Masters in Informatics Engineering
Internship
Final Report

Development of Corporate Collaboration Functionalities for Communication Services

João Miguel Montenegro dos Santos Barroso Penetra

jpenetra@student.dei.uc.pt

DEI Supervisor:

Prof. Dr. Fernando Barros

Wit-Software Supervisor:

Eng. Paulo Sousa

Date: 4th September 2013

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

Abstract

Nowadays, due to globalization, increase of competitiveness and organizational structure, people are required to move around the globe in order to achieve their goals. The need to commute is based on the necessity to meet with collaborators, clients, prospect clients or any other person that is on a different location. Since this need for travelling consumes a large amount of money and time resources, it is necessary to find an alternative that allows people to meet and collaborate anywhere. Consequently, this internship attempts to overcome this collaboration need with two prototypes. The first one is a Web Application that is interoperable with other types of devices in order to maximize the opportunity to collaborate. Although it is not possible to implement all functionalities in every type of device, joining a video conference from a cell phone only with audio is still more valuable than not joining at all. The only way of making all of this possible is using a mature architecture such as the IP Multimedia Subsystem. The second one is an Android Application capable of communicating with XMPP Networks.

Keywords

“Android”; “Audio Calls”, “Business Collaboration Service”, “Conference Calls”, “Instant Messaging”, “IP Multimedia Subsystem”, “File Transfer”, “SIP”, “Video Calls”, “Web Browser”, “XMPP”

Table of Contents

Index of Figures

Index of Tables

Acronyms

3GPP	3rd Generation Partnership Project
AAA	Authentication, Authorization and Accounting
AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
CS	Circuit Switched
CSCF	Call Session Control Function
HSS	Home Subscriber Server
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
I-CSCF	Interrogating Call Session Control Function
IETF	Internet Engineering Taskforce
IMS	IP Multimedia Subsystem
IMS-MGW	IMS Media Gateway Function
IP	Internet Protocol
JCP	Java Community Process
JMS	Java Message Service
JNI	Java Native Interface
JSF	JavaServer Faces
JSP	JavaServer Pages
LTE	Long Term Evolution
Mbps	Megabits per second
MCU	Multipoint Control Unit
MMS	Multimedia Messaging Service
MMTel	Multimedia Telephony Services
MSS	Mobicents SIP Servlets
MRF	Media Resource Function
MSRP	Message Session Relay Protocol
NIO	New Input/Output
NGN	Next Generation Networking

PCM	Pulse-code Modulation
P-CSCF	Proxy Call Session Control Function
POJO	Plain Old Java Object
RCS	Rich Communication Suite
RCS-e	Rich Communication Suite Enhanced
REST	Representational State Transfer
RTP	Real-time Transfer Protocol
S-CSCF	Serving Call Session Control Function
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SMS	Short Message Service
TCP	Transmission Control Protocol
VoIP	Voice over IP
UE	User Equipment
URI	Uniform Resource Identifier

Section 1 - Introduction

1.1. Internship Presentation

The main purpose of this document is to describe the work that was developed during the subject of Dissertation/Internship, constituent of the Masters in Informatics Engineering. The internship took place at the company WIT-Software and its main objective was to develop Corporate Collaboration functionalities for Communication Services.

1.1.1. Institution

WIT is a software company that develops solutions for mobile telecommunications operators. These solutions are based on IP Multimedia Subsystem (IMS) and include voice (VoIP, Mobile VoIP and Voice over LTE), messaging (SMS and MMS), Rich Communication Suite (RCS2.0 and RCS-e) and Multimedia Telephony Services (MMTel). The company was founded in 2001 and it has costumers in more than 15 countries [1].

1.1.2. Motivation

The world we live in, due to its increasing globalization and competitiveness, requires people to be in constant commuting. The purpose of this commuting may be meeting with clients or prospect clients, working with colleagues, attending conferences and several other reasons. This need to move people around the globe is one of the most important motivations that support the development of a service that enables people to collaborate. That said, the platforms that are being developed will help people inside organizations achieve their goals by providing a means of communication between the people on the same team or between the teams and their clients. In addition to providing a means of communication, it is also important that the collaboration service allows teams to have a secure place to keep their work organized and easily accessible.

Although it is difficult to find how the need for travelling impacts companies' expenses and how much can be decreased with collaboration tools, it is important to research this with the aim of supporting the development of this tool. A study done in 2012 by *Business Travel News* [2] interviewed 227 Small and Medium Enterprises in the United States and comprises expenses from air and ground transportation, hotels, meals, parking, etecetera. The study concluded that in 2011, the minimum spent, was half a million U.S dollars and the maximum twelve million U.S. dollars. The following table demonstrates the percentage of companies in a certain interval.

Table 1 - Travel expenses from 227 US Small and Medium Enterprises

Cost in Millions of U.S. dollars	[0.5,1[	[1,2[	[2,3[	[3,4[	[4,5[	[5, 6[	[6,7[	[7,8[	[8,9[	[9,10[	[10,11[	[11,12[
Percentage of Companies	41%	14%	15%	7%	6%	5%	2%	4%	2%	2%	1%	1%

Ideally these expenses should be compared with the profits that each company had in order to take valuable conclusions. However, it was not possible to access those values.

1.1.3. Objectives

The objectives can be divided into two subsets: Project and Personal.

1.1.3.1. Project

Considering the motivational aspects that were presented the main functionalities offered are: User Presence, Instant Messaging, Audio and Video Conferencing, File Transfer and Project Management. The User Presence allows users to maintain a contact list with information about contacts' availability. Instant Messaging will be available in the form of group chat and one to one chat. Audio and Video conferencing will help people having a richer form of communication in order to make decisions and solve problems. File Transfer enables teams to share documents between their members. Finally, Project Management will consist of creating tasks and associating them to someone in order to keep work organized.

After the intermediate presentation, a necessity from the company emerged which resulted in a change of plans. Besides what was initially planned, it was decided that it would be useful to extend an Android Application created by the company using another protocol. This protocol is named eXtensible Messaging and Presence Protocol (XMPP [3]) and as its name suggests, it focuses on the development of functionalities related with the exchange of Instant Messages and User Presence. These features have the same purpose of the ones present in the Collaboration Service and they will be the focus of this prototype. Although it would be possible to connect the prototype with the main service this route was not followed due to time limitations. Though, during the architecture phase, it will be explained how is this convergence possible.

The Android application that is going to be extended is an application developed by the company that brings SMS, MMS and Rich Communication Services [4] (Chat, File Transfer, VoIP, etc) to the same platform with the purpose of merging the most used communication services into a whole application. Therefore, the objective of this prototype is adding another protocol that increases even more the amount of different ways to communicate with other people.

1.1.3.2. Personal

Besides all the technical knowledge that was acquired in new fields such as IMS architecture, SIP or XMPP this project also allowed improving skills acquired in the past and gaining experience working with agile methodologies in an enterprise environment.

1.1.4. Contextualization

As it was highlighted previously, one of the most important objectives of having a collaboration service is the opportunity of working together independently of the current localization. Therefore, one of the most important requirements is portability. Besides being presented as a web application that can be accessed from any Web Browser around the globe, it is very important that there can be other equipment that can be used to put people in contact. This way, it is essential that the architecture chosen lets users in a Web Browser communicate with any kind of Smartphone, Tablet, Cell Phone, Soft Phone and etcetera. To achieve this, and considering the company's clients, it was decided that the most appropriate architecture for the collaboration service is the IP Multimedia Subsystem (IMS). IMS is a network architecture designed by the 3rd Generation Partnership Project (3GPP) commonly used by telecommunication providers who offer multimedia services. It is a standard of the Next Generation Networks (NGN) and the deciding factor to use it in the collaboration service is the fact that it enables establishing multimedia sessions between different platforms. For example, it is possible to have a normal cell phone joining a conference call that is happening between a Web Browser and a Desktop Application. In order to make this possible, the Internet Engineering Taskforce (IETF) built a highly specified infrastructure on top of open Internet Protocols (IP). The main protocol that is used is named Session Initiation Protocol (SIP) and its purpose is to start, alter and terminate media sessions between users. Once this connection has been established, IMS mainly uses Real-time Transfer Protocol (RTP) to exchange the audio or video and it uses Message Session Relay Protocol (MSRP) to exchange instant messages and files. The Annex A (Technical Background) describes this architecture more thoroughly.

Like it has already been mentioned, XMPP is the protocol chosen for the Android Application. Besides fulfilling the portability requirement that is vital for a communication platform, this protocol is also highly specified by the XMPP Standards Foundation [3] and the IETF. These specifications are ideal because they allow developers to implement interoperable solutions without any additional effort. The fact that this protocol is used by well-known companies like Google [5] and Facebook [6] on their messaging systems can also be a great advantage. With the same protocol, it is possible to create an application that also aggregates two of the most used messaging systems. Furthermore, the fact that protocol is built on top of XML gives another great advantage: unlimited extensibility. XMPP supports the implementation of major features such as Instant Messaging, Contact List and Presence Information using XML which is also used as a signalling protocol to enable File Transfers and Audio and Video Calls. XMPP is supported by Client-Server and Server-to-Server architectures. The Client-Server architecture is used when clients need to communicate with the server of their own domain while Server-to-Server is used when clients communicate with clients of other domains. In order to obtain a more thorough description please consult the Annex A.

1.1.5. Risks

Any software project always has its own risks associated. The risks for this project are described below.

1.1.5.1. Learning curve of new technologies

This internship presents several new technologies to the intern. Although the intern has previous knowledge of SIP during the subject of Internet Applications, the IMS architecture represents a whole new world. The familiarization with this architecture and with other new technologies represents a risk because it might cause delays and change the implementation plan.

In order to fight this risk, it was decided that the first semester would consist mostly in researching and implementing prototypes that allowed the familiarization with the new technologies.

1.1.5.2. Inexperience of the intern

This project is, undoubtedly, the biggest challenge that the author has faced throughout the five years of the course of Informatics Engineering. The challenge of planning and implementing a year-long project represents new ground to the author which increases the risk of failure. Hopefully all the support that is available from the company and from the University will be a great help to achieve the desired success.

1.1.5.3. Divergences and necessities

Sometimes it is possible that divergences between the choices of the intern and the company's necessities occur. These divergences can cause alterations and delays in what was initially planned.

This risk became real after the intermediate presentation. The company's necessity to create an Android prototype using the XMPP protocol obliged the intern to alter his plans which also caused the delay of the project delivery.

1.1.6. Structure of the Document

This document is divided in 5 distinct sections.

Section 1 – Introduction – This first section presents the institution where the internship is taking place, the motivation that supports the project, its objectives and it ends with a brief contextualization.

Section 2 – Collaboration Service – This second section describes the several phases of development of this project. It starts with the State of the art of the competitors. After this, it presents the platform design where requirements, high-level prototypes, technologies and the architecture of the system are described. Furthermore, this section contains the description of how was the system implemented and what was done in order to assure its quality. Finally, it presents the outcomes of this service.

Section 3 – Android XMPP Client – This section presents an analysis of the development of the Android client. Although it possesses a structure similar to Section 2, it was decided that it would be better to separate them to achieve maximum clarification. The other alternative was sub-dividing each chapter (state of the art, platform design, implementation and etcetera) in two (Collaboration Service and Android XMPP Client) but this route would break the normal reading flow and would probably confuse the reader.

Section 4 – Methodology – This section describes the methodology that was used in this project and presents what was done during the internship.

Section 5 – Conclusions – This chapter presents a conclusion of the work developed during the internship, the future work and the lessons learned.

Section 2 - Collaboration Service

2.1. State of the Art

The Web Collaboration Market has been around for more than a decade. During this period, there were a large number of companies that released significantly different products but always with the goal of succeeding in this market. There are well-known companies such as Adobe, Cisco, IBM, and Microsoft that attempt to create complete solutions for companies of the same stature but there are other companies such as 37signals or Atlassian that focus on helping smaller companies. This first subsection starts with a description of the state of the art of these Collaboration tools. In this description, high-level features such as Instant Messaging or Audio/Video Conference might be present in all but they will always be described because they comprise smaller features that differentiate tools. The second subsection creates comparison between each of the tools and the third and last contains a market analysis.

2.1.1. Competitors

2.1.1.1. Adobe Connect

Adobe Connect [7] is a web conferencing platform created with the objective of improving productivity inside organizations. Its most important features are: High Quality Video Conferencing, File Transfer, Animated Presentations, Instant Messaging and Portability. Adobe claims that their Conferencing provides DVD quality and it also indicates who is speaking at any moment. File Transfer enables transferring and storing any kind of files. Animated Presentations include sharing data such as images, audio, video and also collaborate on whiteboards and polls. Instant Messaging allows interaction between the meeting participants but it also integrates with users using the Microsoft Live Communications Server or Microsoft Office Communications Server. Adobe Connect is available in Web Browsers that have Adobe Flash-Plug-in installed, in Mobile Devices such as Android, the Blackberry Playbook and iOS and in teleconferencing systems that support SIP and H.264. H.264 is a video codec that is considered a standard for video compression.

2.1.1.2. BlueJeans

BlueJeans [8] is a video conferencing service created with the purpose of providing interoperability between the maximum type of devices. Instead of offering a vast amount of functionalities, this product focuses on one of the biggest problems of collaboration services: interoperability. The service supports devices such as Room Systems, Desktop Systems, Desktop Video Software (Skype, Google Video Chat), Mobile Video Software (Skype, Google Video Chat), Multipoint Control Units, Web Browsers and normal Telephones.

2.1.1.3. Campfire

Campfire [9] is a tool created by 37signals to help teams achieve their goals. It possesses a real-time communication system using both written messages and voice messages. It is available in form of a web application and in devices that run the iOS operating system. The written messages are updated automatically without the need to refresh the page and this is only possible by using AJAX. This application only supports group communications and it does not let the user create a chat room without creating a group. In order to make voice calls possible, it is necessary that the one person creates a conference call and everyone is sent or asks for a number and a code. After this, they can join the call by calling the given number and authenticate themselves by using the given code. It is also possible to send files to the members of the group, know the status (Available, Busy or Away) of the members, check the chat and file history and invite people who do not belong to the group. This is extremely useful because people can invite clients or other third parties to participate in a discussion without being registered on the service. Finally, this tool provides a REST API that allows developers to enrich their chat rooms with content from external applications or feeds.

2.1.1.4. HipChat

HipChat [10] is another tool that was created by Atlassian and is intended to help teams collaborate on their tasks. It has desktop applications for the 3 major platforms (Windows, Linux and Mac OS) and mobile applications for the two biggest platforms (Android and iOS). The main focus of this application is instant messaging. It is possible to chat in group or privately, share files and invite people that are not registered on the system. Though, these external people who enter the conversation cannot see what was written before. To conclude, this tool also has the opportunity to integrate with external services using a REST API and with other messaging applications through XMPP/Jabber.

2.1.1.5. Flowdock

Flowdock [11] is an online collaboration tool that is available as a web application. Its main focus is the exchange of instant messages in group chat rooms. These rooms are called flows. Besides being able to upload files to the system, the users can integrate their flows with other platforms in order to facilitate the access to new information. It is possible to make Flowdock search for a certain keyword in Twitter or subscribe RSS feeds. There are several other platforms that can provide great value for this application. There are applications for Version Control like Git, SVN and Mercurial, Project Management tools like JIRA and Redmine and finally applications like Google Calendar that allows teams to keep track of everyone’s schedule.

2.1.1.6. FuzeBox

FuzeBox is tool dedicated to Collaboration and Video Conferencing [12]. This away, besides connecting up to 12 users in High Definition Video Conferences, it supports other features such as: File Transfer, Desktop or Application Sharing and Instant Messaging. One great innovation that this tool provides is having the system call you to join a certain meeting. File transfer allows participants to upload several different types of files in order to make rich presentations. It is also possible to show presentations without transferring files by using Desktop or Application Sharing. Finally, private or public chats with other participants are also available. If it is necessary, participants can also record meetings to watch them later.

This tool is available in Desktop Applications for Mac and PC, Mobile Applications for Android and iOS, in any major Web Browser and with Room Systems.

2.1.1.7. GoToMeeting

GoToMeeting [13] is a web conferencing software created by Citrix. The most important features of this software are: High Definition Video, Animated Presentations and Instant Messaging. This product includes HDFaces™ which enables video conferencing with up to 6 attendees with High Definition Video. These conferences are also available through IP Phones or normal telephones and can be recorded. Animated Presentations allow users to share screens, highlight parts of the screen, collaborate on whiteboards and provide remote control when necessary. Instant Messaging allows exchanging messages between the participants. GoToMeeting is available as a Desktop Application (Mac or PC) and as a free Mobile Application (Android and iOS).

2.1.1.8. Huddle

Huddle [14] is a collaboration tool that has a different approach from the three tools that we have seen before. Although it has the same purpose of helping teams in their work tasks, this tool does not favor the exchange of instant messages. There is a place where members can keep their discussions but it takes form of internet forums. Members can create topics and others can post their answers. Like every application that we have seen, it is possible to share files with the other members of the team. The differentiating functionality that this application has that is not available in the others is the possibility to create Word and Excel files to be edited by the team. Although this option is not as valuable as using Google Documents because it is not possible for people to edit the same file at the same time, it is highly better than downloading, making changes and uploading the file every time a change is made. To make this possible, a lock is created when a user is modifying the file.

This tool also allows its members to schedule meetings with audio conference but these meetings were programmed the same way as Campfire. After the creation of the meeting the invitees receive a notification with the number they should call and a certain code to authenticate themselves.

Finally, the last functionality that is worth mentioning is the ability to create tasks. It is extremely useful having a centralized place where people can create tasks and assign them to a certain member. These tasks have a start and finish date and the task list can be exported to a CSV file.

2.1.1.9. IBM Lotus Sametime

IBM Lotus Sametime is another collaboration solution that focuses on integrating data, video and audio on the same platform [15]. The most relevant features of this solution are: Enterprise Instant Messaging, Rich User Presence, Rich Presentations, File Transfer and Audio/Video Conferencing. Enterprise Instant Messaging has features such as chat history, rich text formatting and emoticons. Rich User Presence indicates the availability of the users as well as a custom status message. Rich Presentations allow sharing Applications or Desktop with other participants. File Transfer enables participants to share files between themselves. Audio and Video Conferencing allow users to communicate when they are on different locations. This tool allows participants to see who is talking at a certain moment, it allows moderators to lock rooms and eject participants. This tool is available for PC, Mac, Android, BlackBerry and iOS devices.

2.1.1.10. iMeet

iMeet is a collaboration tool created by PGi, short name for Premiere Global Services, Inc [16]. The most important features of this tool are: Desktop Video Conferencing, Instant Messaging, File Transfer and Social Networking. Video Conferencing allows conferences with up to 15 participants and has the functionality of having the tool call you by clicking a single button. Once it has been established, it is possible to view who is the active talker to avoid misunderstandings. The tool also allows inviting external clients that are not registered on the platform. Instant Messaging enables side conversations with one or more participants. File Transfer allows users to share files in the platform but it also lets users add new files by emailing them to the room. Social Networking shows more information about one participant and enables participants to connect and interact with each other on other platforms such as LinkedIn, Facebook, Twitter and etcetera. This tool is available for PC, Mac and iOS devices.

