 2014 January 14]. Available from: http://www.slideshare.net/bge20/2013-11-mobile-eating-the-world?utm_source=slideshow01&utm_medium=ssemail&utm_campaign=share_slideshow_loggedout.

3 Janssen C. What is an Over-the-Top Application (OTT)? [Internet]. [cited 2013 October 24]. Available from: http://www.techopedia.com/definition/29145/over-the-top-application-ott.

4 Whitfield K. 17 Incredible Facts about Mobile Messaging that you should know. [Internet]. 2013 [cited 2014 January 19]. Available from: http://www.portioresearch.com/en/blog/2013/17-incredible-facts-about-mobile-messaging-that-you-should-know.aspx.

5 Wikipedia. GSM Association. [Internet]. [cited 2014 January 19]. Available from: http://en.wikipedia.org/wiki/GSM_Association.

6 GSMA. Rich Communications | Future Communications. [Internet]. [cited 2014 January 19]. Available from: http://www.gsma.com/futurecommunications/rcs/.

7 GSMA. Specs and Product Docs | Future Communications. [Internet]. [cited 2014 January 19]. Available from: http://www.gsma.com/futurecommunications/rcs/specs-and-product-docs/.

8 Lardinois F. Report: Cloud Storage Services Now Have Over 375M Users, Could Reach 500M By Year-End. [Internet]. 2012 [cited 2013 December 24]. Available from: http://techcrunch.com/2012/10/15/report-cloud-storage-services-now-have-over-375m-users-could-reach-500m-by-year-end/.

9 Facebook. What's a sticker and how do I send one? [Internet]. [cited 2014 January 4]. Available from: https://www.facebook.com/help/333033546818929.

10 Geron T. Facebook Messenger Takes On SMS, With No Account Needed. [Internet]. 2012 [cited 2013 October 28]. Available from: http://www.forbes.com/sites/tomiogeron/2012/12/04/facebook-messenger-takes-on-sms-with-no-account-needed/.

11 Facebook. Key Facts - Facebook's latest news, announcements and media resources. [Internet]. [cited 2013 December 27]. Available from: http://newsroom.fb.com/Key-Facts.

12 Constine J. Facebook Mobile User Counts Revealed: 192M Android, 147M iPhone, 48M iPad, 56M Messenger. [Internet]. 2013 [cited 2013 October 28]. Available from: http://techcrunch.com/2013/01/04/how-many-mobile-users-does-facebook-have/.

13 iTunes. Facebook Messenger on the App Store on iTunes. [Internet]. [cited 2014 January 4]. Available from: https://itunes.apple.com/us/app/facebook-messenger/id454638411.

14 Google Play. Facebook Messenger - Android Apps on Google Play. [Internet]. [cited 2014 January 4]. Available from: https://play.google.com/store/apps/details?id=com.facebook.orca&hl=en.

15 Topolsky J. Steve Jobs live from WWDC 2010. [Internet]. 2010 [cited 2013 October 28]. Available from: http://www.engadget.com/2010/06/07/steve-jobs-live-from-wwdc-

2010/.

16 Murph D. WWDC 2011 liveblog/ Steve Jobs talks iOS 5, OS X Lion, iCloud and more! [Internet]. 2011 [cited 2013 October 28]. Available from: http://www.engadget.com/2011/06/06/wwdc-2011-liveblog-steve-jobs-talks-ios-5-os-x-lion-icloud-an/.

17 Smith J. Apple's popular iMessage and FaceTime services are suffering downtime [Update: back]. [Internet]. 2013 [cited 2013 October 28]. Available from: http://www.pocket-lint.com/news/120411-apple-imessage-down-april.

18 Protalinski E. Microsoft confirms Messenger will be retired and users migrated to Skype on March 15. [Internet]. 2013 [cited 2013 October 28]. Available from: http://thenextweb.com/microsoft/2013/01/09/microsoft-emails-messenger-users-to-let-them-know-the-service-is-retiring-on-march-15-and-to-upgrade-to-skype/.

19 Swider M. Microsoft highlights 299M Skype users, 1.5B Halo games played. [Internet]. 2013 [cited 2013 October 28]. Available from: http://www.techradar.com/news/software/operating-systems/xbox-live-upgrade-includes-%20300-000-servers-600-times-more-than-its-debut-1161749.

