Masters in Informatics Engineering

Internship 2015/2016

Final Report

Online context for voice communications

José Manuel Marques Grilo

jgrilo@student.dei.uc.pt

Supervisors:

Jorge Sousa
Luís Matos

Carlos Bento

July, 1st 2016

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

Department of Informatics Engineering
Faculty of Sciences and Technology
University of Coimbra
Pólo II, Pinhal de Marrocos, 3030-290 Coimbra

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

WIT Software, S.A.
Centro de Empresas de Taveiro
Estrada de Condeixa, 3045-508 Taveiro, Coimbra

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

Candidate:
José Manuel Marques Grilo
jgrilo@student.dei.uc.pt, jose.grilo@wit-software.com

DEI Supervisor:
Carlos Bento
bento@dei.uc.pt

WIT SOFTWARE Supervisor:
Jorge Sousa
Jorge.sousa@wit-software.com
Luís Matos
luis.t.matos@wit-software.com

Acknowledgements

First, I would like to express my gratitude to WIT software for giving me this opportunity and providing me this great experience over almost a year.

I would like to thank to my supervisors, Jorge Sousa, Carlos Bento and Luís Matos for their support, patience and time spent helping me the entire internship. Their push was one of the main reasons for my success.

I also would like to thank to all my friends for the great and some bad moments, especially to the ones that shared all the difficulties of this year. To all a big thank you.

Finally, I would like to thank to my family for all the support, in particular to the ones that made this possible, my parents. Thank you for the sacrifices you made, for raising me and my brother the way you did and for helping me to achieve my goals. I owe everything I am today to you both, thank you.

Resumo

Em qualquer serviço na internet é importante garantir que o serviço ao cliente é eficaz e pessoal. Para tal, torna-se indispensável a utilização de ferramentas que permitam que o utilizador tenha o melhor seguimento por parte do fornecedor de serviços.

Atualmente, já existem serviços de *helpdesk* e *chat*. No entanto, estes não são eficazes nem proporcionam uma experiência agradável ao consumidor uma vez que, sempre que o utilizador é atendido tem de explicar o contexto da sua questão ao operador.

O principal objetivo deste projeto é a criação de um produto demonstrável com qualidade e diferenciador de modo a atrair potenciais clientes.

Este produto irá permitir a integração de um *widget* num sítio web. Este *widget* dotará o sítio com capacidades de *chat* e comunicação por voz e ainda recolha do contexto da comunicação, que trará um atendimento pessoal e eficaz por parte do operador aos utilizadores.

O presente relatório tem como finalidade apresentar todos os processos e fases envolvidas na execução do projeto referido anteriormente.

Abstract

In any Internet service it is important to make sure that the customer service is effective and personal. In order to do so, it is necessary that the use of tools allow the customer to have the best experience and follow up by the service provider.

Nowadays, there are helpdesk and chat services. However, they are neither effective nor give a pleasant user experience because they are slow and boring to the customer – the customer is attended by one operator, then redirected to another meanwhile all the context gets lost.

The main purpose of this internship is to create a high quality demonstrable product, which is different from the competition and that will attract potential customers.

The internship product will allow the integration of a widget for websites. This widget will give the website the capabilities of chat and voice communication. This will also allow context collection, which will give a personal, fast and effective response to the customers.

The present report has the purpose to present all the processes and steps related with the planning and execution of the above referred to product.

Keywords

“WebRTC”, “Context collection”, “Live Support Software”, “Communication”, “Real-time communication”, “Widget”, “Chat”, “Voice Communication”

Index

Index of Tables

Index of Figures

Glossary

API	An API is a set of software methods used by third parties in order to use a software as a service.
DoD	Refers to a document, which establish the requirements to meet in order to state a User Story as concluded.
Framework	It is a set of methods and functionalities that help the software development.
Macro	Rule or standard that specifies a certain sequence of characters.
Product Backlog	It is a set of all the defined User Stories defined in a SCRUM project.
Product Owner	Responsible for Product Backlog management in a SCRUM project.
Snippet	Small piece of code ready to be copied and pasted to integrate a widget at the page.
SCRUM	It is an agile development framework methodology.
Sprint	It is a time established number of weeks (between 1 and 4 weeks) in which the development team should develop a set of features from the Product Backlog.
Widget	Small application with limited functionalities with a secondary role that only take a portion of the page and does something useful with the page’s information.