2.1.1.11. Microsoft Lync

Microsoft Lync [17] is a communication platform developed by Microsoft that is intended to the enterprise world. The main features include: Instant Messaging, User Presence, Conferencing, Enterprise Voice Capabilities and Portability. Instant Messaging includes features such as individual and group chat, message persistence, notifications and search. User Presence offers information about whether users are using mobile device or IP Phones, allows tagging to facilitate notifications and privacy settings. Lync Meetings is the codename for conferencing. The most important features that are available during conferencing are: High Definition or VGA Video, Desktop or Application Sharing, Whiteboard, Polling and File Transfer .This platform also attempts to maximize interoperability by supporting several different devices. These devices include: Room Systems, PC Applications, HTML5 compliant Web Browsers, Mobile Clients (Android, iOS and Windows Phone) and IP Phones. It also allows people to include Skype Users and normal Lync Users.

2.1.1.12. Saba Meeting

Saba Meeting is a Collaboration tool created by Saba Software [18]. The most important features are: Audio/Video Conferencing, Application and Desktop Sharing, Instant Messaging, Rich Presentations and File Transfer. Audio/Video Conferencing allows users to communicate with co-workers, clients and prospect clients. To smooth organization during meetings with several people, the system shows who is talking in a certain moment. Application and Desktop Sharing allows people to collaborate on tasks. Instant Messaging provides fast and easy communication. Rich Presentations include slideshow preview, whiteboards and real-time annotations. File Transfer allows users to share any type of files with other participants. This tool is available for PC, Mac, Linux, all major Web Browsers, and Android and iOS devices.

2.1.1.13. Spontania

Spontania, created by Dialcom, is a Collaboration tool that focuses on video conferencing [19]. Other important features that the tool includes are: User Presence, Instant Messaging, Application and Desktop Sharing, Rich Presentations, File Transfer and Session Recording. User Presence allows participants to determine who availability of other users. Instant Messaging provides a fast and easy way to communicate. Application and Desktop Sharing allows users to share their computer and also adds Remote Control to facilitate support. Rich Presentations include polls, whiteboards and allows users to notify the presenter that they would like to interrupt. File transfer allows participants to exchange any kind of files. Participants are also allowed to record sessions in two types of files: WMP and Spontania's proprietary file. The last feature is the opportunity of inviting external users to participate in any session. This tool is available for PC, Mac, Android and iOS and it also works with legacy Room Systems like Polycom.

2.1.1.14. Teambox

Teambox [20] is an online collaboration tool that is highly complete and comprises an appealing interface. Its main functionalities are group chat, project creation, discussions, tasks, notes and files. The group chat is still in development; therefore, it only has the content of the messages and the time when they were written. Although it is possible to create projects to optimize organization, group chats are not available for each of the projects. This way, project discussions are made, by now, in the form of forums. Because each task possesses a start and finish date, *Teambox*, creates a Gantt Diagram to assist members in keeping track of the progress of the project. Each user also has a separate tab where he can identify which tasks were assigned to himself or herself and if they are finished or not. Finally, it is also possible to update the numbers of hours that have been spent in a certain task.

Teambox also has integration with other platforms. The most valuable are Dropbox and Google Documents. The integration with Dropbox is extremely useful because every file that is uploaded to the platform is sent to the team folder. Due to the fact that Google Documents provides one of the best live editing experiences, it makes sense that the application tries to take advantage of this. What happens is that a member can create a document directly from the web site and the link to that document is stored under the files directory.

2.1.1.15. Voxeet

Voxeet is a Web Conferencing tool created with the purpose of eliminating confusing conference calls [21]. The three most important features of this tool are: High Definition Audio, High Speaker Recognition and High Mobility. High Definition Audio is achieved by reducing noise and echo. High Speaker Recognition consists in informing the participants who is talking by showing a visual cue. This feature also allows participants to reposition others in certain places in order to take advantage of surround sound. High Mobility allows participants to transfer calls from one device to another. This tool has a Desktop Application for PC and Mobile Applications for Android and iOS.

2.1.1.16. WebEx

WebEx [22] is a collaboration tool created by Cisco. The main purpose of this tool is to help people inside organizations achieve their goals without the need to travel. Their motto is "Less travel, more green". The most relevant specialized product is denominated Collaborate Solutions and its main features are: Audio and Video Conferencing, Instant Messaging, File Transfer, Desktop or Application sharing and Portability. Audio and Video Conferencing allow enriched meetings with up to seven webcam feeds simultaneously. This product also differentiates itself from some others on call setup. While most of the others require users to call a certain number in order to join a conference, this one allows people to be invited automatically. Instant Messaging allows participants to check who is online and chat with those users. File transfer enables transferring files at any time and the tool also stores all files so they can be viewed later. Participants who are using the conferencing option also have the opportunity to share their desktop or a certain application with the other attendees. Finally, this tool increases its value by providing support for several systems. Besides being available in all Desktop Platforms (Linux, OS X and Windows), it has Mobile applications for: Android, BlackBerry and iOs.

2.1.2. Functionalities Comparison

Table 2 - Collaboration Service - Competitors Functionalities Comparison

	Adobe Connect	BlueJeans	Campfire	HipChat	Flowdock	FuzeBox	GoToMeeting	Huddle	IBM Lotus	iMeet	Lync	Saba Meeting	Spontania	TeamBox	Voxeet	WebEx
Audio Calls	✓	✓	✓	1 to 1	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
Video Calls	✓	✓	✗	1 to 1	✗	✓	✓	✗	✓	✓	✓	✓	✓	✗	✗	✓
Active Speaker	✓	✓	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✓	✓
Call Me	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓
Manage Surround	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
Record Calls	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓	✓	✓	✗	✓	✓
Instant Messaging	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓
IM History	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓
User Presence	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓
File Transfer	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓
Desktop Sharing	✓	✓	✗	✗	✗	✓	✓	✗	✓	✓	✓	✓	✓	✗	✗	✓
Remote Control	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗	✓	✓	✓	✗	✗	✓
Whiteboards	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗	✓	✓	✓	✗	✗	✓
Polls	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✓	✗	✗	✓
Tasks	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✓
Post by Email	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗

Guest Access	✗	✓	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✓	✗	✗	✓
Social Network	✗	✓	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗
Gantt Diagram	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
3rd Party Apps	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SMS	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Platforms	PC Mac Android BlackBerry iOS Room Systems	PC Mac Linux Web Browser Android, iOS Telephone Room Systems	Web Browser iOS	PC Mac Linux Firefox Chrome Safari Internet Explorer 8+ Android iOS	Mac Chrome Web App iOS Web Browser	Mac Web Browser Android iOS	PC Mac Android iOS	Android BlackBerry iOS	Android BlackBerry iOS Symbian Windows Mobile	PC Mac iOS	PC HTML5 Browsers Android iOS Windows Phone IP Phones Room Systems	PC Linux Mac Major Web Browsers Android iOS	PC Mac Android iOS Room Systems	Web Browser iOS Chrome Web App Gmail Web App	PC Android iOS	PC Mac Linux Android BlackBerry iOS

This comparison table illustrates the different functionalities that each tool provides. The analysis of this table leads to the conclusion that almost all tools focus their development in features such as Audio and Video conferences, Instant Messaging, User Presence and File Transfer. After providing the basic features they try to enrich their tools with smaller features that help create valuable differences. These features are: showing the active speaker in an audio or video call, share and control an Application or Desktop from another user, enabling guest access in order to collaborate with clients or prospect clients, collaborating on whiteboards and create polls to help make decisions. Moreover, companies try to add value to their products with totally distinctive features. This includes features such as managing the position of the participants in order to make use of surround sound, having the system call the participants in order to join a meeting, allowing users to participate in discussions via email, SMS notifications, Task creation and assignment and automatically draw Gantt Diagrams based on tasks to improve planning and organization. In terms of platforms, with few exceptions, almost all tools support the three most common types of devices: Desktop computers, Mobile devices and Web Browsers but only a few support room systems and IP phones. This list is very complete and it possesses some truly innovative functionalities. Ideally, it would be necessary to implement a great amount of these and create new ones in order to stand a chance of entering this market. However, because this is a proof of concept created by just one person and the methodologies and learning are more important than the final product, it won't be possible to develop a product as rich as WebEx or Adobe Connect.

2.1.3. Market Analysis

In order to understand who is most successful in the corporate collaboration market, a high level market analysis was conducted by the author. With more resources it would be possible to analyze all competitors in terms of strategy, innovation and financial results but this path would consume too much time. This way, the analysis consisted in three parameters: Web Site Ranking, Gartner distinction and User review. The Web Site Ranking will allow determining the amount of visitors that each tool has on their website and it will be done using Alexa [23]. There are some tools that will not be classified because Alexa cannot determine page views if the website does not possess its own domain. Gartner is a well-known research company that focuses their work on information technology and attempts to select the best companies in certain market sectors. The sector that was chosen is Web Conferencing [24]. It is extremely difficult to find a valuable method to review tools that are bought in a traditional and enterprise way. This way, the solution found was including the average rating and the number of reviews in the mobile application stores.

Table 3 - Collaboration Service - Competitors Market Analysis

	Alexa Rank	Gartner	Android	BlackBerry	iOS
Adobe Connect	✗	✓	4.4 (891)	3.0 (48)	3.5 (295)
BlueJeans	220,209	✗	✗	✗	✗
Campfire	19,965	✗	3.8 (17)	✗	3.0 (123)
HipChat	32,426	✗	2.7 (229)	✗	3.0 (75)
Flowdock	68,280	✗	✗	✗	3.5 (9)
FuzeBox	123,710	✓	3.7(332)	✗	2.5 (1,345)
GoToMeeting	631	✓	2.3 (4,008)	✗	3.5 (865)
Huddle	✗	✗	✗	2.5 (2)	4.5 (59)
IBM Lotus Sametime	✗	✓	3.4 (289)	3.0 (3)	3.5 (90)
iMeet	490,399	✗	✗	✗	4.0 (35)
Microsoft Lync	✗	✓	3.4 (1,932)	✗	3.5 (329)
Saba Meeting	251,329	✓	5.0 (12)	✗	0 reviews
Spontania	4,572,912	✓	4.2 (6)	✗	0 reviews
Teambox	13,186	✗	✗	✗	4.0 (105)
Voxeet	980,233	✗	4.1 (29)	✗	4.5 (16)
WebEx	1,926	✓	4.2 (2,148)	4 (55)	3.0 (6,319)

2.2. Platform Design

This chapter describes the steps that were necessary in order to design the whole Collaboration Service.

2.2.1. Analysis of Requirements

This first section will present the requirements of the collaboration Service. This list was based on the first internship proposal, the state of the art of the competitors and the meetings that occurred with the Product Owner and the Scrum Master during the internship.

2.2.1.1. Functional Requirements

There are three types of requirements in this section: User requirements, Client requirements and System requirements. User requirements specify actions that the user executes and Client requirements specify actions that the client must execute as a consequence of User actions. System requirements identify the system's responsibilities in order to make every user requirement possible.

2.2.1.1.1. User Requirements

In most cases, each user will be required to create an account before using the system. The only exception will be when an external user is asked to join a chat group or a conference call. This way, User Management requirements include all functionalities that are related to the accounts that most users create.

Table 4 – Collaboration Service - User Management Requirements

Id	Title	Description	Priority
CS_FR_UM_01	Create User Accounts	Users must be able to create new accounts.	Critical
CS_FR_UM_02	Edit User Account	Users must be able to edit the details of their accounts.	High
CS_FR_UM_03	Delete User Account	Users must be able to delete their account.	Critical
CS_FR_UM_04	User Authentication	Users must be able to authenticate themselves after having completed the registration process in the past.	Critical
CS_FR_UM_05	User Session Termination	Users must be able to terminate their session after authentication has occurred.	Critical
CS_FR_UM_06	User Presence	Users must be able to communicate their availability (Available, Busy, Offline) to the system.	Critical
CS_FR_UM_07	Contact List	Users must be able to keep a contact list. This list should include all the employees of the organization.	High
CS_FR_UM_08	Receive Notifications	Users should be notified when they are not online.	Low

Project Management requirements allow users to manage a certain Project. These projects possess a list of people who are working on that project and they also provide ways to keep work organized. This organization is made by separating chats that are associated with the project and also through the usage of tasks. Tasks allow users to delegate work to another user and have a start date and when it should be finished. When a task is finished the person who did it should change its status to complete.

Table 5 - Collaboration Service - Project Management Requirements

Id	Title	Description	Priority
CS_FR_PM_01	Add Project	Users must be able to add new projects.	Medium
CS_FR_PM_02	Edit Project	Users must be able to edit project details.	Medium
CS_FR_PM_03	Manage Project	Users must be able to manage project members.	Medium
CS_FR_PM_04	Remove Project	Users must be able to remove projects.	Medium
CS_FR_PM_05	List Projects	Users must be able to access a list with all the projects in which he or she is participating.	Medium
CS_FR_PM_06	Add new Task	Users must be able to add new tasks to a certain project.	Medium
CS_FR_PM_07	Assign Task	Users must be able to assign tasks to a certain user.	Medium
CS_FR_PM_08	Complete Task	Users must be able to set tasks as complete.	Medium
CS_FR_PM_09	Delete Task	Users must be able to delete tasks.	Medium
CS_FR_PM_10	List User Tasks	Users must be able to list their tasks.	Medium
CS_FR_PM_11	List Project Tasks	Users must be able to see a list of tasks assigned to a project.	Medium
CS_FR_PM_12	Download Information	Users must be able to download tasks to be visualized while offline.	Low

Instant Messaging Requirements include all the requirements that are related to Instant Messaging. Instant Messaging allows people to have discussions in real time.

Table 6 - Collaboration Service - Instant Messaging Requirements

Id	Title	Description	Priority
CS_FR_IM_01	One to one IM	Users must be able to have private chats with another user.	Critical
CS_FR_IM_02	Group IM	Users must be able to group chat with a group of users of their choice.	Critical
CS_FR_IM_03	IM History	Users must be able to view the messages that were sent in previous conversations.	Critical
CS_FR_IM_04	Import from email	Users must be able to forward an email to the service in order to include previous discussions in a new chat.	Medium
CS_FR_IM_05	Invite External Users	Users must be able to invite external Users (e.g. Clients) to participate in a chat conversation.	High

File Transfer requirements allow users to share files with other participants.

Table 7 - Collaboration Service - File Transfer Requirements

Id	Title	Description	Priority
CS_FR_FT_01	Private File Transfer	Users must be able to send files privately to a certain user.	Critical
CS_FR_FT_02	File Transfer	Users must be able to send files in any of the chats that exist. These files will be associated with the chat and with the project if applicable.	Critical
CS_FR_FT_03	File History	Users must be able to download files that were uploaded in the past or when they were offline.	High

Conferencing requirements include all the requirements that are connected to the possibility of creating audio and video conferences.

Table 8 - Collaboration Service - Conferencing Requirements

Id	Title	Description	Priority
CS_FR_CR_01	One to one Audio Call	Users must be able to make private audio calls with another user.	Critical
CS_FR_CR_02	Audio Conference Call	Users must be able to make audio conference calls with a group of users of their choice.	High
CS_FR_CR_03	One to one Video Call	Users must be able to make private video calls with another user.	Critical
CS_FR_CR_04	Video Conference Call	Users must be able to make video conference calls with a group of users of their choice.	High
CS_FR_CR_05	Invite External Users	Users must be able to invite external users to participte in conferences.	Medium
CS_FR_CR_06	Download Conferences	Users must be able to download conferences that were held in the past.	Low

2.2.1.1.2. Client Requirements

Client requirements are related to the all the responsibilities that the Web Client will need to execute in order to make everything work as expected.

Table 9 - Collaboration Service - Client Requirements

Id	Title	Description	Priority
CS_FR_C_01	SIP Registration	The Web Client must register the user in IMS after a registration or an authentication is done.	Critical
CS_FR_C_02	SIP Deregistration	The Web Client must deregister the user in IMS before terminating or when the user requires a session termination.	Critical

2.2.1.1.3. System Requirements

As it was mentioned in the beginning of this chapter, System requirements describe all the responsibilities that the System has in order to make everything work as expected. These requirements describe actions that are to be executed as a consequence of User actions or Client actions. They are divided in IMS requirements, Gateway requirements and Application Server requirements.

Table 10 - Collaboration Service - IMS Requirements

Id	Title	Description	Priority
CS_FR_IMS_01	SIP Registration	The IMS core must be able to receive and process the register requests from the clients and challenge them to complete registration.	Critical
CS_FR_IMS_02	SIP Deregistration	The IMS core must be able to receive and process the unregister requests from the clients and challenge them to complete unregistration.	Critical
CS_FR_IMS_03	Initial Filter Criteria	The IMS core must be able to interpret the requests made by the clients in order to redirect these requests to the correct Application Servers.	Critical

Table 11 – Collaboration Service - Gateway Requirements

Id	Title	Description	Priority
CS_FR_WS_01	Back To Back User Agent	The Gateway Server that serves the Web users must be able to mediate every request and response sent to or from the Web Browsers.	Critical
CS_FR_WS_02	HSS Provisioning	System must be able to provision new users in the HSS database when they register their accounts.	Critical
CS_FR_WS_03	Delete HSS information	System must be able to delete users in the HSS when they delete their accounts.	Medium

Table 12 - Collaboration Service - Application Server Requirements

Id	Title	Description	Priority
CS_FR_AS_01	SIP Compliance	The Application Server must be able to interpret SIP messages in order to execute all the necessary business logic and store all the necessary data.	Critical
CS_FR_AS_02	User Information	The Application Server must be accessible to persist any kind of information related to the user that is not related to SIP.	Critical

2.2.1.2. Non-functional Requirements

Non-functional requirements describe how the system should be built in order to ensure performance, scalability, security, usability and robustness. These requirements are important because they provide guarantees that the system is feasible to implement and that it possesses a great building block in order to make scalability possible.

2.2.1.2.1. Performance

Performance requirements are highly important because they are one of the decisive factors to achieve clients' satisfaction.

Table 13 - Collaboration Service - Performance Requirements

Id	Title	Description	Priority
CS_NFR_P_01	Response Time	For most of the actions, the response time should not be larger than one second. For longer actions like login and history fetching, a progress bar must be shown.	High
CS_NFR_P_02	Memory Consumption	The memory consumption should always be kept low on all the system's components.	Critical

2.2.1.2.2. Scalability

Scalability is defined in two different ways: vertically and horizontally. In order to scale vertically it is required to increase the capacity of the machines available. In order to scale horizontally, it is necessary to increase the number of machines to distribute the available work. The vertical scalability is easy to achieve but has greater costs while the horizontal is much more difficult but brings lesser costs. One of the challenges that horizontal scalability brings is how the association between users and their data is created. If there is a centralized database the horizontal scalability will eventually create a serious bottleneck. If the database is distributed it has to be ensured that the user will always be served by the same machine. To achieve this, it is required to have some kind of load balancer that decides which machine serves each user.

2.2.1.2.3. Security

Security is possibly the most important non-functional requirement of this application. This happens because the content that will be shared inside the application may be very sensitive and extremely valuable to the organization. That said, the following set of requirements will define what is necessary to implement in order to ensure that all the communications are secure and cannot be exploited by third parties. Furthermore, all the stored data must only be accessible to the people who are supposed to access it.