20 Skype. Skype Features - Call, video, instant message, text & share more online. [Internet]. [cited 2013 October 28]. Available from: http://www.skype.com/en/features/.

21 iTunes. Skype for iPhone on the App Store on iTunes. [Internet]. [cited 2014 January 4]. Available from: https://itunes.apple.com/us/app/skype-for-iphone/id304878510.

22 Google Play. Skype - free IM & video calls - Android Apps on Google Play. [Internet]. [cited 2014 January 4]. Available from: https://play.google.com/store/apps/details?id=com.skype.raider&hl=en.

23 Wauters R. 200.000.000 mobile users, for starters – Q&A with Viber CEO Talmon Marco. [Internet]. 2013 [cited 2013 October 26]. Available from: http://thenextweb.com/insider/2013/05/07/viber-ceo-talmon-marco-interview/#!rm52G.

24 Viber. Viber - Free calls, Free text messages, photo and location sharing. [Internet]. [cited 2013 October 26]. Available from: http://www.viber.com.

25 iTunes. Viber on the App Store on iTunes. [Internet]. [cited 2014 January 4]. Available from: https://itunes.apple.com/us/app/viber/id382617920.

26 Google Play. Viber - Android Apps on Google Play. [Internet]. [cited 2014 January 4]. Available from: https://play.google.com/store/apps/details?id=com.viber.voip&hl=en.

27 Bradshaw T. WhatsApp users get the message. [Internet]. 2011 [cited 2013 October 26]. Available from: http://www.ft.com/intl/cms/s/2/30fd99a2-0c60-11e1-88c6-00144feabdc0.html.

28 Newton C. WhatsApp now has over 400 million monthly users. [Internet]. 2013 [cited 2014 January 4]. Available from: http://www.theverge.com/2013/12/19/5228656/whatsapp-now-has-over-400-million-monthly-users.

29 WhatsApp. WhatsApp : Home. [Internet]. [cited 2013 October 26]. Available from: http://www.whatsapp.com.

30 iTunes. WhatsApp Messenger on the App Store on iTunes. [Internet]. [cited 2014 January 4]. Available from: https://itunes.apple.com/us/app/whatsapp-messenger/id310633997.

31 Google Play. WhatsApp Messenger - Android Apps on Google Play. [Internet]. [cited 2014 January 4]. Available from: https://play.google.com/store/apps/details?id=com.whatsapp&hl=en.

32 Facebook. Facebook Help Center. [Internet]. [cited 2013 December 27]. Available from: https://www.facebook.com/help/393277774048285.

33 iTunes. Facebook on the App Store on iTunes. [Internet]. [cited 2013 December 27]. Available from: https://itunes.apple.com/us/app/facebook/id284882215.

34 Google Play. Facebook - Android Apps on Google Play. [Internet]. [cited 2013 December 27]. Available from: https://play.google.com/store/apps/details?id=com.facebook.katana&hl=en.

35 Miller CC. Another Try by Google to Take On Facebook. [Internet]. 2011 [cited 2014 January 2]. Available from: http://www.nytimes.com/2011/06/29/technology/29google.html.

36 McGee M. Google+ Hits 300 Million Active Monthly "In-Stream" Users, 540 Million Across Google. [Internet]. 2013 [cited 2014 January 2]. Available from: http://marketingland.com/google-hits-300-million-active-monthly-in-stream-users-540-million-across-google-63354.

37 Google+. Get the most out of Google+ - Google+. [Internet]. [cited 2014 January 2]. Available from: http://www.google.com/+/learnmore/features.html?hl=en.

38 iTunes. Google+ on the App Store on iTunes. [Internet]. [cited 2014 January 2]. Available from: https://itunes.apple.com/us/app/google+/id447119634.

39 Google Play. Google+ - Android Apps on Google Play. [Internet]. [cited 2014 January 2]. Available from: https://play.google.com/store/apps/details?id=com.google.android.apps.plus&hl=en.

40 Edwards J. Twitter's 'Dark Pool': IPO Doesn't Mention 651 Million Users Who Abandoned Twitter. [Internet]. 2013 [cited 2014 January 3]. Available from: http://www.businessinsider.com/twitter-total-registered-users-v-monthly-active-users-2013-11.