Acronyms

API	Application Programming Interface
BO	Back Office
CBD	Component-Based Development
CPQ	Configure, Price and Quote
CPU	Central Processing Unit
CRM	Customer Relationship Manager
CSS	Cascading Style Sheets
DOM	Document Object Model
ERP	Entity Resource Planning
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
ICE	Interactive Connectivity Establishment
IDE	Integrated Development Environment
IP	Internet Protocol
JS	JavaScript
MVVM	Model-View-View-Model
OCVC	Online Context for Voice Communications
OO	Oriented-Object
OS	Operative system
POC	Proof of Concept
RUM	Real User Monitoring
SDK	Software Development Kit
UI	User Interface
UML	Unified Modeling Language
URL	Uniform Resource Locator
UUID	Universal Unique Identifier
VoIP	Voice over Internet Protocol

1 Introduction

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

This work was supervised by Jorge Sousa and Luís Matos, engineers at WIT Software, SA and Carlos Bento, PhD professor at Department of Informatics Engineering of the University of Coimbra.

This internship took place at WIT Software, a software development company specialized in rich and unified communications for Mobile Operators and Mobile Internet companies.

This introductory chapter is divided into four sub-sections. The first sub-section presents the context of this internship in the area of the communication and live support online. The second sub-section presents the motivation that originated the need for this internship and why it is important to the company. The goals of the internship are presented in section three. The last sub-section presents an overview of the entire document and its structure.

1.1 Context

During the past years we have seen a society that is strongly dependent on the internet and the services it provides. The internet solutions, services and communications reached a new level, providing new experiences to the end user and new opportunities to service providers. Nowadays, it is quite easy to develop and launch applications that provide communication services. This results in saturation of the online communications market. To fight the saturation, communication providers had to find new features so that their services could stand out on the market.

Besides offering communication, service providers started using its communication services as analytics and context collectors to help them find the most relevant customers. This kind of communications is known as Live support software. These applications are embedded in the customer’s website and launch a communication window. Initially, the focus was on text but, more recently, they have expanded to voice and video in order to give the user a better and more personal communication experience.

With the internet commerce growing, the use of live support software is very tempting because they provide the website with a communication and tracking tool. Even if the user does not want to talk to the operator, the operator will always know his moves.

Internet entrepreneurs want an easy and cheap way to get and maintain clients. Live support software gets the job done, because it helps operators to give a personal assistance while keeping human cost down.

1.2 Motivation

Introducing a Live support online system in the WIT Software, SA’s set of products would be a great plus for the company. Due to the main type of clients of the company, the internship product can be of great interest for them. Live support systems can help improve the relationships between providers and customers and improve the sales rates.

To sum this up, the motivation of this internship is to enrich the company's set of products with one potential product. This innovative product allows the company to stand out by offering a solution with chat and voice communication that allows the context collection.

1.3 Goals

For many students, the internship is the first contact with the working world. The supposed goals of this project can be divided in two major perspectives: internship goals and internship product goals.

1.3.1 Internship

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

1.3.2 Internship Product

The company is making an investment in the trainees, providing all the resources needed to develop software with quality. Therefore, in this view, the main goal is to finish the project successfully.

The company expects a high quality product that can be used as a Proof and Demonstration of Concept, and be presented to potential clients. At the end of the internship, products aims for three major goals:

- Collect user context : track pages' Uniform Resource Locator (URL), browser, operative system (OS) and location's information;
- Communicate via chat and voice with WebRTC [1] between a widget planted in a customer's website and an operator's back office;
- Help users navigate in the customer's website with URL push.