Table 14 - Collaboration Service - Security Requirements

Id	Title	Description	Priority
CS_NFR_SEC_01	Client Secure Connection	The connection between the Web Browser and the Gateway Server must be encrypted to protect data transmission.	Critical
CS_NFR_SEC_02	Gateway Secure Connection	The connection between the Gateway Server and the IMS Core must be encrypted to protect data transmission.	Critical

CS_NFR_SEC_03	IMS Core Connection	The connection between the IMS Core and the Application Servers must be encrypted to protect data transmission.	Critical
CS_NFR_SEC_04	Session Credentials	All components must avoid storing sensitive credentials while user sessions are inactive.	Critical

Depending on the type of connection used the form of encryption will have to be different. When the architecture is described, more information will be provided on this matter.

2.2.1.2.4. Usability

Usability requirements are also important because they may also define the success rate of an application. Besides having an attractive design, all the available actions must be easily and intuitively accessible. Because it is difficult to quantify these requirements, during the development phase discussions will occur in order to take the best decisions possible. These discussions include implementation of asynchronous tasks in order to avoid blocking the User Interface, retry mechanisms to avoid repetitive tasks and correct illustration of errors.

2.2.1.2.5. Robustness

Robustness requirements are important in order to improve even more the usability of the system. In order to keep the user satisfied the system's availability will have to be larger than 99%. Furthermore, because the system will have several components, it is important that some or all of them implement retry mechanisms. This way, if any component fails during the execution of an action, it will be repeated without informing the user. Only if the number of retries reaches a pre-defined maximum amount will the system notify the user about the failure.

2.2.2. High-level Prototypes

In order to truly understand what the final application should be before the development phase starts, some high-level prototypes were created. These high-level prototypes allow stakeholders to sketch the ideas that they have about what the application should look like and how actions should be available to the end user. These prototypes are described in the Annex B.

2.2.3. Technologies

Researching technologies is another important aspect of the platform design phase. This research allows determining which technologies are more appropriated for each component that the system will comprise. Considering that this project deals with several different types of components, the research conducted is described in the Annex C. This document should be consulted whenever questions arise.

2.2.4. Architecture

This chapter describes the architecture of the Collaboration Service. Firstly, a high level architecture is presented with a brief description of the purpose of the main components. Then, each component will be described in terms of subcomponents and how they interact with each other.

2.2.4.1. High Level Architecture

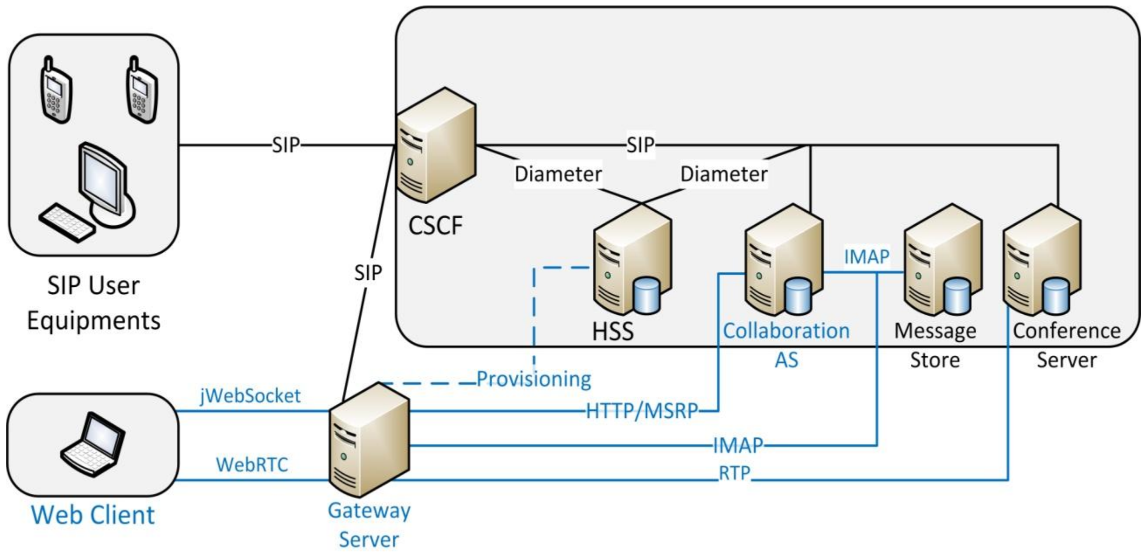


Figure 1 - Collaboration Service Architecture

The figure 1 illustrates the implemented architecture. The components that have a blue colour indicate that they will require contribution from the author.

The system has its core located on IMS. Its usage enables interoperability between different devices and it also provides an easy integration with other systems due to its highly specified architecture. This core is built using the open source project named OpenIMSCore which implements a CSCF and a HSS (see Annex A – Technical Background). Moreover, there are Application Servers that are responsible for executing all the necessary business logic upon the reception of client requests. The Conference Server enables audio and video conferencing with the use of WebRTC and RTP. The Collaboration AS is a central point that contains the majority of the functionalities. It is accessible through the IMS for everything related to SIP: chat messages, file transfers and user presence. It is also accessible from HTTPS in order to provide user account registrations, conference management, remote contact list and project and task management. Finally, there is a Message Store that is responsible for storing the users' messages and files that were exchanged on the platform.

Around the IMS there can be any type of device that uses SIP to connect to the IMS. Although the functionalities that are available might differ depending on the device in use, the opportunity to collaborate will always be present. The usual SIP clients such as VoIP phones, VoIP desktop applications or legacy Room Systems are already compatible with IMS but the same does not happen with Web Browsers. As it was mentioned during the analysis of the technologies (see Annex C – Technologies), it is mandatory to have a component capable of linking the Web Browsers to the IMS Network. This component can be seen as a Gateway Server that is also responsible for serving the Web Clients on tasks they cannot perform as a result of their inability to work with protocols different than HTTP.

In a real IMS environment, all the data that is exchanged between the Gateway Server and the CSCF and the Application Servers, should be exchanged with a component named Session Border Controller (SBC). This component should be the only point of entry of the IMS Network in order to have more control on what enters the network. However, this component was not used on this internship because it would introduce unnecessary complexity.

2.2.4.2. IMS Core

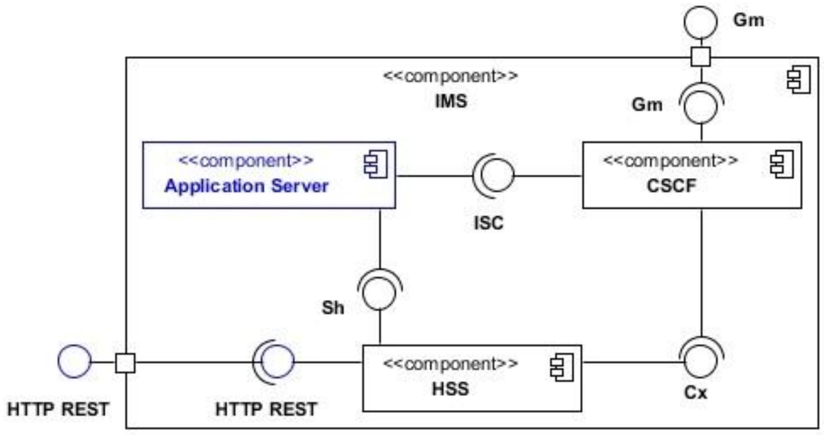


Figure 2 - OpenIMSCore Component Diagram

The Figure above describes all the subcomponents that compose OpenIMSCore. The ISC, Sh and Cx interfaces enable communication between the several subcomponents of IMS. However, there are two external interfaces that are also required. The Gm interface is the standard for incoming and outgoing SIP from the P-CSCF but it is also necessary to create a new interface that enables direct access to the HSS. This direct access to the HSS is required because it is necessary to register users before they attempt to interact with the system. This process of registering users for the first time in the HSS can also be called provisioning. The section below describes the research that was made in order to decide how the provisioning should be implemented.

2.2.4.3. Gateway Server

The Gateway Server is in charge of assisting the Web Clients in tasks they cannot perform by themselves given their inability to work with protocols different from HTTP. This fact makes this component extremely valuable because it can be seen as a complement of the Web Client. Without it, it would be way more difficult to achieve interoperability with IMS Networks. Apart from all the SIP/WebSocket negotiation that will be relayed by the Gateway Server between the Web Client and the OpenIMS Network, there are some heavy load operations that are also performed by the Gateway Server. To be exact, the chat Session using MSRP over TLS and the Audio/Video session using SRTP requires that the Gateway Server establishes a connection between the user and its destination and then relay the data between the two endpoints. The fact that it stands between two users indicates that this component requires high resources and may become a bottleneck when serving a considerable amount of users. To assure confidentiality and authenticity the SIP connections between the Gateway and the OpenIMS should be done over TLS and the connection between the Gateway and the Web Clients should use WebSocket Secure (WSS).

As it was mentioned during the research of the technologies the Gateway Server is WCAS, the Application Server developed at WIT that can also be used to serve HTTP Clients.

2.2.4.3.1. WCAS Architecture

WCAS is a converged server built with the aim of supporting the maximum amount of clients with several different protocols. At the moment it supports Flash Clients using RTMP, HTTP Clients, SIP Clients and WebSocket clients. This section describes how WCAS was designed to allow the support of these protocols and to be flexible in terms of configuration and mode of operation.

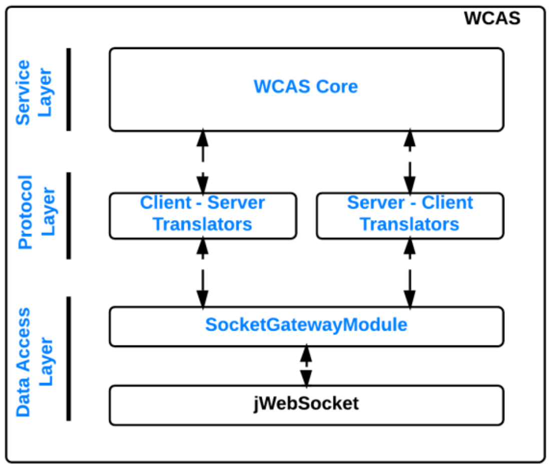


Figure 3 - Collaboration Service - WCAS Architecture

The figure above shows that this component is separated in three different layers: the Data Access Layer, responsible for establishing the connection with the several supported protocols; the Protocol Layer, in charge of translating the content received to a uniform request that the upper layer can understand and responsible for translating the content that comes from the upper layer to the specific protocol; the Service Layer, which comprises all the business logic of the server.

Data Access Layer

This layer comprises the necessary libraries that the several protocols require. Just like the SIP Clients oblige the use of SIP Servlets and a SIP container to be able to communicate, the WebSocket Clients oblige the use of the jWebSocket framework on the server. Furthermore, the Socket Gateway Module is responsible for invoking the required translators for the incoming requests and calling the outgoing translators before the responses are sent.

Protocol Layer

The Protocol Layer contains the aforementioned translators. These translators can be Server Translators and Client Translators. The Server Translators are necessary to construct the Uniform Requests that will be sent to the Service Layer. For example, it might be needed to decode a Base64 String into a Java byte array, or a JavaScript Object into a Java HashMap. The Client translators are used to translate outgoing requests to the specific protocol that the client uses. Both types of translators possess methods capable of translating the responses made to the initial request.

Service Layer

The Service Layer contains all the server logic that enables the features available. This logic is separated into Signalling Delegates and Managers. In order to enable different modes of operation, the core of the server is configured using the Spring framework. With this framework, the configuration is done on XML files which are read before the initialization of the server components. For example, when WCAS is operating as a Gateway Server, it should load a proxy signalling delegate capable of forwarding all the requests from the users to another server. On this project, the OpenIMS is the target server. When WCAS is operating as an Application Server, it will load a signalling delegate that acts as if all the clients that reach the AS through the S-CSCF are local clients.

2.2.4.4. Web Browser Client

The Web Browser Client is the component that is responsible for establishing the connection between the Web Users and the rest of the system. This component communicates with the Gateway Server using the jWebSocket framework for most of the functionalities and uses WebRTC to be able to communicate using audio and video calls. It is extremely important to state that this should not be seen as a traditional web site where user navigation results in HTTP requests and responses. This Client should be seen as a fat Web Client that downloads all the Javascript logic on the first access and then it is capable of interacting with the server without the need to refresh the page.

Once more, the architecture of this component is divided in layers with the aim of providing flexibility. This flexibility favours the creation of different clients using the same underlying structure. The figure below describes the several layers that are part of this component.

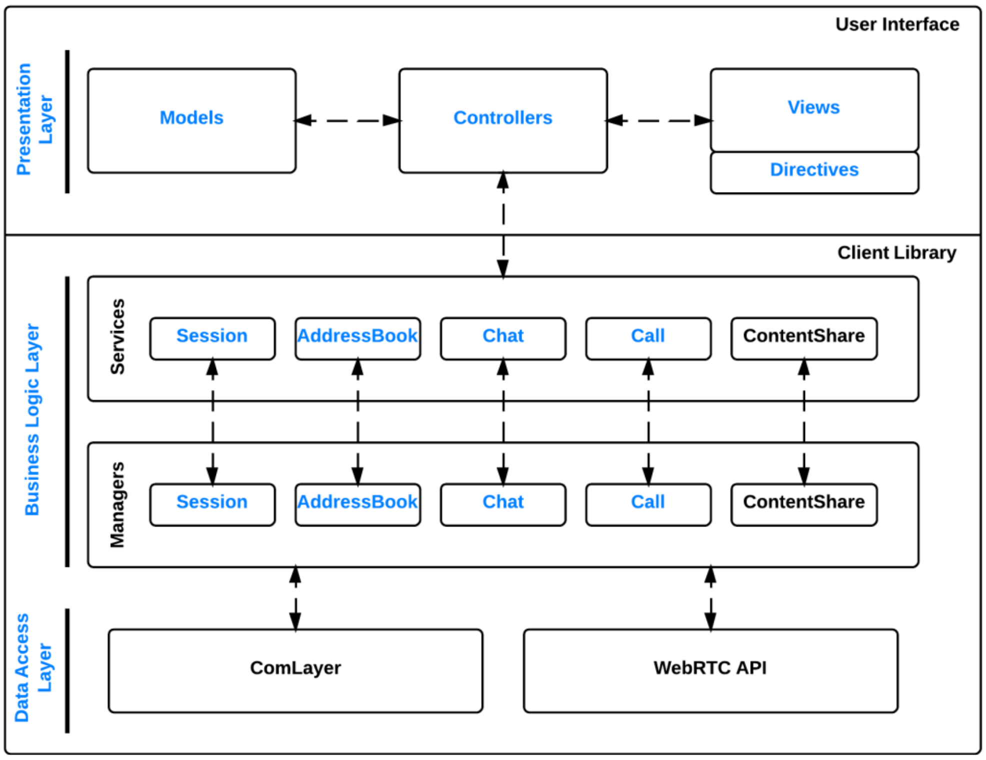


Figure 4 - Collaboration Service - Web Browser Client Architecture

The figure above indicates that there is a clear separation between the UI and the Javascript Library. Additionally, the architecture of the Web Browser Client is divided in three layers: the Presentation Layer, the Business Logic Layer and the Data Access Layer. The Presentation Layer is developed using AngularJS and it is in charge of managing the interaction with the user and presenting the data that comes from lower layers. The Business Logic Layer is comprised by Services and Managers that together can be viewed as Modules for each of the features available. Finally, the Data Access Layer is composed by the ComLayer and the WebRTC API which are responsible for establishing connections with the desired endpoints.

Client Library

Similarly to what happened with other components of this internship, the use of asynchronous calls was highly preferred given its ability to avoid blocking the User Interface while the Client Library is working. However, this requires the existence of event subscription and promises. The subscription of events allows maintaining a list of entities that are interested in a certain type of resource. This way, when newly arrived content matches one of those resource types, all the entities that subscribed that event are notified. For example, when the Call Manager receives a call invite it will process that invite and notify the Call Service. In its turn, the Call Service will iterate through the list of controllers that subscribed the call invite event and notify those controllers. Promises are, as the name suggests, something that will be executed in the future. These promises are extremely useful on the Presentation Layer. Almost all public methods exposed by the Services return a promise to the Controllers. When the Service receives the answer it will mark the promise as successful or unsuccessful and the controller is informed.

Business Logic Layer

The Business Logic Layer possesses several modules that are deeply connected with all the features available. Each module is separated into a Service and a Manager. The Services are responsible for managing the subscription of events, avoiding a direct access between the User Interface and the Managers by exposing a public API and storing all the data that will be accessed by the Presentation Layer. The Managers are in charge of interacting with the Data Access Layer and storing data that enables the communication with the lower layer. For example, the SIP headers that allow identifying a single session must be kept in memory while the session is active. Another concept that is important on this layer is the creation of Commands. These commands are identified by a name and are used to communicate with the Gateway. For instance, to create a session the SessionManager creates a command named SESSION_CREATE that is sent to the server through the ComLayer. When this command reaches the Socket Gateway Module on the server, it will invoke the respective Server translator to create a Uniform Request.

The table below describes all the Modules available.

Name	Description
SessionModule	Module responsible for initiating the interaction with the Gateway Server. It allows creating and terminating User Sessions.
AddressBookModule	Module accountable for the contact list and the project list. Possesses all the methods necessary to manage these two entities.
ChatModule	Module in charge of the exchange of instant messages privately or inside projects. Also responsible for the delivery, display and composing notifications and for the subscription of the state of project chats.
CallModule	Module responsible for establishing and terminating the calls that the users make. It also manages the subscription of the state of conference calls.
ContentShareModule	Module that makes possible sharing files between users. It comprises all the logic that splits or joins the file chunks and encodes and decodes the Base64 data.
CapabilitiesModule	Module that makes possible sending and receiving the presence of the user as well as the capabilities that the client possesses.
MessageStoreModule	Module responsible for interacting with the server that possesses the chats and files that were shared on the platform.

Figure 5 - Collaboration Service - Client Library - Modules Description

Data Access Layer

The Data Access Layer is composed by the ComLayer and the WebRTC API. The ComLayer is responsible for instantiating the jWebSocket framework. The reason behind the creation of this object is the fact that the jWebSocket framework does not support several responses to the same request. Considering that the SIP protocol is rich in provisional answers it was necessary to create this class to be capable of matching the several responses with the initial request. The WebRTC API can be viewed as all the functionalities that the supported browsers provide in order to establish audio and video calls.

User Interface

The User Interface is responsible for the interaction with the user. This component is developed using the AngularJS framework which is normally associated with the Model-View-Controller Design Pattern. This framework is perfect for the type of client that is being developed given its ability to synchronize the Model and the View automatically. This synchronization is achieved with a technique named two-way data binding. What happens is that all the views that are connected to one controller are also connected with the Models that are associated with that controller. This way, the changes done in the Model affect the View and the user input inside one view also changes the value inside the Model.

This framework also provides directives which facilitate the programmer's job on converting some model types to HTML. For example, a directive was created to convert timestamps to a specific time format and another one was created to scroll to the bottom of a chat conversation when necessary.

The figure below describes the steps executed during the registration of the user in a sequence diagram. These steps make clear how the components are connected. However, in order to avoid increasing complexity, the Services and the Managers were merged into Modules.