41 iTunes. Twitter on the App Store on iTunes. [Internet]. [cited 2014 January 3]. Available from: https://itunes.apple.com/us/app/twitter/id333903271.

42 Google Play. Twitter - Android Apps on Google Play. [Internet]. [cited 2014 January 3]. Available from: https://play.google.com/store/apps/details?id=com.twitter.android&hl=en.

43 Constine J. Dropbox Hits 200M Users, Unveils New "For Business" Client Combining Work And Personal Files. [Internet]. 2013 [cited 2013 December 24]. Available from: http://techcrunch.com/2013/11/13/dropbox-hits-200-million-users-and-announces-new-products-for-businesses/.

44 iTunes. Dropbox on the App Store on iTunes. [Internet]. [cited 2013 December 24]. Available from: https://itunes.apple.com/us/app/dropbox/id327630330.

45 Google Play. Dropbox - Android Apps on Google Play. [Internet]. [cited 2013 December 24]. Available from: https://play.google.com/store/apps/details?id=com.dropbox.android&hl=en.

46 Gannes L. With 120M Users, Google Drive Gets Tighter Integration With Gmail. [Internet]. 2013 [cited 2013 December 24]. Available from: http://allthingsd.com/20131112/with-120m-users-google-drive-gets-tighter-integration-with-gmail/.

47 iTunes. Google Drive on the App Store on iTunes. [Internet]. [cited 2013 December 24]. Available from: https://itunes.apple.com/us/app/google-drive/id507874739.

48 Google Play. Google Drive - Android Apps on Google Play. [Internet]. [cited 2013 December 24]. Available from: https://play.google.com/store/apps/details?id=com.google.android.apps.docs&hl=en.

49 Shahine O. Over 250M people using SkyDrive. [Internet]. 2013 [cited 2013 December 24]. Available from: http://blogs.windows.com/skydrive/b/skydrive/archive/2013/05/06/over-250m-people-using-skydrive.aspx.

50 iTunes. SkyDrive on the App Store on iTunes. [Internet]. [cited 2013 December 24]. Available from: https://itunes.apple.com/us/app/skydrive/id477537958.

51 Google Play. SkyDrive - Android Apps on Google Play. [Internet]. [cited 2013 December 24]. Available from: https://play.google.com/store/apps/details?id=com.microsoft.skydrive&hl=en.

52 Gohring N. Is Microsoft's claim of 250 million SkyDrive users fuzzy math? [Internet]. Available from: http://www.infoworld.com/t/microsoft-windows/microsofts-claim-of-250-million-skydrive-users-fuzzy-math-218090.

53 Newton C. Confide brings its beautiful self-destructing messages to Android | The Verge. [Internet]. 2014 Available from: http://www.theverge.com/2014/4/24/5644238/confide-brings-its-beautiful-self-destructing-messages-to-android.

54 Confide. Your off-the-record messenger. [Internet]. 2014 Available from: https://getconfide.com.

55 iTunes. Confide on the App Store on iTunes. [Internet]. [cited 2014 Jun 15]. Available from: https://itunes.apple.com/us/app/confide/id779883419.

56 Google Play. Confide - Android Apps on Google Play. [Internet]. [cited 2014 Jun 15]. Available from: https://play.google.com/store/apps/details?id=cm.confide.android&hl=en.

57 Snapchat. Snapchat. [Internet]. [cited 2014 January 4]. Available from: http://blog.snapchat.com/page/3.

58 Shontell A. Snapchat active users exceed 30 million. [Internet]. 2013 [cited 2014 January 3]. Available from: http://www.businessinsider.com/snapchat-active-users-exceed-30-million-2013-12.

59 Keller J. Facebook's Poke Is a Wild Success—for Rival Snapchat. [Internet]. 2012 [cited 2014 January 3]. Available from: http://www.businessweek.com/articles/2012-12-28/facebooks-poke-is-a-wild-success-for-rival-snapchat.