So, to summarize, the aim of the company is to make an investment in the trainees work in order to receive internship products that can add value to the company.

1.4 Document Structure

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

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

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

- Appendix A – State of The Art: In this document, a detailed description of the current solutions and technologies is presented.
- Appendix B – Approach: This appendix details the software development methodology used and the planning of the project.

- Appendix C – Solution Architecture: Details the solution architecture, development patterns, data model and communication interfaces used on the project.
- Appendix D – Development: This document describes the development done during the internship, the difficulties, tests specification and results and the future work planning.

2 State of the Art

This chapter intends to expose detailed information about the competitors analyzed and about the technologies that will be used.

The document will be divided in three big sections. The first one will give the reader bases about Live Support Online including its goals and main functionalities.

In section 2.2, market analysis is performed about competitors of the internship software, whether direct or indirect. This analysis aims to study the market's saturation level, possibilities of success, impact level and inherent risks.

Lastly, a technology analysis was performed and is shown at section 2.3. Like section 2.2, this section aims to study the technologies available and evaluate their advantages and limitations in order to select the ones to use.

For both market and technologies analyses a brief description of the services/technologies, relevant features and main limitations were studied. Each section contains all the studied services or technologies, sorted in alphabetical order.

2.1 Live Support Online

Stated as an evolution from the helpdesk systems, the Live Support Online systems allow to the customer an immediate personalized service while he keep browsing the website. Unlike helpdesk systems where the service is not immediate because the customer needs to go to an assistance store or has to wait for a callback to be made or an email to be replied, in the Live Support Online this attendance is made through a popup window that allows the customer to chat or make a Voice over Internet Protocol (VoIP) call by the time the problem arise.

This system has two components:

- A chat window through which the customer communicates.
- A dashboard which allows the operator to talk back to the customers.

The chat window is usually injected through a snippet (e.g.: JavaScript (JS) code) which is pasted at the website source code and the dashboard can either be native or web developed.

The start of the conversation can be done following two approaches:

- Broadcast: the conversation is initiated by the customer that sends a message to the server and then it broadcast the message to all the available operators, which then one of them is selected to answer the call.
- Proactive engagement: in this approach the operator is who starts the conversation with a selected customer. Once the operator sends a message it will be delivered to the customer and then he opts to answer or not. The beginning of the conversation is usually defined by a customer achievement, which was specified by the operator (e.g.: number of clicks in the website). This approach is especially effective in sales due to its brute force approach.

More innovator approaches, besides providing an immediate attendance allows a personalized attendance as well. They use techniques such as Real User Monitoring (RUM), which allow them to collect data about their customers and customers' navigation. This approach makes the service more efficient avoiding the context exchange between customer and operator.

2.2 Competitors

The competitors against the internship product will be described in this section. Two groups can be created:

- Direct competitors: it was stated as a direct competitor the company that presents a service or application with at least some of the features that the internship product aims to develop.
- Indirect competitors: it was stated as an indirect competitor the company that presents a service or application with different features from the ones that the internship product aims to develop, yet can still offer a service that can compete against the internship product

To see a full detailed analysis with all the 17 direct competitors analyzed and the competitors' main features, please refer to **Annex A, chapter 3 – Competitors**.

2.2.1 Direct Competitors

a) Competitors Analysis

Nowadays, there is a set of companies offering competitor service like the one proposed in the internship. For a matter of convenience, and since that there are many similarities in the services, only the most popular companies were selected. The companies were selected based on the amount of references in the search.

Below there is an analysis of the five most important direct competitors, based on the features related with tracked data and communication channels offered. Their comparison is done at the next section.

Vivocha [2]

Vivocha is a self-called startup with offices in San Francisco, California, USA, Milan and Cagliari, Italy. It was founded by Gianluca Ferranti and Federico Pinna, currently the company's CEO and CTO.