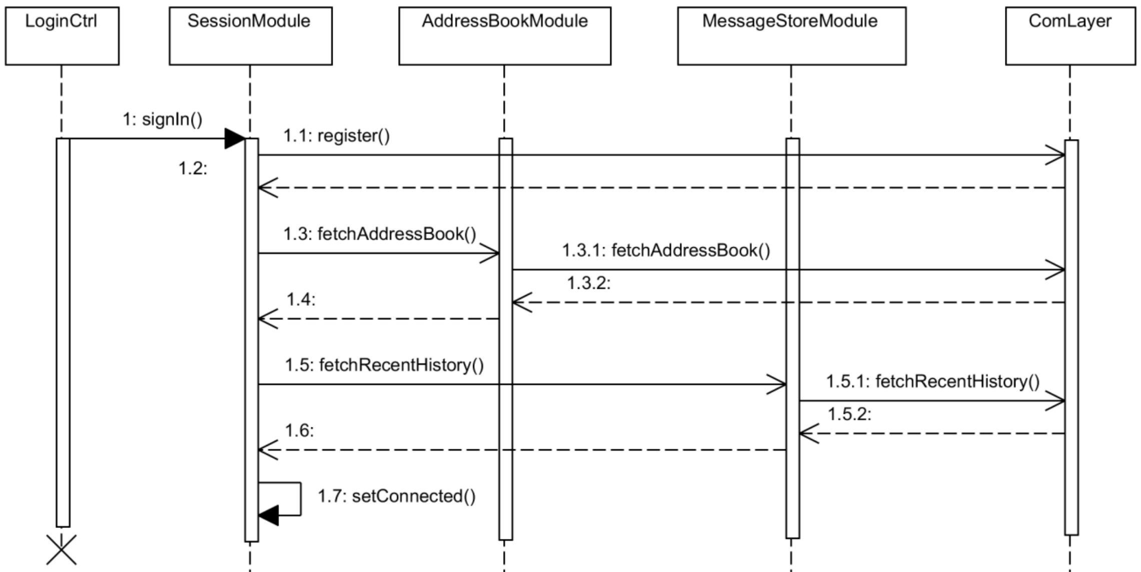


Figure 6 - Collaboration Service - User Interface flow for Login

This sequence diagram presents a sample flow of execution but the last step of setting the user as connected introduces two concepts that are important to understand: watchers and digests. The AngularJS framework provides a way of creating watchers of certain Model variables. These watchers are useful to be able to determine if certain changes occurred. Each watcher receives two functions as arguments. The first one compares the current value of the variable with the value that the variable had on the previous call. The second function contains the code that should be executed when the current value is different from the previous value. For example, the component that is responsible for managing the views of the application watches the connected variable that is present on the SessionService. This way, when the SessionService inside the SessionModule is notified that all the needed content was fetched, it sets the user as connected. However, in order to inform the watchers of the change it is necessary to call a method name digest. What this method does is triggering all the compare functions that are passed to the watchers. If there are any changes on the watched variable the second function is executed.

2.2.4.5. Collaboration AS

The Collaboration AS is the core component responsible for the business logic of almost all the features available. Besides being accessible through SIP in order to coordinate functionalities such as chats, user presence and file transfers, this Application Server is accessible using HTTP/XCAP to enable contact list and project list, it exposes a Web Service in order to enable account registration and it is responsible for persisting the data exchanged on the service to the Message Store. To accomplish this persistence there has to be a shift in what is a traditional setup and creation of a media session such as a chat or a file transfer. Normally, media sessions are established with the help of the IMS Network but once they are established the media is sent directly between the participants. However, if this route was followed it would be necessary to put the clients in charge of the persisting the desired messages. The alternative found was obliging the SIP Invites to pass through the Collaboration AS, as if this session was a conference with multiple participants. This way, each participant establishes a connection with the Application Server which is responsible for relaying the messages or the files between the participants and sending them to the Message Store.

If it was present on the requirements, this Application Server would provide the convergence between the IMS network and the XMPP network. This would require the capability of initiating different sessions with the two protocols and associating them with each other as it is made with WebSockets and MSRP.

As it was also mentioned during the research of the technologies, the Collaboration AS will be built on top of another instance of WCAS. Although it has the same name, the server is configured to be inside an IMS Network. To understand how WCAS works please consult the section 2.2.4.3.1 - WCAS Architecture.

Database

Besides the information that is stored in the Home Subscriber Server, it was decided that it would be necessary to have another database that contained all the information that was not related with the IMS core. This approach allows us to decrease the access load on the HSS and makes it easier to create an API that can be used by third parties. Examples of these third parties are Conference Servers or Instant Messaging Servers. Because the Entity Relationship would be illegible inside this document, only the main classes are described.

Subscribers - The SUBSCRIBER table contains information about the users registered on the system.

Contacts – The CONTACT table stores personal and information about the contacts in the subscriber's contact list.

Projects – The PROJECT table stores information about a certain project and which subscribers belong to that project.

2.2.4.6. Message Store

The Message Store is a crucial component that enables the persistance of instant messages and files exchanged on the platform. During the initial plan of this internship, this server was supposed to be present in another internship and the only development that would be necessary was the integration between the client and the Message Store. However, given that this was an extremely useful component that provoked a big differentiation from what already exists, it was decided that the intern would develop what was necessary to store the messages and files and also to fetch them.

Considering that this component is not available at the company it was also decided that the implementation should be able to integrate with the existing servers and clients that the company possesses. That said, the most correct way to do this was following the RCS 5.1 specification [26] along with the OMA CPM Message Store [27]. These two specifications state that the Message Store should be implemented with the use of a server capable of using the IMAP protocol [28].

As the High-level architecture showed, this server is accessible by the Collaboration AS and the Gateway Server. The Collaboration AS is responsible for storing all the media that passes by this component while the Gateway Server is responsible for fetching the content when the Web Clients request it. Both these interactions with the Message Store Server should be done over TLS.

2.2.4.7. Conference Server

During the first semester the research conducted on Conference Servers led to the conclusion that the best option was using the MCU Media Server. However, when the implementation phase started it was decided that another route would be more challenging and more valuable. Considering that another intern at the company was developing a conference server to enable audio and video conference calls using SIP it made sense to interlink the two internships and produce something worthy.

This Conference Server can be seen as another Application Server that is reachable through the OpenIMS using SIP and SRTP. The SIP protocol is used to negotiate the terms of the media session and then the media is passed using SRTP. This Conference Server is built on top of JAIN SLEE, one of the researched Application Servers, and Mobicents Media Server.

2.3. Implementation

Considering that the author of this document worked with several different existing clients and libraries in order to develop this internship, it is extremely important to make clear which components were developed by the intern and which decisions were necessary to accomplish the pre-defined objectives.

Throughout the development of this service, one of the biggest concerns was reaching maximum interoperability between devices. This way, there were several functionalities that required several hours of studying in order to make sure that the existing specifications were being followed correctly.

2.3.1. Web Client

The Web Client is the most important component of the system as it is responsible for interacting with the user. Consequently, this was the component that required more effort by the author. The development was done on all the layers but it focused primarily on the Presentation Layer and the Business Logic Layer. The Presentation Layer was entirely created by the author and so were the services present on the Business Logic Layer. Furthermore, it was also necessary to add or extend some of the Managers that are used by the Services.

2.3.1.1. Remote Configuration

One of the first tasks that were executed was the ability to work with WCAS when it is running as a Gateway Server. To achieve this, the Web Client must send to the server a set of parameters when it tries to create a new session. For example, it is necessary to inform that the contact list should be fetched remotely and pass to the server the URL of the remote server and the credentials that should be used to complete authentication. Or that the Instant Messaging will also be done remotely.

2.3.1.2. Project List

The project management was also implemented from scratch. It was decided that the AddressBook Module that contains the contact list would be extended to support the project management. This module now supports the creation, edition and deletion of projects.

2.3.1.3. Project Chat and Project Call

The possibility to chat and call a group of users inside a project was also developed by the author. These features required changes throughout the whole Client. In addition to the necessity of passing the identities of the participants to start the conference (chat or call), it is crucial that the Client is capable of subscribing its state and invite new users while it is already taking place (SIP Subscribe/Notify). That said, the ChatManager and CallManager were extended to pass the desired participants, subscribe a conference, parse the received updates about that conference and referring (SIP Refer) new participants.

2.3.1.4. Interaction with the Message Store Server

The integration with the Message Store with the help of the Gateway Server was also designed and built by the author. It was necessary to develop three types of interaction between the Web Client and the Gateway Server. Firstly, after fetching the contacts and the projects, the Client can fetch the most recent history that is stored on the Message Server. Secondly, when the user is reading through the messages and the files that were shared, it will reach the last fetched message. When this happens, the Client lets the user click a button to fetch more content. This content will always be related to the user or the project that is open. Finally, the last interaction is connected with the opportunity to fetch remote files that are stored on the platform. This route was followed because it did not make sense to pre-fetch large files that the user might not want to download. This way, the user can see that a certain file with a certain name was shared and if it is his or her desire, it is possible to fetch that file.

For the first type of interaction, the implementation on the client side is simple because no parameter has to be passed onto the Gateway but the same is not true to the interaction of fetching more content. In order to achieve maximum performance, the IMAP protocol was designed to facilitate the synchronization of different devices. To facilitate this synchronization, the messages stored on the server are associated with an increasing Unique Identifier Message Attribute. This identifier allows informing the server which content the client already possesses by sending the first and the last identifier. This way, when initial history is returned by the Message Store, these unique identifiers are parsed on the client side and are associated with the respective users or projects. Finally, the process of fetching a remote file also makes use of this unique identifier by sending the one that is associated with the desired file.

These interactions obliged the creation of several components. The MessageStoreManager on the Business Logic Layer was created to send the commands to the ComLayer and returning the content to the MessageStoreService. This service is responsible for parsing the content and persist it on the TimelineService. The TimelineService is different from the other services because it has no manager associated. Its purpose is holding the several events that are associated with a certain contact or a project with the aim of centralizing the content. By having it all centralized, the necessity of ordering the content after fetching it from different sources disappears.

2.3.2. Gateway Server

The capacity to serve Web Socket clients while in proxy mode had not been tested before the start of this internship. This way, it was necessary to implement several Client and Server Translators that enabled some core functionalities that were already present, as well as implementing translators for new functionalities. These translators are described on the table below.

Table 15 - Collaboration Service - Gateway Server Translators

Module	Description
AddressBook	Add, delete, edit, get and list project. The main effort on these translators is spent on the translation of the responses back to the client. They receive a List of contacts and create a set of JSON objects to be sent to the client.
Call	Extend Invite translators (Server and Client) to include the desired participants on a conference call. Implement Server Conference Referral translator to enable inviting participants to an ongoing call.
Chat	Extend Invite translators (Server and Client) to include the desired participants on a conference chat. Implement Server Conference Referral translator to enable inviting participants to an ongoing call.
Event	Implement Server Subscribe translator that enables the subscription of conference calls and conference chats.

MessageStore	Implement Server translator to fetch recent messages and files.

2.3.2.1. Fetching Content from Message Store

The Gateway Server is responsible for fetching the content from the Message Store so it can be delivered to the users. As it was mentioned on the description of the Web Client, there are three use cases that were necessary to implement: fetch recent history, fetch older history and fetch remote file. But the first step to enable these three use cases is the creation of the IMAP connection between the Gateway and the MessageStore. When a Web Client creates a session, it will pass the aforementioned parameters that define the configuration that should be used. If that configuration specifies the MessageStore that the user wants to use, the information is passed onto a class named MessageStoreManager. However, the connection to the IMAP Server is not created automatically. This Manager is also responsible for receiving the commands of the three use cases and for launching a new thread to fetch the desired content. The first step of this thread is establishing the connection with the server. After this, the execution depends on the use case that was requested.

On the first use case that fetches recent history, the Gateway iterates through the last messages that were persisted and analyses its headers to determine the content type. If the content type is textual, the content is fetched, if not, only the headers are returned to allow identifying the file.

The second and third use cases required a different approach. Considering that the Gateway receives the lower unique identifier from the client or the identifier of the message that contains a desired file, the first step is finding the message that matches that identifier. To speed up this process, it was decided that a binary search would be appropriated. After finding the searched message, the Gateway just needs to iterate backwardly to load previous content or fetch the desired file.

2.3.3. Collaboration AS

After understanding how the client requests the content and how the Gateway Server fetches it, the only thing left is understanding how the content shared on the platform ends up on the Message Store. The architecture chapter described how the session must be setup in order to put the Collaboration AS as the receiver and transmitter of the messages and files. Considering this, what was left to implement was finding an efficient way of receiving, relaying and storing the content. It is easy to understand that this task has to be executed asynchronously or the delivery of the messages would be delayed considerately. This way, when the content reaches the Collaboration AS, it has to be processed and kept in memory in order to be persisted later. The several messages and files that reach the server are associated with their session. Furthermore, since the files are sent in several MSRP chunks, the Collaboration AS is responsible for joining them before the persistence is made. When the chat session or the file transfer session terminates, this Server will construct and persist the messages on the server.

2.3.4. Web Services

The Web Services were developed with the aim of provisioning the several components of the system. From the Web Client's perspective, there is only one Web Service that is used to register the account but this Web Service is responsible for spreading the provisioning. The other components that require the provisioning are the Home Subscriber Server, for the communication using SIP, the Application Server, for the use of the contact list and the project management and finally the Message Store that holds the information about the messages and files.

The separation in different databases obliges extreme care in terms of database consistency. Ideally, it would be important to implement a two-phase commit algorithm that enabled the synchronization of the several databases before committing to the databases. However, what was implemented was a mechanism that rolls back the previously executed actions if one of them fails. The decision to take this route was based on a trade-off in terms of complexity and available time resources and it would certainly be something that would require revision when launching a product based on this system.

Finally, the author also implemented a Web Service capable of managing the creation, edition and termination of Conference Calls. This Web Service is used by the aforementioned Conference Server developed by another intern.

2.4. Quality Assurance

The design and the development of the platform are important phases of a project's lifecycle but the quality assurance is equally important. This chapter can be divided into three subsections: Unit Integration Testing, Acceptance Testing and Performance Testing.

2.4.1. Integration Testing

The Integration tests are normally executed by grouping individual units to be tested as a group. For example, during the development of the Web Service that enables account registration and provisioning of the other components, it was extremely important to confirm that the addition, edition and deletion of new information maintained the consistency of the databases. If, for example, the HSS provisioning failed after the user had already been stored on the Collaboration database it would be necessary to rollback the executed action. These tests were executed using the framework JUnit [32] and they helped find some edge cases that were not being accounted initially. Furthermore, given that the HSS provisioning is a complex process that involves the creation of several different identities (IMSU, IMPI, IMPU) it was necessary to attest if the creation of these identities would not perturb the expected behaviour of the OpenIMS framework. To accomplish this, the following tests were executed: user equipment registration, voice calls and navigating on the administration pages of the OpenIMS.

2.4.2. Acceptance Testing

The Acceptance tests assure that the application meets the requirements elicited during the design phase of the platform. In order to achieve this, it was necessary to create a test specification with test cases that could be executed with the features developed. Given its length, the test specification is present on the Annex E – Acceptance Testing. In addition to the Test specification, this document presents the report of the tests ran with that specification and the encountered issues.

2.4.3. Performance Testing

Performance testing helps determining how the system behaves under a certain workload and in certain conditions. The primary objective of the tests that were conducted was finding the maximum capacity of the system. In order to simulate a real environment it would be necessary to possess a cluster and a load balancer to distribute the workload but this could not be achieved due to time and resource limitations. Nevertheless, the tests were executed with the best resources available.

Before the start of the tests it was necessary to modify the Web Client developed for the Collaboration Service. Basically, the login procedure was changed so it could be possible to have several sessions running simultaneously but independently. Furthermore, most of the Presentation Layer was deleted because it did not make sense to show the test information and a new View was added to present some statistics of the tests. It was also decided that the tests would consist in sending chat messages and presence information between users and that the load would be increase incrementally. In order to assess how the system was behaving, a Python script was created to measure memory consumption and processor utilization. Finally, it was decided that the tests would be considered terminated when a certain number of sessions failed consecutively.

In order to find the maximum capacity of the system it was necessary to reconfigure the development environment to a production environment. The changes involved resetting the log level, increasing the maximum memory available, increasing the pools of threads and increasing the number of processors available. This was a lengthy process which involved numerous tests but in order to keep this document short only the final conclusions are presented. Furthermore, the several servers were spread, as much as possible, into several different computers to achieve better performance. The Gateway Server was installed on a machine with an Intel Core i7-3615 Quad-core Mobile @ 2.3GHz, 8 GB of RAM and running Windows 7 64bit. The OpenIMS, the Collaboration AS and the Conference Server were running on a Virtual Machine with the following specifications: 4 Processors Intel Xeon X3363 @ 2.83GHz, 4 GB of RAM and with Ubuntu 12.04. The Message Store ran on the machine provided by the company. This machine possesses the following specifications: Intel Core Duo P8400 @ 2.26GHz, 3 GB of RAM and Windows Vista 32-bit. Finally, it is important to state that the Web Browser that executed the tests was running on the machine of the Gateway Server.

The best scenario is characterized by the creation of ten different sessions every fifteen seconds and that the test would be considered as terminated when five sessions were denied or timed out. In terms of messages and presence status, it was decided that each session would send them randomly with intervals of at least three and five seconds. This scenario ran five times and the average results are described on the following table.

Table 16 – Collaboration Service – Performance Results

	Sessions	Sent Messages	Sent Presence	Failed Messages	Failed Presence
Average	619	100914	40852	4267	158
Worst	600	103553	37566	4924	160

The results show that it was possible to reach more than six hundred simultaneous clients sending an average bigger than one hundred thousand messages and forty thousand presence requests. The following figure describes the growth of the number of sessions of the worse case.

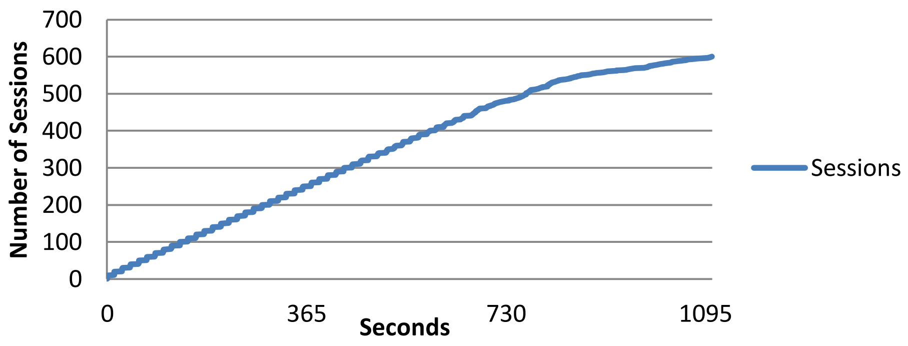


Figure 7 - Collaboration Service - Session Growth

As the figure above demonstrates, the growth of the sessions is steady for the majority of the test but it starts to slow down when the system reaches its maximum capacity. To figure out which component was causing the increase of the responses it was necessary to analyze the memory consumption and the processors' utilization.