60 Souppouris A. Google reportedly tried to outbid Facebook for Snapchat with $4 billion offer. [Internet]. 2013 [cited 2014 January 4]. Available from: http://www.theverge.com/2013/11/15/5106950/google-snapchat-4-billion-buyout-rumor.

61 iTunes. Snapchat on the App Store on iTunes. [Internet]. [cited 2014 January 3]. Available from: https://itunes.apple.com/us/app/snapchat/id447188370.

62 Google Play. Snapchat - Android Apps on Google Play. [Internet]. [cited 2014 January 3]. Available from: https://play.google.com/store/apps/details?id=com.snapchat.android&hl=en.

63 Mathis J. Cook: More than 700 million iOS devices sold | Macworld. [Internet]. 2013 [cited 2014 Jun 15]. Available from: http://www.macworld.com/article/2048491/cook-more-than-700-million-ios-devices-sold.html.

64 iTunes. Find My Friends on the App Store on iTunes. [Internet]. [cited 2014 Jun 15]. Available from: https://itunes.apple.com/us/app/find-my-friends/id466122094.

65 Bolton M. Press Release. [Internet]. 2013 [cited 2014 Jun 15]. Available from: http://www.glympse.com/news/press/15.

66 iTunes. Glympse - Share location with friends on the App Store on iTunes. [Internet].

[cited 2014 Jun 15]. Available from: https://itunes.apple.com/us/app/glympse-share-location-friends/id330316698.

67 Google play. Glympse - Share your location - Android Apps on Google Play. [Internet]. [cited 2014 Jun 15]. Available from: https://play.google.com/store/apps/details?id=com.glympse.android.glympse&hl=en.

68 Schwaber K, Sutherland J. The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game. [Internet]. [cited 2014 January 22]. Available from: https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide.pdf#zoom=100.

69 Wikipedia. User story. [Internet]. [cited 2013 January 23]. Available from: http://en.wikipedia.org/wiki/User_story.

70 Neuburg M. iOS 7 Programming Fundamentals. O'Reilly Media; 2013.

71 Neuberg M. Programming iOS 7, 4th Edition. O'Reilly Media; 2013.

72 Nahavandipoor V. iOS 7 Programming Cookbook. O'Reilly Media; 2013.

73 Apple. iOS Developer Library. [Internet]. Available from: https://developer.apple.com/library/ios/navigation/.

74 StackOverflow. [Internet]. Available from: http://stackoverflow.com.

75 Stroustrup B. The C++ Programming Language: Special Edition (3rd Edition). Addison-Wesley Professional; 2000.

76 Boost Community. Boost C++ Libraries. [Internet]. Available from: http://www.boost.org.

77 ProjectFuture2. What is a project risk. [Internet]. [cited 2014 January 19]. Available from: http://www.projectfuture.net/uk/projectfuture-software/frequently-asked-questions/what-is-a-projectrisk.

78 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.

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

80 Wikipedia. Bézier curve. [Internet]. [cited 2013 June 25]. Available from: http://en.wikipedia.org/wiki/Bézier_curve.

81 Poplauschi B. SDWebImage. [Internet]. [cited 2014 Jun 28]. Available from: https://github.com/rs/SDWebImage.

82 Apple. NSRunLoop Class Reference. [Internet]. [cited 2014 Jun 15]. Available from: https://developer.apple.com/library/ios/Documentation/Cocoa/Reference/Foundation/Classes/NSRunLoop_Class/Reference/Reference.html.

83 Dropbox. Dropbox Forums. [Internet]. [cited 2014 Jun 25]. Available from: https://forums.dropbox.com.

84 Stack Overflow. linker error when using dropbox core api and its datastore api together in ios. [Internet]. Available from: http://stackoverflow.com/questions/19294594/linker-error-when-using-dropbox-core-api-and-its-datastore-api-together-in-ios/23867651#23867651.

85 Stack Overflow. ios - Dropbox Core API and Sync API. [Internet]. Available from: http://stackoverflow.com/questions/21930557/dropbox-core-api-and-sync-api/23867763#23867763.

86 Swider M. iOS 8 release date, news and features | News | TechRadar. [Internet]. 2014 [cited 2014 June 29]. Available from: http://www.techradar.com/news/phone-and-communications/mobile-phones/ios-8-10-things-we-want-to-see-1166133.