Beyond its monitoring features, a large set of communications combinations are offered with the possibility of chat, voice, video and callbacks.

Main features:

- Real time monitoring.
- Web dashboard to monitor and assist multiple customers.
- Multiple communication options: chat, voice, video and callbacks.

Livezilla [3]

Livezilla GmbH was founded in 2009 and is based at Singen, Germany.

The company offers a set of integrated solutions of customer support. This set includes a live chat for communication between customers and operators with the possibility of proactive engagement by the operators, real time monitoring and a tickets system to allow offline contact and customers support. Besides that, operators can communicate with each other, both from their dashboard and mobile app.

Main features:

- Real time monitoring.
- Helpdesk ant ticket system for offline service.

Negatives:

- The widget needs to be downloaded and installed.
- Do not allow voice calls.

ClickDesk [4]

ClickDesk is a self-funded company based in Silicon Valley. The company was founded after a search made for a live support chat with the ability to receive phone calls instantly.

Besides the chat and phone abilities, the company intended to keep the service cloud-based.

Main features:

- Multiple communication options: chat, voice.
- Multiples Customer Relationship Manager (CRM) and salesforce software integrations available.

Negatives:

- Do not track customers' heat map.
- Do not allow URL push to the client.

WhosOn [5] **by Parker Software** [6]

Parker Software was founded in 2003 and it is a company based on real-time communication. Its main product is WhosOn, first launched in 2002 with the purpose to track and analyse customer's journey across a website. In 2007, WhosOn was developed into a live chat and tracking solution, used to analyze, engage and chat with customers.

Main features:

- Real time monitoring.

- Possibility of screen sharing.
- Possibility of voice calls.

Negatives:

- Do not allow URL push to the client.

Live Guide [7] **by Netop** [8]

Netop employes 130 people and has subsidiaries in the USA, Great Britain, China Romania and Switzerland and it is headquartered in Denmark.

The company believe in a world where people connect with anyone, anywhere, anytime. For that reason they have three products oriented to online connection: Live Chat with chat, voice and video call, Secure Room Control for remote access and Classroom Management with screen sharing and student supervision.

Main features:

- Real time monitoring.
- Possibility of screen sharing.
- Possibility of voice calls.

Negatives:

- Do not allow URL push to the client.

b) Relevant features

Live support online systems allow communication between a customer and operator in a way that the customer can clear his doubts and questions immediately. However, with the technology evolution and the rise of new concepts to satisfy the customers' demand, new features need to be added. This will allow for a simpler, faster and more efficient attendance system that requires less time and effort both from customers and operators.

To make a complete analysis from the competitors a features' survey was performed. It was taken in count all the features that not only allow the customer to communicate with an operator but also the features that allow the operator to collect navigation context among others. However, many of the analyzed features will be out of the scope of the internship, so a small set of features was selected that allows a comparison between the competitors and the internship product.

The set of features created is a set that was considered to be the minimal acceptable set of features to create a product that allowed costumers to ask for help to operators while operators could see the navigation context. This minimal acceptable set must have at least communication features such as chat and VoIP and context collection features in Real Time. Besides that, the customers' widget injection and a web back office dashboard for operator are also requirements.

The selected set was:

- **Web dashboard**: A web dashboard that allows the operator to communicate with the customers and see their navigation data.