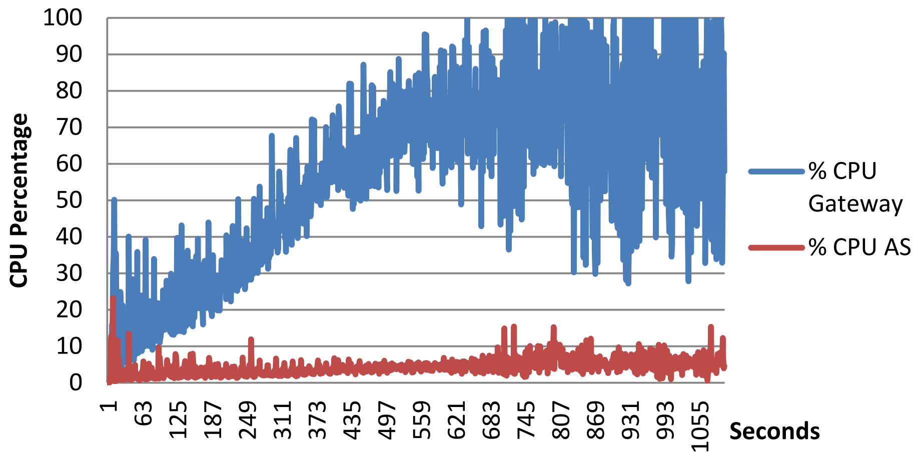


Figure 8 - Collaboration Service - CPU Utilization

The figure above clearly demonstrates that the bottleneck is located on the Gateway Server. There are several reasons that justify this bottleneck. Firstly, for each chat session between two participants, four connections have to be established on the Gateway Server. Two of them using jWebSocket to communicate with the clients and two more using MSRP to communicate with the Collaboration AS. Furthermore, the machine where the Gateway was running was also hosting the clients that were executing the tests.

The last figure that is important to show is the memory consumption. This figure shows that Gateway Server also consumes more memory but it does not reach the maximum that had been defined: 3072 MB. On the contrary, the Collaboration AS maintains a steady amount of memory consumption. It is important to remind the reader that these are average values taken from the five tests that were run.

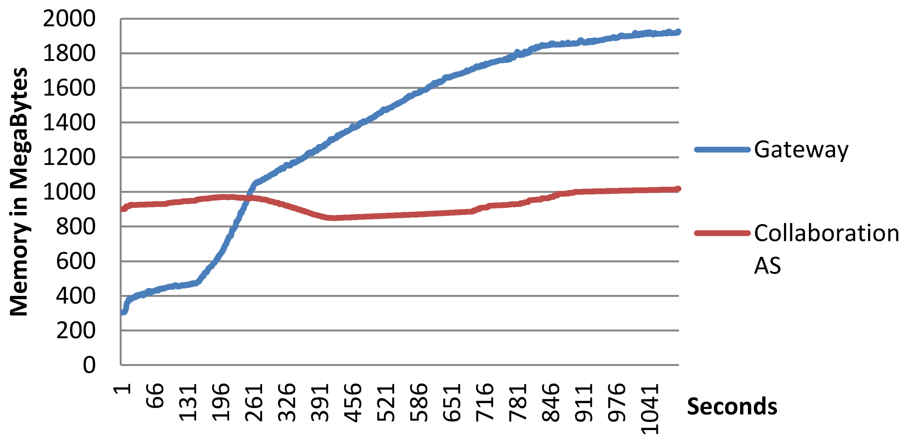


Figure 9 - Collaboration Service - Memory Consumption

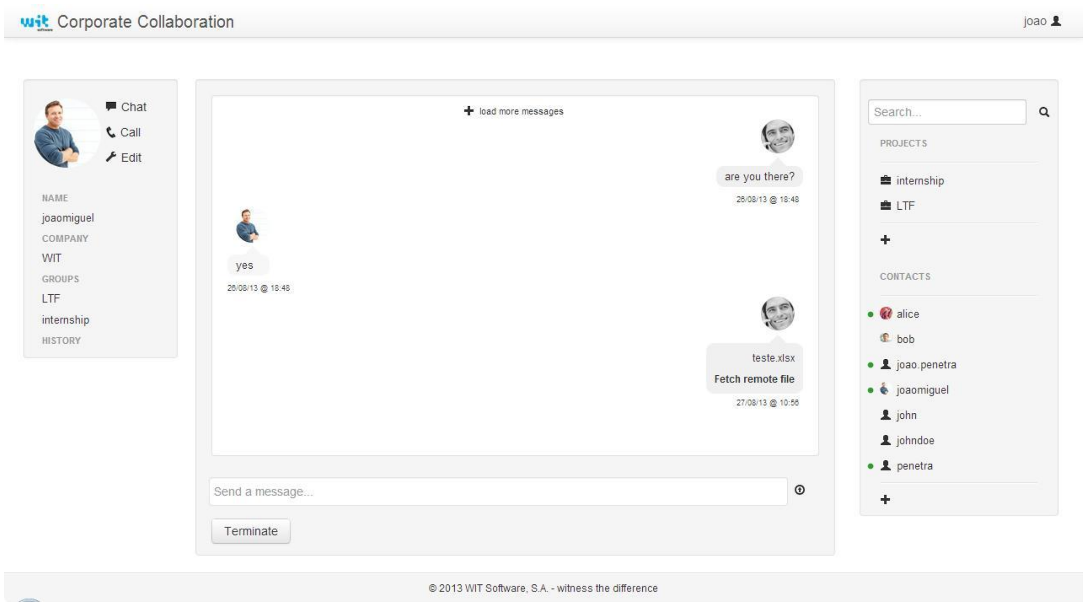
2.5. Work Accomplished

This section describes what was accomplished during the internship. Firstly, a table resumes what was done and then some screenshots are presented to illustrate the final product.

Table 17 - Collaboration Service – Work Accomplished

Features	Status
User Management	✓
Project Management	✓
Task Management	✗
One-to-One Chat	✓
Project Chat	✓
Chat History	✓
One-to-One File Transfer	✓
Project File Transfer	✗
One-to-One Audio Call	✓
One-to-One Video Call	partial
Project Call	✓
Project Video Call	✗

The development of the Task Management, the Project File Transfer and Video Call were scoped out of the project. The Task Management would have a structure similar to the contact list and project list but it had a lower priority. The Project Video Call was scoped out of the internship that developed the Conference Server on the second semester which made impossible the integration with this feature. Finally, the Project File Transfer was postponed because it is not supported yet by WCAS. At the time of writing, the One-to-One Video call was developed on the client side but the Gateway Server was not working correctly when relaying the video.



are you there?
26/08/13 @ 18:48

yes
26/08/13 @ 18:48

teste.xlsx
Fetch remote file
27/08/13 @ 10:56

Figure 10 - Collaboration Service - One to One Chat

The figure above shows the Contact Chat View on the left and the middle and the contact list and the project list on the right. The contact list possesses the contacts and their presence status. The left panel shows information about the user. Finally, the dashboard on the middle shows the messages and files that were shared on the platform. The last message indicates that a file was shared previously and that this file can be fetched.

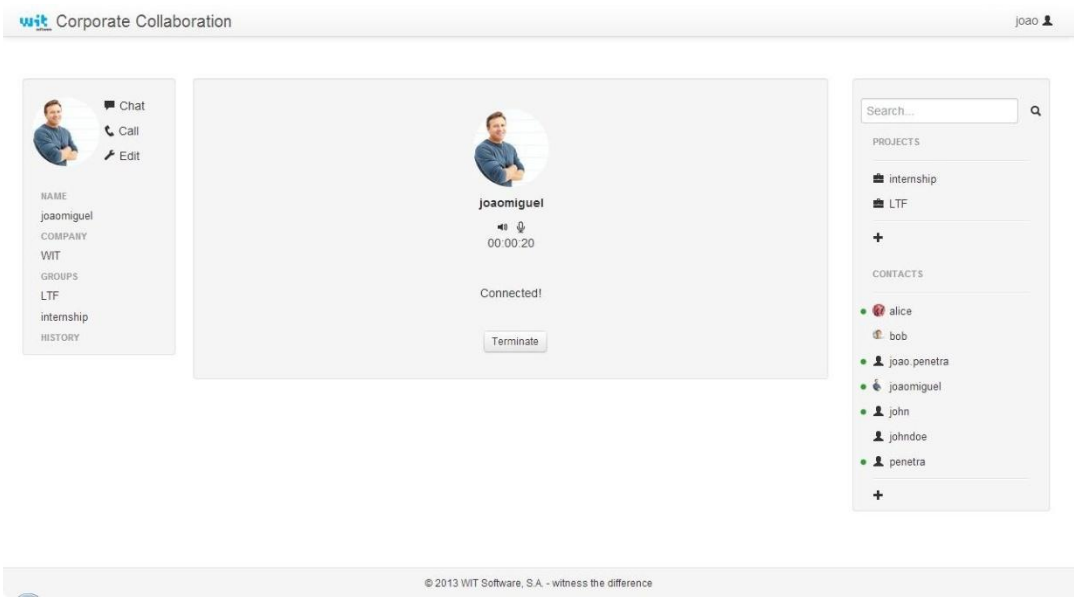
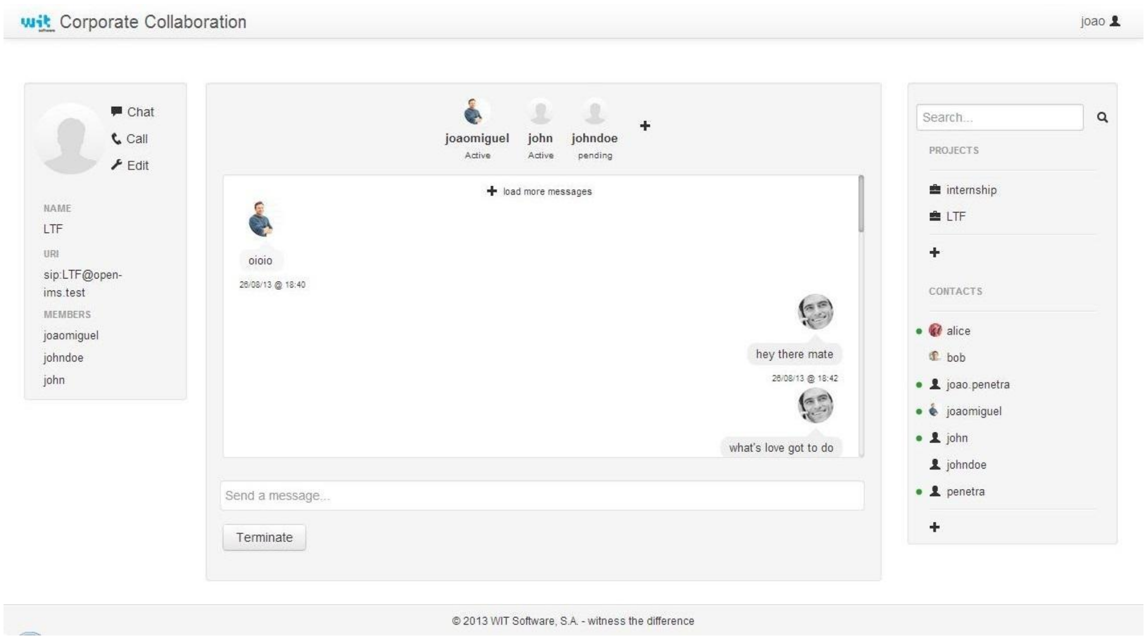


Figure 11 – Collaboration Service – One to One Audio Call

The figure above shows a call that is taking place between two users. It is important to note that the left and right panel are the same but the dashboard represents an audio call that is happening. This flexibility in terms of content organization is provided easily by AngularJS.



oioio
26/08/13 @ 18:40

hey there mate
26/08/13 @ 18:42

what's love got to do

Figure 12 - Collaboration Service - Project Chat

This last figure shows a project chat that is happening between three users and another one is offline. This view differs from the normal chat because it shows who is participating and it is possible to invite more people using the plus sign. There is also a button to fetch previous messages of this conversation.

Section 3 - Android XMPP Client

3.1. State of the Art

The XMPP protocol was first released in 1999 and since then a substantial number of different clients using this protocol have been released [25]. The first idea behind the analysis of the state of the art was finding clients that merged RCS features with the XMPP protocol. However, this search proved to be ineffective because no results were found. This way, it was decided that the scope would be reduced to Android Applications that use XMPP as their main signalling protocol.

3.1.1. Functionalities Comparison

This section creates a comparison of the several applications that were reported as Android Messaging Applications that use XMPP.

Table 18 - Android XMPP Client - Functionalities Comparison

	Agile Messenger	BeeJive IM	BEEM	Gibber	IM+	Octrotalk	Talkonaut	Xabber
Manage Contact list	✓	✓	✓	✓	✓	✓	✓	✓
Instant Messaging	✓	✓	✓	✓	✓	✓	✓	✓
Group Messaging	✓	✓	✓	✓	✓	✓	✓	✓
Message History	✓	✓	✓	✓	✓	✓	✓	✓
Message Composing	✗	✗	✗	✗	✗	✗	✓	✓
Custom Status	✗	✓	✗	✗	✗	✓	✗	✗
Share Files	✓	✓	✗	✗	✓	✓	✓	✗
Presence	✓	✓	✓	✓	✓	✓	✓	✓
SIP compatible (Audio or Video)	✗	✗	✗	✗	✗	✓	✓	✗
Multiple Accounts	✓	✓	✗	✓	✓	✗	✓	✓
Price	Free	Free/$9.99	Free	Free	$4.99	Free	Free	Free
Platforms	Android, BlackBerry, iOS, Nokia, Palm, Sony Ericcson, Windows Mobile	Android BlackBerry iOS	Android	Android	Android BlackBerry iOS J2ME Windows Phone	Android iOS BlackBerry Windows Phone PC OS X	Android iOS J2ME Symbian Windows Phone	Android

The research demonstrated unexpected results. Although it is not fully connected with RCS and the GSMA, there were two clients capable of using SIP to establish audio and video calls: Octrotalk and Talkonaut. Furthermore, the analysis of these applications led to the conclusion that most of them were prototypes with poor design and basic functionalities. In terms of pricing and platform support, the applications offer mostly free Android applications interoperable with other operative systems such as iOS, BlackBerry and WMP.

3.2. Platform Design

This chapter describes the design of the Android XMPP Client. Because it is just a small prototype that is implemented on top of a pre-existing client it was not necessary to design it so thoroughly. Additionally, because the pre-existing client already has its own requirements and definitions force this client to follow those requirements and definitions.

3.2.1. Analysis of Requirements

The meetings that occurred with the stakeholders and the analysis of the pre-existing client originated the following requirements. They are separated in two categories: General Requirements and Functional Requirements.

3.2.1.1. General Requirements

Table 19 - Android XMPP Client - General Requirements

Id	Title	Description	Priority
XAC_GR _01	Extend Pre-existing Client	The Client must be implemented on top the pre-existing Android Client developed by Wit-Software.	Critical
XAC_GR _02	Compatibility of Technologies	The Client must use technologies compatible with the ones being used by the pre-existing Android Client.	Critical
XAC_GR _03	Compatible UI/UX	The implemented User Interface and User Experience of the Client must be compatible with the rest of the pre-existing Client.	High
XAC_GR_04	Follow XMPP Specification	The Client must follow the XMPP protocol specification in order to support any existing server.	Critical
XAC_GR_05	Security	The Client must use SASL or OAUTH2 security mechanisms in order to assure the confidienciality of the users' credentials during the authentication phase.	Critical
XAC_GR_06	Pre-Configuration	The Client must be pre-configurable with well-known XMPP Servers to avoid asking the users for this information.	High

3.2.1.2. Functional Requirements

The functional requirements describe the actions that the user can execute. These requirements are divided in two categories: Account Management and Instant Messaging.

Table 20 - Android XMPP Client - Account Management Requirements

Id	Title	Description	Priority
XAC_AM _01	Add Accounts	Users must be able to add multiple accounts to the client.	Critical
XAC_AM _02	Edit Accounts	Users must be able to edit multiple accounts present in the client.	Critical
XAC_AM _03	Delete Accounts	Users must be able to edit delete accounts present in the client.	Critical
XAC_AM _04	Create Session	Users must be able to create a session by authenticating accounts that were added previously.	Critical
XAC_AM_05	Terminate Session	Users must be able to terminate created sessions.	Critical

XAC_AM_06	Contact List	Users must be able to access the Contact List received by the XMPP Server after the account authentication.	Critical

Table 21 - Android XMPP Client - Instant Messaging Requirements

Id	Title	Description	Priority
XAC_IM _01	One-to-One Chat	Users must be able to create one-to-one chat sessions with users present in their Contact List.	Critical
XAC_IM _02	Typing-notifications	Users must be able to send and receive typing notifications.	Medium
XAC_IM _03	Message History	Users must be able to see a list of previous conversations with their contacts and its respective content.	Critical

3.2.2. Technologies

In order to implement the desired prototype, it was necessary to find the most appropriate XMPP library to communicate with XMPP Servers. Considering that the logic layer of the pre-existing client was programmed in C and C++, it was necessary to find a library developed using one of the two programming languages. It was also necessary that the library was under a free license. This task proved to be a great challenge due to the low number of libraries available and because most of them were not under a royalty-free license. The research conducted is present in the Annex C (Technologies) and it concluded that the best library is named Libjingle. This library in C++ was developed by Google which assured before-hand the compatibility with their XMPP Servers. Although most of the Android Applications are developed using Java as its programming language, the developers of this Client make use of the Java Native Interface (JNI) [29] to be able to run native code developed in C++. Besides boosting the performance of the application because native code runs faster than the code ran in the JVM, it allows sharing code with the iOS application.

What proved to be another great challenge was compiling this library to work with the pre-existing Android Client. The first challenge was cross-compiling the library. Cross-compiling can be defined as compiling a certain piece of code when the host architecture is different from the build architecture. In this case the host architecture was ARM (typical in Android devices) while the host architecture was x86. The other challenge was linking the XMPP library with the pre-existing library because there were duplicate libraries used by both of them. To overcome this difficulty, it was necessary to alter the compilation process of the XMPP library to use the same libraries that were already in the pre-existing library.

3.2.3. Architecture

This section describes the architecture where the Android XMPP Client is inserted. Firstly, a high-level view of the system is shown with the aim of comprehending which components are present. Secondly, the client is separated in several layers to demonstrate how each one works and how they connect with each other.

3.2.3.1. High-Level Architecture

The figure below presents the main components of the system with a clear separation of the components of the client and the server. It is possible to see that the Client side is composed by the Android User Interface, the Client Library and a database and the Server side includes any XMPP Server available.

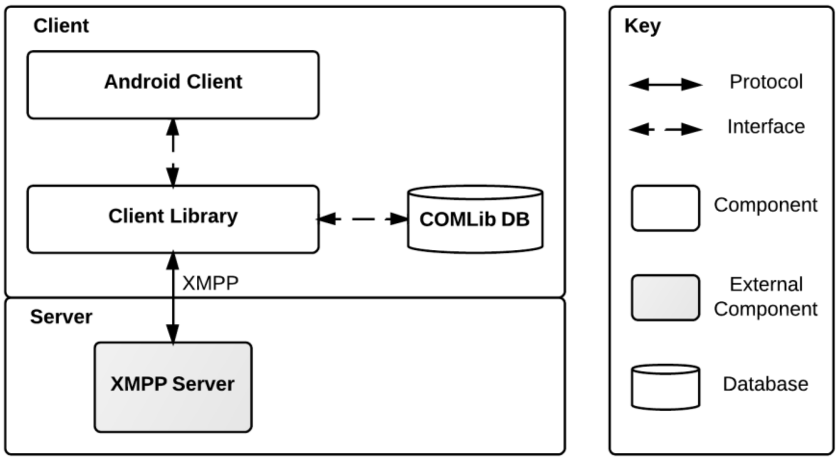


Figure 13 - Android XMPP Client - High-Level Architecture

Given that the main focus of this section will be on the Client side, the next figure provides a separation of the Client side in different layers.

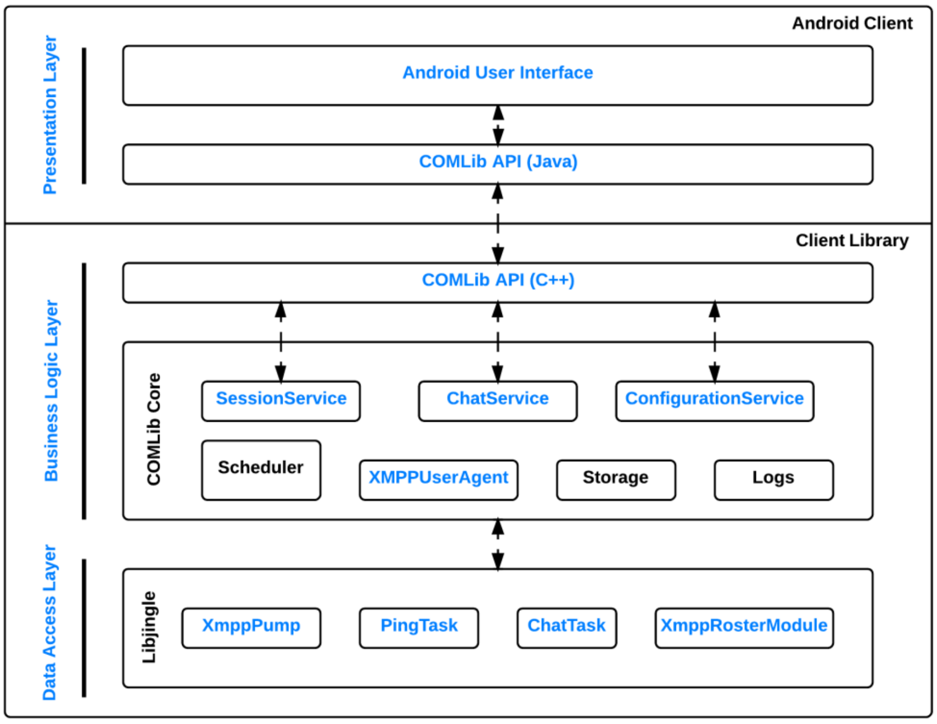


Figure 14 - Android XMPP Client - Client Architecture

The figure demonstrates that the Android Client is composed by the Presentation Layer while the Client Library possesses the Business Logic Layer and the Data Access Layer.

One big advantage of this separation is allowing multiple applications with different Presentation layers using the same Client Library. The reutilization of the Client library can be extremely important to decrease the development cost of new applications and it allows enriching the Client Library when new requirements arise from new clients. It is also important to have a clear separation between the Business Logic and the Data Access Layer in order to avoid being dependent of a single library such as Libjingle.

3.2.3.2. Client Library

The Figure below illustrates the architecture of the Client Library. For the sake of simplicity only the main components that are useful for this client are shown. This decision was made because the pre-existing client is much more complex and possesses a large amount of different components.

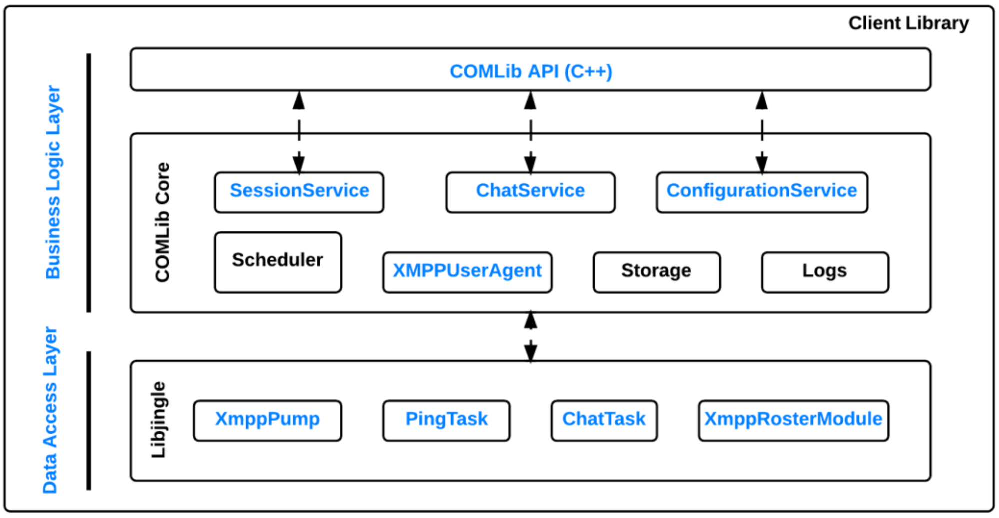


Figure 15 - Android XMPP Client – Client Library Architecture

Before describing the two layers there are some important concepts that, given their extreme importance, should be presented before-hand. These concepts are: Asynchronous calls, Event Subscriptions and Callbacks. The Asynchronous Calls are critical to provide a better User Experience because the User Interface is not blocked while a desired action is being executed. However, the use of Asynchronous Calls requires the existence of an extraordinary method of getting the result for the executed action. The Event Subscriptions and Callbacks are able to solve this problem.

The subscription of events is the best mechanism to guarantee fast and reliable communication between the User Interface and the Client Library. For example, before attempting to create a new session, the application should subscribe an event named RegistrationState. This way, when the user commands the application to register, an asynchronous call is made. Then, after the registration is complete, the application is notified through the change of the RegistrationState. Finally, if the application happens to lose connectivity to the network, the application is immediately notified through this event.

Callbacks are a traditional programming technique where a certain piece of code is passed as an argument to the called method. After that method has finished, the passed piece of code is executed. Contextualizing, if the action that is being executed is not associated with a common task and therefore, not available by subscribing an event, the caller function can pass a piece of code that will be executed when the asynchronous call has finished its course.

Business Logic Layer

The Business Logic Layer plays a central role on the Client Library and is composed by the COMLib API and the COMLib Core. This layer is responsible for receiving all the actions from the user and for alerting the user when new content arrives from the Data Access Layer. Considering this, when the Client Application invokes a method on the COMLib API, the API is in charge of forwarding that request to the respective Service. After this, the Service will schedule an asynchronous job through the Scheduler class to accomplish the task. The advantages of using this Scheduler are: avoiding blocking the application with the usage of asynchronous calls, scheduling Jobs to be ran more than once for periodic tasks and trying to reschedule Jobs when errors occur. The type of the execution of the Job differs from action to action. It might be necessary to launch a Job that registers the user when the application is started, it might be necessary to fetch the contact list from the server or fetch previous message history from the local database. The table below provides more information about this layer.

Table 22 - Android XMPP Client - Business Logic Layer Components

Name	Description
COMLib API – C++	Exposes public interfaces to all the available Services.
SessionService	Exposes internal interfaces that allow Account Management and Session Management and subscription of events.
ChatService	Exposes internal interfaces to manage the exchange of instant messages and subscription of events.
ConfigurationService	Exposes internal interfaces to change the configuration of the application and subscription of events.
Scheduler	Module that allows the creation and scheduling of Jobs.
XMPPUserAgent	UserAgent that exposes internal interfaces to enable the communication with the XMPP Server.
Storage	Module used by Services to persist and fetch content.
Logs	Module used by Services to enable logging management.

Data Access Layer

The Data Access Layer is in charge of connecting the upper layers of the Client Library with the needed third-party libraries. While this prototype requires the usage of Libjingle to communicate with XMPP servers, the pre-existing client uses other libraries to communicate with SIP, MSRP or RTP.

In order to establish the connection with the server, this layer is responsible for creating an abstraction to what lies below but also implement the necessary operations to translate the data that comes from the XMPP server. Once again, the use of asynchronous calls obliges the use of Callbacks to communicate with the upper layers of the application. Therefore, before the start of any interaction with the Server, the XMPPUserAgent registers a set of Callbacks on this layer. After that, when the upper layers request something or new content reaches this layer the callbacks are executed.

Table 23 - Android XMPP Client - Data Access Layer Components

Name	Description
XmppPump	XmppPump handles the connection with the server. It possesses methods to login, logout and send stanzas. Because some of these actions also work asynchronously, the object that creates this XmppPump should implement the interface XmppPumpNotify and pass itself to the constructor of the XmppPump. This way, when the result reaches the XmppPump it knows what needs to be executed afterwards.
XmppTask	Base Class for asynchronous XMPP tasks. The classes that derive from this class are able to act as stanza senders and stanza listeners. They are responsible for receiving and translating the content and alert the registered listeners.
PingTask	PingTask extends XmppTask and runs on a separate thread to perform periodic pings to the server. Alerts the upper layer when it does not receive a any response to configurable number of consecutive pings.
ChatReceiveTask	ChatReceiveTask also extends XmppTask in order to be able to receive chat messages and composing notifications.
XmppRosterModule	This Module is responsible for sending and processing stanzas that contain presence information.

3.2.3.3. Android Client

The Presentation Layer is the only layer of the topmost component that is responsible for displaying to the user the content that comes from the lower layers. This layer is composed by the Android User Interface and the COMLib API in Java. Both components are developed using Java and the COMLib API requires the use of the Java Native Interface to communicate with the public API present in the Client Library. In order to function properly, it is also necessary to instruct the JNI how the parameters passed to the Java API should be translated to the C++ API and vice versa. Although the COMLib API in Java is an adapter of the COMLib API in C++ it should be seen as a component of the Presentation Layer. If for example we want to integrate the Client Library with an iOS application, the Java API will not be used. The next figure describes the packages and their relationships of the Android User Interface in a UML 2.0 Package Diagram.

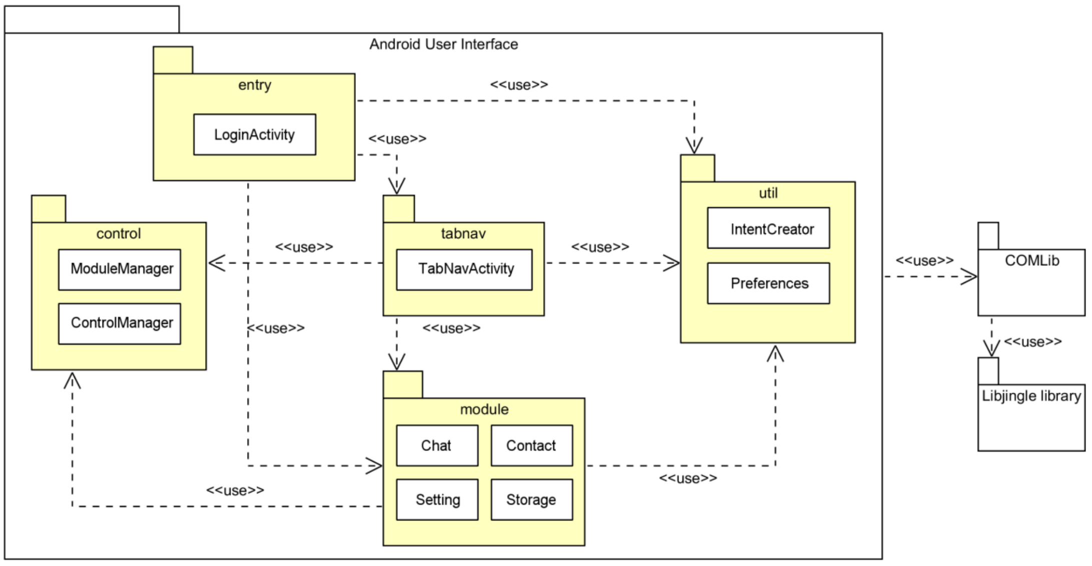


Figure 16 - Android XMPP Client - Package Diagram

The following table provides a description of each package and its elements. Once again, only the most important components for this prototype are shown due to the complexity of the component.

Table 24 - Android XMPP Client – Android User Interface Components

Package Name	Description
entry	The Login Activity is the main entry point when the application is started for the first time. This activity is responsible for checking the actual configuration of the device. If needed be, the activity may prompt the user for credentials before interacting with COMLib to finish the configuration procedure. After this, the LoginActivity launches the TabNavActivity and is destroyed.
tabnav	The TabNavActivity is the only component of this package and it is in charge of loading all the necessary UI modules. The architecture of this Client also favours reusability due to its ability to load UI modules based on a configuration. For example, the normal application is separated in three distinct tabs: Contact List, Chat List and Settings. But if a new client only wants to possess the chat list only, this can be easily done by changing the XML configuration files and recompiling the application.
control	This package contains the ModuleManager and the ControlManager. The ModuleManager is responsible for interacting with the aforementioned XML configuration files so the UI modules are loaded correctly. The ControlManager is in charge of managing the sessions of the application. This is the manager that is responsible for registering the client on XMPP servers with the assistance of the COMLib and Libjingle.
module	This package contains several different types of modules that are connected with the available features. There are modules for the Contacts, Chats, Settings and Storage. Each module is then comprised by different components. There are managers responsible for the interaction with the COMLib, adapters responsible for the conversion of the data sent from the COMLib and User Interface components that receive and display content from and to the user.
util	This package contains classes that are helpful to several components of the system. For example, the IntentCreator class allows the creation of new Activities.
COMLib	Library developed in C/C++ by WIT-Software that is able to use several different types of technologies with the aim of putting people in contact. These technologies include SIP/IMS, SMS and MMS. This prototype adds the XMPP protocol to this list.
Libjingle	XMPP library developed in C++ by Google responsible for the interaction with the XMPP Servers.

For a better understanding of how the architecture of the application works, the next diagram illustrates an example sequence diagram from the Android User Interface to the bottom of the *Data Access Layer*. This example portraits what is necessary to register the user on the XMPP Server.

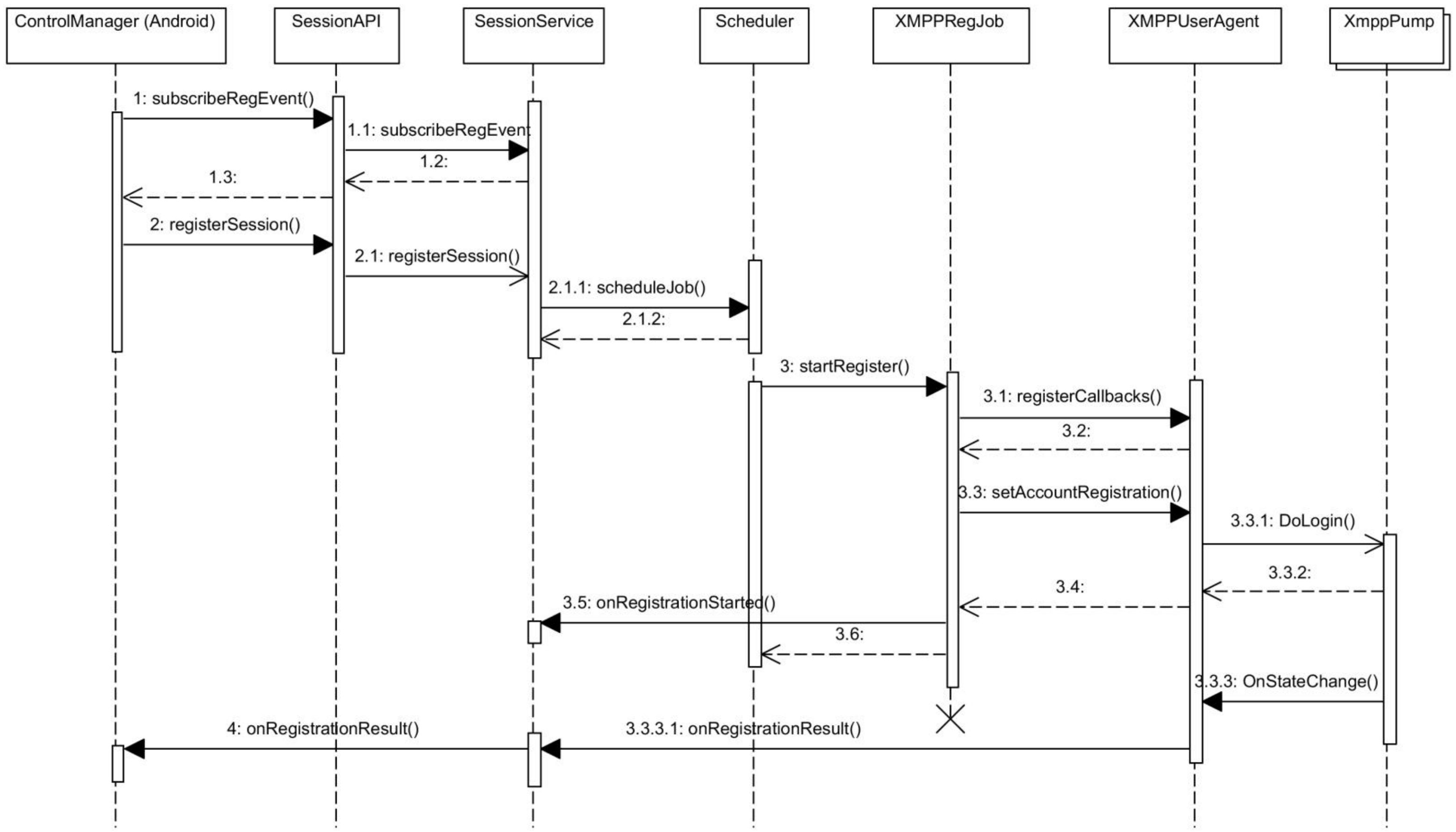


Figure 17 - Android XMPP Client - Registration Sequence Diagram

As it was mentioned, before the registration starts, the ControlManager should subscribe the registration state. After that, it invokes the registerSession method that will schedule a XMPPRegistrationJob with the help of the JobScheduler. This Job is responsible for initializing the XMPPUserAgent with a set of callbacks if necessary and then it should pass the user credentials to the XmppPump which will proceed with the registration. When the server completes or denies the registration, the previously registered callbacks are invoked to alert the User Interface of the result. From the moment that the ControlManager subscribes the registration state, if there is any change, the ControlManager is notified.

3.3. Implementation

The first step of the implementation phase was extending the Client Library to enable the communication with XMPP Servers. Secondly, the modified library was integrated with the Android Client. Finally, the Android Client was extended to use the new functionalities provided by the new Client Library.

3.3.1. Client Library

This required cross compiling the selected XMPP library as it was already mentioned during the platform design phase. This task was extremely enriching because it allowed revisiting concepts that were lectured during the subject of Compilers and learning new concepts such as linking dynamic and shared libraries. This task also required familiarization with a tool named Generate Your Projects (GYP) which facilitated the creation of the static library that would be appended to the Client Library. This tool provides an easy way to specify the source code, the dependencies and the flags that are necessary to process, compile and link the desired code to a target operating system. Besides creating the library that was used on Android, this tool was capable of generating the library that was used to build a simple command line application capable of invoking the public API exposed by the Business Logic Layer.

3.3.1.1. SessionService

The SessionService is a crucial component that is responsible for managing the registration in XMPP servers. Moreover, since this component also holds the registration states of the users, it is accessed by several other components that require this information before proceeding with their tasks. This fact obliges a clear notion of which states exist and which transitions are possible from state to state. The state diagram below illustrates the several states and transitions possible.

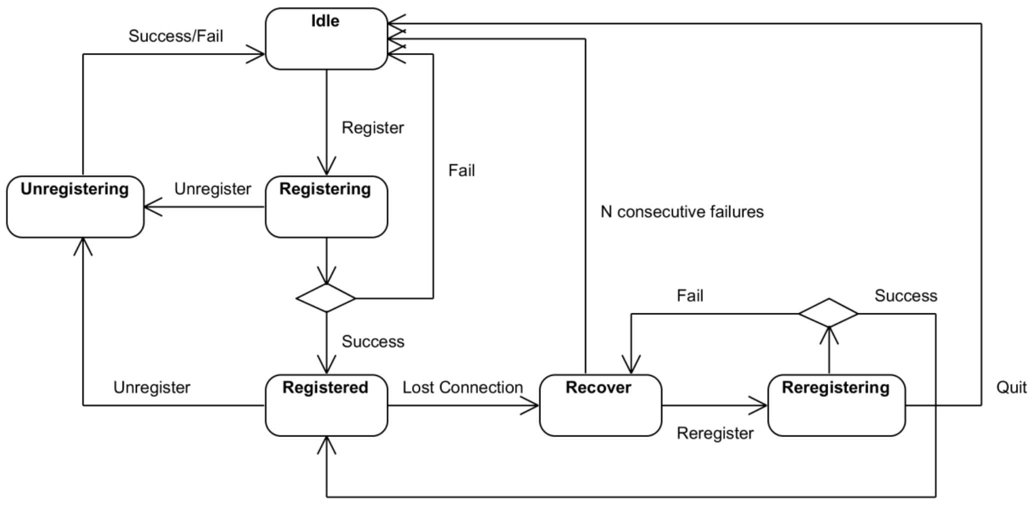


Figure 18 - Android XMPP Library - Registration State Machine Diagram

To accomplish registration the SessionService is responsible for creating and scheduling an asynchronous RegistrationJob. The following class diagram describes the classes that were developed in order to enable the start of the registration and the cascade of callbacks that allow the return of the result. This class diagram completes the sequence diagram that was presented on the description of the architecture.

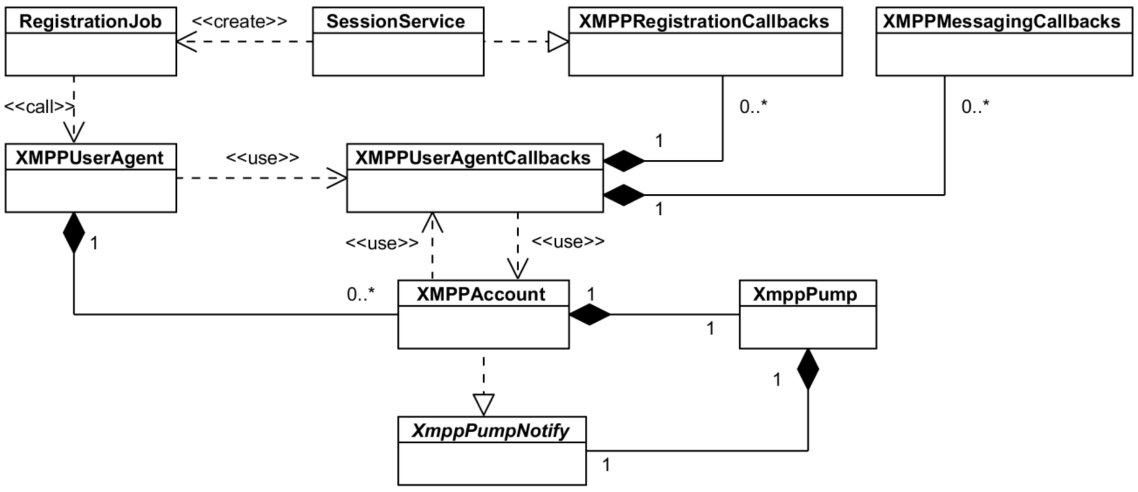


Figure 19 - Android XMPP Client - Client Library Class Diagram

Besides the creation of Registration jobs, the Session Service implements the XMPPRegistrationCallbacks which binds itself on the XMPPUserAgentCallbacks when instantiated. Similarly, the XMPPMessagingCallbacks will be implemented by the Chat Service and the XMPPContactCallbacks by the Contact Service. This way, the XMPPUserAgentCallbacks is seen as the class that knows which Services should be alerted when certain events occur. But this class also needs to be informed by the lower components. That said, before the UserAgent starts the registration, it is necessary that the XMPPUserAgentCallbacks passess a set of Callbacks to the XMPPAccount that is being registered. These Callback functions iterate through the bound XMPPRegistrationCallbacks and alert the Services. Finally, because the XMPP library also works asynchronously, the XMPPAccount implementes the XmppPumpNotify and passes itself to the constructor of the XmppPump in order to be notified when the registration is done.

3.3.2. Integration between Android Client and Client Library

The integration between the Android Client and the Client Library is done through the use of the Java Native Interface (JNI). This integration requires the conversion of the C++ API to the Java API and the translation of the C++ data structures to Java data structures and vice versa. The conversion was done using a python script developed by the team who built the Client Library. What this script does is parsing the C++ code and producing the equivalent Java code that is included on the Android Client. For example, a function that returns a std::vector in the C++ API will be converted to a Java.List. The translation of the objects, however, is not done automatically. Considering the same example where it is necessary to return a vector with the Contact List, it was necessary to create two different translators. The first one receives a JNIEnv object which allows the interaction with the Java environment and a std::vector that contains the contacts. What this translator does is creating a new Java.ArrayList and then for each member of the std::vector it calls the second translator that will translate a single Contact. This second translator also receives the JNIEnv object and it will create a Java Contact based on the content of the C++ Contact.

3.3.3. Android Client

The Android Client represents the last step necessary to accomplish the final objective of the prototype. This step represented another challenge since it was the first time that the author was developing for the Android Operative System. The majority of the work involved integrating the existing application with the new Services that were available through the new Public API. The feature that required more effort to integrate was the contact list.

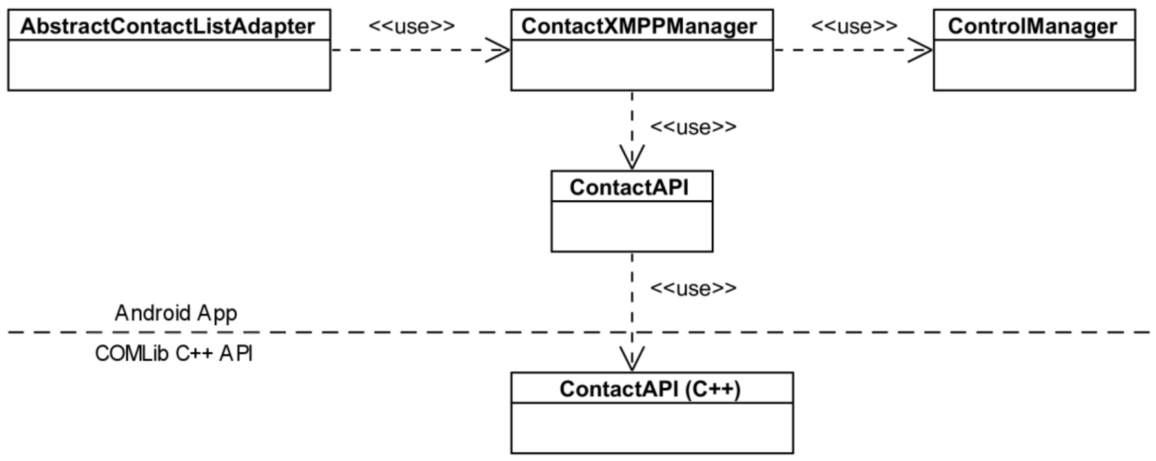


Figure 20 - Android XMPP Client - Contact List Class Diagram

The figure above illustrates what is necessary to present the contact list to the user. Once again, this example makes use of several event subscriptions and callbacks in order to function asynchronously. Firstly, the ContactXMPPManager uses the ControlManager to subscribe the RegistrationState of the application. Once the registration has been completed, the ContactXMPPManager is notified and starts the process of fetching the contact list by invoking the ContactAPI. When the contact list has been fetched, the COMLib will notify the ContanctXMPPManager which will notify the AbstractContactListAdapter. This adapter is responsible for making the bridge between the model objects and the presentation view.

3.4. Quality Assurance

The Android prototype was also object of quality assurance but comprehensively not as much as the Collaboration Service. During the development of the Data Access Layer and the Business Logic Layer, it was necessary to develop a command line application capable of interacting with the XMPP Server. This command line application isolated the Client Library from the Android Application which detecting the source of the encountered issues.

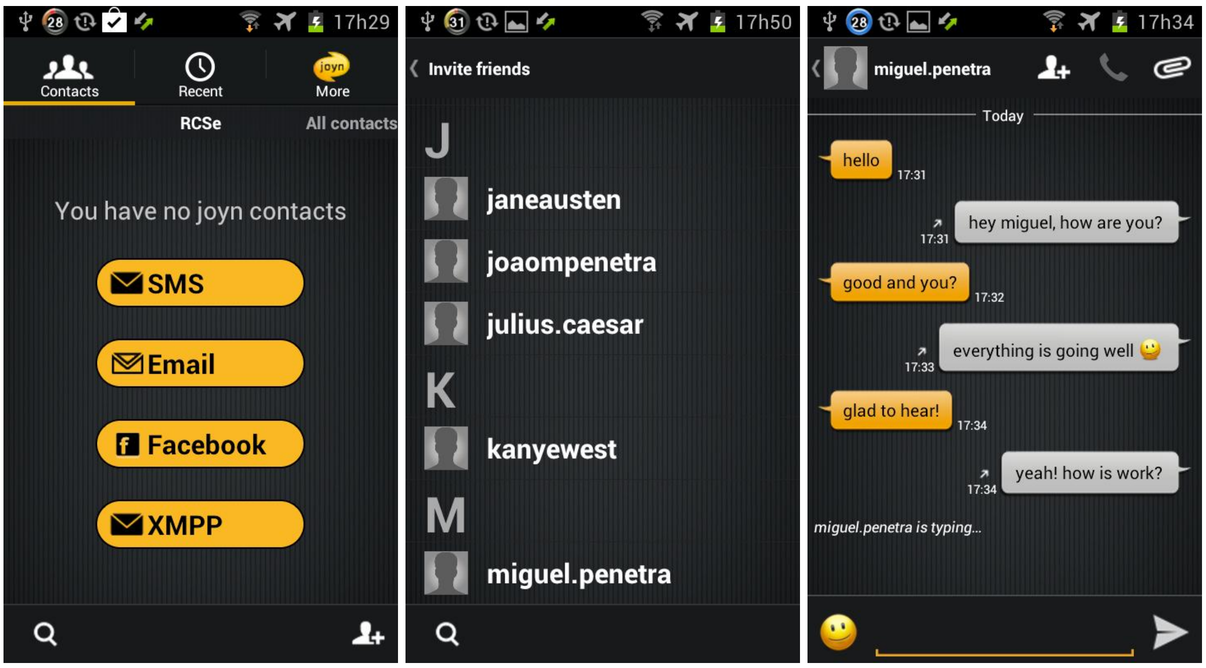
3.5. Work Accomplished

The following table describes what was accomplished on the Android Client.

Table 25 - Android XMPP Client - Work Accomplished

Features	Account Management	Contact List	1-to-1 Chat	Chat History	Composing Notifications
Result	✓	✓	✓	✓	✓

The following figures demonstrate the screens of the application that the author worked on.



hello 17:31

hey miguel, how are you? 17:31

good and you? 17:32

everything is going well 17:33

glad to hear! 17:34

yeah! how is work? 17:34

miguel.penetra is typing...

Figure 21 - Android XMPP Client - Application outcome

The first screen is the splash screen of the application and it shows the several ways that the user can use to communicate with other people. On this screen it was necessary to add the XMPP button and the logic necessary to navigate to the next screen. The second screen contains the several contacts that the user possesses. To provide a better User Experience, the contacts are ordered and separated by their first letter. Finally, the third screenshot displays the chat view with a certain user. This third screen also shows the composing notification that is received when the other user is composing a message.

Section 4 - Methodology

This chapter describes the methodology that was followed during this project. It is extremely important to understand how the chosen methodology works and explain why it was chosen.

4.1. Scrum

The chosen methodology is based on Scrum [30]. Scrum, which aims to improve the profession of software development, is a development framework intended to manage software projects. This framework is based on an iterative and incremental process that defines that there should be repeated cycles in which the development is focused on different small parts of the final product. Each one of these iterations, which are called Sprints in Scrum, can be divided in four major tasks: requirements analysis, design, implementation and testing. As soon as each sprint is finished, the product is shown to the client to minimize the risk of having to redevelop something very complex. This way, besides having frequent demonstrations that enable quick changes without losing a large amount of time and resources, the several iterations allow teams to take advantage of what was learned previously and act accordingly. Scrum consists of artifacts, teams and events.

4.2. Artifacts

The main purpose of the artifacts is representing the work related to the project. The most important artifacts are the product backlog and the sprint backlog. The Product Backlog is an ordered list of all the tasks related to the project. With the evolution of the project, the product backlog is constantly updated with new items or with item reorganization. The Sprint Backlog contains the list of the requirements that are planned to be implemented during a sprint. This type of backlog is also very dynamic because the development team may learn that a new feature is needed or that their initial predictions were incorrect and the backlog needs to be adjusted.

4.3. Team

Scrum teams have three different types of members: Product Owner, Development Team and Scrum Master. The Product Owner is the person responsible for the product and for the content that is present on the Product Backlog. During the several sprints, the Product Owner is also responsible for assuring the quality of the work that is being developed. The Development Team is constituted by a varying number of members and it is suggested that this number should be between three and nine. This team is responsible for forecasting what is going to be accomplished during the next Sprint. The Scrum Master is the person responsible for ensuring that all the theories, practices and rules are followed in order to maximize the value created by the team. This person might be required to work with the Product Owner in order to find effective techniques to manage the Product Backlog or to develop a long-term plan for the product. Finally, this person is also responsible for coaching and leading the Development Team to create high-value products.

4.4. Events

There are several types of events that allow the development to be as inspective and as transparent as possible. The heart event of Scrum is the Sprint. A Sprint represents one of the iterations and it should not exceed the period of one month. During each Sprint there are other events such as the Sprint Planning, Daily Scrum, Sprint Review and Sprint Retrospective. The Sprint Planning allows the team to forecast what is going to be done and who will be responsible for achieving it. The Daily Scrum is a 15-minute daily event that allows the Development Team to synchronize the work that has been accomplished since the last meeting and plan what is going to be done in the next 24 hours. The Sprint Review occurs at the end of each Sprint and its purpose is to determine what was done and what was not, discuss what went well and what went wrong during and how were the problems solved. The conclusions of these review meetings also bring valuable inputs to the next Sprint Planning. Finally, the Sprint Retrospective is an event that has the purpose to complement the conclusions that may be taken for the previous sprint. While the Sprint Planning attempts to plan what is going to be accomplished, the Sprint Retrospective inspects the relationships between members, the processes and tools used in order to implement team improvements on the next Sprint.

4.5. Implemented Methodology

Although there are some differences, the methodology used is highly based on Scrum. The members of the team are: Paulo Sousa, the Product Owner, João Costa, the Scrum Master and the author of this report is the only member of the Development Team. The fact that there is only one member on the Development Team obliges that there is more interaction with the other members in order to increase productivity and constructiveness. This way, it was decided that each Sprint would take two weeks. The rest of the events and processes will occur as suggested.

4.6. First Semester Work

It is important to mention that the internship plan presented was substantially altered. During the first semester, work consisted in researching and planning everything while the second semester comprised the implementation.

Table 26 - Work developed on 1st Semester

Sprint #	From	To	Tasks
1	03/09/2012	17/09/2012	• Planning Internship meetings • Scrum Session • State of the Art
2	17/09/2012	01/10/2012	• State of the Art • Requirements elicitation • Research IMS and implementations • Installing and configuring OpenIMS
3	01/10/2012	15/10/2012	• First Meeting DEI Supervisor • Research HSS Provisioning • Prototype HSS Provisioning
4	15/10/2012	29/10/2012	• Architecture Draft Research • Research SIP Servlets vs JAIN SLEE • Prototype SIP Servlets and JAIN SLEE
5	29/10/2012	12/11/2012	• Research Application Servers • Second Meeting DEI Supervisor • Internship Report
6	12/11/2012	26/11/2012	• Research WebSockets • Research Interaction between Browser and Server
7	26/11/2012	10/12/2012	• Resarch JavaScript SIP and MSRP Stacks • Prototype SIP Registration in from JAIN SIP JS
8	10/12/2012	24/12/2012	• Research RIA • Research Media Server • Third Meeting DEI Supervisor
9	24/12/2012	07/01/2013	• Architecture Specification • Review State of the Art
10	07/01/2013	21/01/2013	• Internship Report
11	21/01/2013	04/02/2013	• Internship Report • Presentation

4.7. Second Semester Initial Plan

Sprint #	From	To	Tasks
12	04/02/2013	18/02/2013	• Intermediate Presentation – 16 hours • Application Mockups - 24 hours • HSS Provisioning and WS Registration – 32 hours • Research/Configure WCAS – 16 hours
13	18/02/2013	04/03/2013	• Research XMPP – 24 hours • Build COMLib and Libjingle for linux – 16 hours • Build COMLib and Libjingle for Android – 16 hours • XMPPUserAgent – 24 hours
14	04/03/2013	18/03/2013	• XMPPUAAccount – 16 hours • Ping Task – 12 hours • Chat Receive Task – 12 hours • Roster and Presence – 16 hours • SessionService – 24 hours
15	18/03/2013	01/04/2013	• SessionService – 16 hours • Contact List in COMLib – 24 hours • ContactList API - 8 hours • Chat 1-1 in Android – 24 hours • Research/Configure WCAS – 8 hours
16	01/10/2013	15/04/2013	• Research AngularJS – 8 hours • Edit Account details – 16 hours • IMS Registration through WCAS – 16 hours • WCAS as an Application Server – 40 hours
17	15/04/2013	29/04/2013	• WCAS as an Application Server – 24 hours • Research Message Store – 8 hours • WCAS AddressBook Translators – 12 hours • WCAS AddressBook Web Client – 12 hours • Project Support in JavaScript Client – 16 hours
18	29/04/2013	13/05/2013	• Chat one-to-one – 80 hours
19	13/05/2013	27/05/2013	• Call one-to-one – 56 hours • Project Call – 24 hours
20	27/05/2013	10/06/2013	• Project Call – 32 hours • Project Chat – 48 hours
21	10/06/2013	24/06/2013	• Project Chat – 16 hours • File Transfer one-to-one – 24 hours • Composing Notifications – 16 hours • MessageStore on Collaboration AS – 24 hours
22	24/06/2013	08/07/2013	• MessageStore on Collaboration AS – 80 hours
23	08/07/2013	22/07/2013	• MessagesStore on Gateway Server – 32 hours • Fetch Recent History in JavaScript– 24 hours • Fetch Older History in JavaScript – 24 hours
24	22/07/2013	05/08/2013	• Bug fixing – 40 hours • Acceptance Tests – 40 hours
25	05/08/2013	19/08/2013	• Performance Tests – 80 hours
26	19/08/2013	02/09/203	• Internship Report – 80 hours

4.8. Second Semester Final Plan

Sprint #	From	To	Tasks
12	04/02/2013	18/02/2013	• Intermediate Presentation – 18 hours • Application Mockups - 15 hours • HSS Provisioning and WS Registration – 28 hours • Research/Configure WCAS – 18 hours
13	18/02/2013	04/03/2013	• Research XMPP – 8 hours • Research XMPP C++ Library – 16 hours • Build COMLib with Libjingle – 32 hours • Compile Libjingle for linux – 10 hours • Build libjingle for android – 16 hours
14	04/03/2013	18/03/2013	• XMPPUserAgent – 20 hours • XMPPUAAccount – 20 hours • Ping Task – 12 hours • Chat Receive Task – 12 hours • Roster and Presence – 16 hours
15	18/03/2013	01/04/2013	• SessionService – 40 hours • Contact List in COMLib – 15 hours • ContactList API - 5 hours • Chat 1-1 in Android – 20 hours
16	01/10/2013	15/04/2013	• Research/Configure WCAS – 12 hours • Research AngularJS – 8 hours • Edit Account details – 16 hours • IMS Registration through WCAS – 15 hours
17	15/04/2013	29/04/2013	• WCAS as an Application Server – 54 hours • Research Message Store – 8 hours • WCAS AddressBook Translators – 12 hours • WCAS AddressBook Web Client – 12 hours
18	29/04/2013	13/05/2013	• Project Support in JavaScript Client – 20 hours • Chat one-to-one – 60 hours
19	13/05/2013	27/05/2013	• Call one-to-one – 48 hours • Project Call – 32 hours
20	27/05/2013	10/06/2013	• Project Call – 32 hours • Project Chat – 48 hours
21	10/06/2013	24/06/2013	• Project Chat – 16 hours • File Transfer one-to-one – 24 hours • Composing Notifications – 16 hours • MessageStore on Collaboration AS – 24 hours
22	24/06/2013	08/07/2013	• MessageStore on Collaboration AS – 80 hours
23	08/07/2013	22/07/2013	• MessagesStore on Gateway Server – 32 hours • Fetch Recent History in JavaScript– 24 hours • Fetch Older History in JavaScript – 24 hours
24	22/07/2013	05/08/2013	• Bug fixing – 40 hours • Acceptance Tests – 40 hours
25	05/08/2013	19/08/2013	• Performance Tests – 80 hours
26	19/08/2013	02/09/203	• Internship Report – 80 hours

Section 5 - Conclusions

The work developed during this internship was the biggest challenge that the author has ever faced throughout his academic life. On a personal level, it contributed greatly to the growth as a person and as a software developer. On the view of the company, it resulted in a product that extends what already exists on the company with new functionalities and new technologies.

One of the biggest advantages of this internship was the possibility to work with the several products of the company and with its developers. These products are immensely different from each other which was great in terms of getting in contact with new technologies, different programming languages as well as new software paradigms. In order to work with products such as WCAS, the JavaScript Client Library, COMLib and the Android Client, it was necessary to study their infrastructure, interpret decisions and understand how they could be extended. This extensibility also required a high interaction with other developers which was incredibly invaluable.

Considering the work developed on the Collaboration Service, the opportunity to work in an SIP environment represented a unique professional opportunity and allowed reaching a great specialization in this area. Furthermore, the contact with technologies such as Web Sockets and WebRTC was extremely valuable to the author.

The development of the Android Client supported the need to learn the basics of developing to this operative system and it allowed learning how to work with the Java Native Interface.

Finally, the technical knowledge and the experience that the author obtained during this internship along with the several lessons learned will certainly help him throughout his future as a Software Engineer.

5.1. Future Work

The work developed during this internship enabled the creation of two products capable of shortening the distance between people that have the necessity or wish to communicate with others. These products were fully functional but the opportunity to extend them will enrich the final result even more.

Section 6 - References

[1] Wit-Software, "Wit-Software," [Online]. Available: http://wit-software.pt/.

[2] Business Travel News, "Small and Medium Enterprise Report," [Online]. Available: http://businesstravelnews.texterity.com/businesstravelnews/20120416#pg9.

[3] XMPP Standards Foundation, "The XMPP Standards Foundation," [Online]. Available: http://xmpp.org/.

[4] GSMA, "Rich Communications," [Online]. Available: http://www.gsma.com/rcs/.

[5] Google, "Google Developers," [Online]. Available: https://developers.google.com/talk/talk_developers_home.

[6] Facebook, "Facebook Developers," [Online]. Available: https://developers.facebook.com/docs/chat/.

[7] Adobe, "Web conferencing software - Conference services | Adobe Connect 9," [Online]. Available: http://www.adobe.com/products/adobeconnect.html.

[8] BlueJeans, "Video Conferencing Service - Interoperable, Cloud-based, Affordable | Blue Jeans," [Online]. Available: http://bluejeans.com/.

[9] 37signals, "Business group chat, file sharing, group decision making: Campfire," [Online]. Available: http://campfirenow.com/.

[10] HipChat, "Private group chat and IM, business and team collaboration - HipChat," [Online]. Available: https://www.hipchat.com/.

[11] Flowdock, "Flowdock," [Online]. Available: https://www.flowdock.com/.

[12] FuzeBox, "FuzeBox," [Online]. Available: https://www.fuzebox.com/.

[13] Citrix, "Citrix GoToMeeting," [Online]. Available: http://www.citrix.com/products/gotomeeting/overview.html.

[14] Huddle, "The Enterprise Content Collaboration Platform," [Online]. Available: http://www.huddle.com/.

[15] IBM, "IBM Sametime," [Online]. Available: http://www-01.ibm.com/software/lotus/sametime/.

[16] PGi, "iMeet," [Online]. Available: http://www.pgi.com/us/en/conferencing/iMeet/.

[17] Microsoft, "Unified Communications - Lync 2013," [Online]. Available: http://lync.microsoft.com/en-us/Pages/unified-communications.aspx.

[18] Safa Software, "Saba Meeting," [Online]. Available: http://www.sabameeting.com/.

[19] Dialcom, "Spontania," [Online]. Available: http://www.dialcom.com/en/.

[20] Teambox, "Collaboration Software - Online project management tool for teams," [Online]. Available: http://teambox.com/.

[21] Voxeet, "Voxeet," [Online]. Available: http://www.voxeet.com/index.html.

[22] Cisco, "Cisco WebEx Web Conferencing, Online Meetings, Desktop Sharing, Video Conferencing," [Online]. Available: http://www.webex.com/.

[23] Alexa, "Alexa - The Web Information Company," [Online]. Available: http://www.alexa.com/.

[24] Gartner, "Magic Quadrant for Web Conferencing," [Online]. Available: http://www.gartner.com/technology/reprints.do?id=1-1D78VS4&ct=121212&st=sb#.

[25] Slashdot, "Open Real time Messaging System," [Online]. Available: http://slashdot.org/story/99/01/04/1621211/open-real-time-messaging-system.

[26] GSMA, "RCS," [Online]. Available: http://www.gsma.com/rcs/.

[27] Open Mobile Alliance, "CPMMSGSTORE," [Online]. Available: http://member.openmobilealliance.org/ftp/public_documents/COM/COMCPM/Permanent_documents/OMA-TS-CPM_MessageStorage-V1_0-.

[28] IETF, "RFC3501 - IMAP," [Online]. Available: http://tools.ietf.org/html/rfc3501.

[29] Google Android, "JNI - Android Developers," [Online]. Available: http://developer.android.com/training/articles/perf-jni.html.

[30] K. Schwaber and J. Sutherland, "Scrum Guide," [Online]. Available: http://www.scrum.org/ScrumGuide.aspx.

[31] J. Sutherland, "Ban Gantt charts," [Online]. Available: http://scrum.jeffsutherland.com/2006/02/why-gantt-charts-were-banned-in-first.html.

[32] Kent Beck, "JUnit Web Page," [Online]. Available: https://github.com/kentbeck/junit/wiki.

[33] Cisco. [Online]. Available: http://www.cisco.com/en/US/prod/collateral/voicesw/ps6788/vcallcon/ps6562/product_data_sheet0900aecd80396990.html.

[34] Alcatel-Lucent. [Online]. Available: http://www.alcatel-lucent.com/wps/portal/!ut/p/kcxml/04_Sj9SPykssy0xPLMnMz0vM0Y_QjzKLd4w3MfQFSYGYRq6m-pEoYgbxjgiRIH1vfV-P_NxU_QD9gtzQiHJHR0UAAD_zXg!!/delta/base64xml/L0lJayEvUUd3QndJQSEvNElVRkNBISEvNl9BXzNBRC9lbl93dw!!?LMSG_CABINET=Solution_Product_Catalo.

[35] F. C. Torralba, T. Johansson, H. Österlund and . M. E. T. Díaz-Chirón. [Online]. Available: http://www.faqs.org/patents/app/20120096162#b.

[36] O. I. Core. [Online]. Available: http://www.openimscore.org/.

[37] Orange, [Online]. Available: http://www.orange.com/en/home.

[38] PT Inovação, [Online]. Available: http://www.ptinovacao.pt/.

[39] Open IMS Core, [Online]. Available: http://www.openimscore.org/quotes.

[40] "OpenHSS," [Online]. Available: http://sourceforge.net/projects/openhss/.

[41] G. University of Patras, "OSIMS," [Online]. Available: http://nam.ece.upatras.gr/ppe/?q=node/2.

[42] H. L. Raether and M. J. Rudolph. [Online]. Available: http://www.google.com/patents?id=CPuVAAAAEBAJ&printsec=abstract&zoom=4#v=onepage&q&f=false.

[43] Red Hat, [Online]. Available: http://www.redhat.com/.

[44] Java Community Process, "JSR 311," [Online]. Available: http://jcp.org/en/jsr/detail?id=311.

[45] Java.net, "Jersey," [Online]. Available: http://jersey.java.net/.

[46] Red Hat, "RESTEasy," [Online]. Available: http://www.jboss.org/resteasy.

[47] Apache, "Apache CXF," [Online]. Available: http://cxf.apache.org/docs/jax-rs.html.

[48] "Celtix and XFire merge," [Online]. Available: http://xfire.codehaus.org/XFire+and+Celtix+Merge.

[49] "Atmosphere," [Online]. Available: https://github.com/Atmosphere/atmosphere.

[50] IETF, "WebSocket Protocol," December 2011. [Online]. Available: http://tools.ietf.org/html/rfc6455.

[51] Opera, "Opera Mini," [Online]. Available: http://www.opera.com/mobile/.

[52] Google, "Android Browser," [Online]. Available: http://www.android.com/.

[53] Microsoft, "Internet Explorer," [Online]. Available: http://windows.microsoft.com/en-US/windows-8/internet-explorer.

[54] Springsource, "Groovy Main Page," [Online]. Available: http://groovy.codehaus.org/.

[55] Oracle, "Java Web Page," [Online]. Available: http://www.java.com/en/.

[56] C. Nutter, T. Enebo, O. Bini and N. Sieger, "JRuby Web Page," [Online]. Available: http://jruby.org/.

[57] M. Odersky, "Scala Web Page," [Online]. Available: http://www.scala-lang.org/.

[58] W3C, "HTML5 Server Sent Events Page," [Online]. Available: http://www.w3schools.com/html/html5_serversentevents.asp.

[59] "FlashBridge Web Page," [Online]. Available: http://sourceforge.net/projects/javaflashbridge/.

[60] "Jetty Web Page," [Online]. Available: http://jetty.codehaus.org/jetty/.

[61] Netty, "Netty Web Page," [Online]. Available: https://netty.io/.

[62] Apache Software Foundation, "Tomcat Web Page," [Online]. Available: http://tomcat.apache.org/.

[63] Red Hat, "JBoss Web Page," [Online]. Available: http://www.jboss.org/.

[64] Alcatel-Lucent, "Alcatel-Lucent Web Page," [Online]. Available: http://www.alcatel-lucent.com/wps/portal.

[65] IBM, "IBM Web Page," [Online]. Available: http://www.ibm.com/us/en/.

[66] Sun Microsystems, "Sun Web Page," [Online]. Available: http://www.oracle.com/us/sun/index.htm.

[67] BEA Systems, "BEA Systems Web Page," [Online]. Available: http://www.oracle.com/us/corporate/acquisitions/bea/index.html.

[68] Java Community Process, "JCP Web Page," [Online]. Available: http://jcp.org/en/jsr/detail?id=339.

[69] W3C, "W3C Web Page," [Online]. Available: http://www.w3.org/Submission/wadl/.

[70] Apache CFX, "Apache CFX Testing Web Page," [Online]. Available: https://cwiki.apache.org/confluence/display/CXF20DOC/JAXRS+Testing.

[71] "3Gdb Web Page," [Online]. Available: http://code.google.com/p/hss/.

[72] Apache CFX, "Apache CFX JAX-RS Web Page," [Online]. Available: http://cxf.apache.org/docs/jaxrs-services-description.html.

[73] java.net, "Grizzly Web Page," [Online]. Available: http://grizzly.java.net/.

[74] Java.net, "Jersey Web Page," [Online]. Available: http://jersey.java.net/nonav/documentation/latest/test-framework.html.

[75] JBoss Community, "RESTEasy Web Page," [Online]. Available: http://www.jboss.org/resteasy.

[76] JBoss Community, "RESTEasy Asynchronous details," [Online]. Available: http://docs.jboss.org/resteasy/docs/1.0.0.GA/userguide/html/Asynchronous_HTTP_Request_Processing.html.

[77] Java.net, "GlasshFish Web Page," [Online]. Available: http://glassfish.java.net/.

[78] Apache Software Foundation, "Struts Web Page," [Online]. Available: http://struts.apache.org/.

[79] Google, "Google Web Toolkit Web Page," [Online]. Available: https://developers.google.com/web-toolkit/.

[80] Oracle, "JavaServer Pages Web Page," [Online]. Available: http://www.oracle.com/technetwork/java/javaee/jsp/index.html.

[81] Oracle, "Facelets Web Page," [Online]. Available: http://docs.oracle.com/javaee/6/tutorial/doc/giepx.html.

[82] Java.net, "Facelets vs JSP," [Online]. Available: http://today.java.net/pub/a/today/2006/08/29/developing-with-facelets-jsf-jsp.html.

[83] Zenexity, "Zenexity Web Page," [Online]. Available: http://www.zenexity.com/.

[84] SpringSource, "SpringSource Web Page," [Online]. Available: http://www.springsource.org.

[85] Mobicents, "Mobicents SIP Presence Service," [Online]. Available: http://www.mobicents.org/sip-presence/intro.html.

[86] Kamailio, "Kamailio SIP Server," [Online]. Available: http://www.kamailio.org/w/.

[87] Kamailio, "Kamailio SIP Server," [Online]. Available: http://www.kamailio.org/w/.

[88] OpenSIPS, "openSIPS," [Online]. Available: http://www.opensips.org/.

[89] IETF, "XML Configuration Access Protocol (XCAP)," [Online]. Available: http://tools.ietf.org/html/rfc4825.

[90] OpenXCAP, "OpenXCAP," [Online]. Available: http://openxcap.org/.

[91] SylkServer, "SylkServer, A state of the art, extensible SIP Application Server," [Online]. Available: http://sylkserver.com/.

[92] AG Projects, "SIP Thor," [Online]. Available: http://www.ag-projects.com/sip-thor-products-291.

[93] AG Projects, "SylkServer, A state of the art, extensible SIP Application Server," [Online]. Available: http://sylkserver.com/.

[94] AG Projects, "MSRP Relay," [Online]. Available: http://www.msrprelay.org/.

[95] NEXCOM, "littleIMS," [Online]. Available: http://confluence.cipango.org/display/LITTLEIMS/Home.

[96] NEXCOM, "Cipango," [Online]. Available: http://www.cipango.org/.

[97] Mobicents, "SIP Servlets," [Online]. Available: http://www.mobicents.org/products_sip_servlets.html.

[98] java.net, “GlassFish - Open Source Application Server,” [Online]. Available: http://glassfish.java.net/.

[99] Ars Technica, “Web Browsers market share November 2012,” [Online]. Available: http://arstechnica.com/information-technology/2013/01/internet-explorer-ends-the-year-on-a-high-windows-8-slow-to-get-noticed/.

[100] IETF, “The WebSocket Protocol as a Transport for the Session Initiation Protocol (SIP),” [Online]. Available: http://tools.ietf.org/html/draft-ietf-sipcore-sip-websocket-06.

[101] OverSIP, “OverSIP - the SIP framework you dreamed about,” [Online]. Available: http://www.oversip.net/.

[102] IETF, “ Managing Client-Initiated Connections,” [Online]. Available: http://tools.ietf.org/html/rfc5626#section-3.4.

[103] JsSIP, “JsSIP the Javascript SIP library,” [Online]. Available: http://www.jssip.net/.

[104] WebRTC, “WebRTC,” [Online]. Available: http://www.webrtc.org/home.

[105] JAIN SIP, “JAIN SIP JS,” [Online]. Available: http://code.google.com/p/jain-sip/.

[106] SIP-JS, “SIP-JS,” [Online]. Available: http://code.google.com/p/sip-js/.

[107] IETF, “The WebSocket Protocol as a Transport for the Message Session Relay,” [Online]. Available: http://tools.ietf.org/html/draft-pd-msrp-websocket-02.

[108] Crocodile-RCS, “Crocodile RCS,” [Online]. Available: http://www.crocodile-rcs.com/.

[109] Doubango Telecom, “sipML5,” [Online]. Available: http://sipml5.org/.

[110] IETF, “SIP RFC 3261,” [Online]. Available: http://tools.ietf.org/html/rfc3261.

[111] Adobe, “Adobe,” [Online]. Available: http://www.adobe.com/.

[112] JavaFx, “JavaFx,” [Online]. Available: http://www.oracle.com/technetwork/java/javafx/overview/index.html.

[113] IETF, “SDP: Session Description Protocol,” [Online]. Available: http://tools.ietf.org/html/rfc4566.

[114] Statowl, “Rich Internet Application Market Share,” [Online]. Available: http://www.statowl.com/custom_ria_market_penetration.php?1=1&timeframe=last_3&interval=month&chart_id=13&fltr_br=&fltr_os=&fltr_se=&fltr_cn=&timeframe=last_12.

[115] Microsoft, “CU-RTC-WEB,” [Online]. Available: http://html5labs.interoperabilitybridges.com/cu-rtc-web/cu-rtc-web.htm.

[116] Medooze, “Medooze,” [Online]. Available: http://www.medooze.com/products/mcu.aspx.

[117] IETF, "RFC 4975," [Online]. Available: http://tools.ietf.org/html/rfc4975.

[118] M. Wuthnow, M. Stafford and J. Shih, IMS A New Model for Blending Applications, Auerbach Publications, 2009.

[119] Google, "Google Play Store," [Online]. Available: https://play.google.com.

[120] QT Project, "Signals & Slots | QtCore 5.1," [Online]. Available: http://qt-project.org/doc/qt-5.1/qtcore/signalsandslots.html.