- **Proactive engagement**: A chat conversation may be started every time a defined rule is achieved by a customer, for example if the number of page visited is equal to α.
- **Prioritize chats**: The displayed customers assigned to each operator should be listed in a First-In-First-Served way.
- **Page tracking**: The pages visited inside the website must be saved in order to create a heat map for each customer.
- **Referrer tracking**: The referrer to the website must be saved for each visit.
- **Browser Tracking**: The browser version used in every visit must be saved.
- **Geolocation Tracking**: The operator must see the customer's geolocation if they are allowed.
- **Real time monitoring**: The monitoring of the tracked data must be done in real time during the customer's visit.
- **Online customers list**: An online customers list should be provided to every online operator.
- **Recognize customers**: Returning customers must be identified and their previous information should be loaded to the dashboard.
- **Chat**: Customers should be able to start a chat conversation with an online operator.
- **VoIP without plugins**: Customers should be able to start a voice call with an online operator without the need to install any plugin.
- **Operators login/logout**: Registered operators should be able to login and log out at the dashboard.
- **URL Push**: Operators should be able to push pages to their customers' browser and force them to load that page.
- **Multiple chat windows**: Operators should be able to have multiple conversations at the same time.
- **Chat transfer**: Operators should be able to transfer chat conversations between themselves.
- **Proactive engagement rules**: Operators should be able to configure a set of rules to start proactive engagement with their customers.
- **Widget injection**: The widget should be injected in the websites through a snippet, without the need of software installation.

c) Comparative analysis

This comparative analysis uses all the 15 direct competitors analyzed instead of the five present above. Only using the five presented before would lead to wrong conclusions, so all the competitors were taken into account in order to state where the developed product can be innovative and stand on the market.

In order to easily compare direct competitors, first a table with their features developed by each company is presented. After that, a written summary will highlight the main competitors from the ones analyzed and how the internship product can stand in the market.

The table below compares the services of the companies. The check sign (✓) indicates that the feature is supported, on the other hand, the cross sign (✗) indicates that the feature is not supported. Since it is very hard to collect all the data from the companies' products due to the inability to test all software, the circle sign (●) is used every time that no information is supported or available. For a more complete table, with all the features analyzed, please refer to **Annex A, chapter 6, section 6.1 – Full Competitors Comparative Table**.

	Livezilla	Tawk.to	Vivocha	Provide Sup-	Zopim	BoldChat	Live Help Now	Kayako	Olark	LiveChat	ClickDesk	Comm100	Netop Live	WhosOn	website Alive
Dashboard															
Responsive dashboard	✗	●	✓	●	●	✗	●	✗	✗	●	✓	●	✗	●	●
Proactive engagement	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Prioritize chats	✓	✗	●	●	●	✗	●	●	●	●	●	●	●	●	●
Widget															
Page tracking	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Referrer tracking	✓	●	✓	✓	●	✓	✓	✓	●	✓	✓	✓	✓	✓	✓
Browser tracking	✓	●	✓	✓	●	✓	✓	✓	●	●	✓	●	●	●	●
Geolocation tracking	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Real time monitoring	✓	✓	✓	✓	✓	✓	✓	✓	✗	●	✓	✓	✓	✓	✓
Online customers list	✓	✓	●	✓	✓	●	●	●	✓	●	✓	●	✓	●	●
Recognize customers	✓	✓	●	✗	●	✓	✓	●	●	✓	●	✓	✓	●	✓
Customer															
Chat	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
VoIP without plugins	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	●	✓	✗
Operator															
Operators login/logout	●	✓	✓	●	✓	●	●	✓	●	●	●	✓	●	✓	●
URL push	✓	✗	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	●
Multiple chat windows	✗	●	✓	✓	✓	●	●	●	✓	✓	✓	✓	●	✓	✓
Chat transfer	✓	✓	✓	✓	✓	●	✓	✓	●	✓	✓	✓	✓	✗	✓
Proactive engagement rules	✓	✓	✓	✗	✓	✓	●	●	✓	●	●	✓	✓	✓	✓
Owner															
Widget injection	✗	✗	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 2.1 Direct Competitors Features

Among the features between them, the final result presented is similar when compared, for example the proactive engagement allows operators to start a chat conversation without the customer allow to, through operator's initiative or the achievement of some kind of metric.

As stated previously, it is difficult to know exactly the features that are presented in each competitor due to the difficulty to test all services. Some features like the tracking ones are difficult to know if they are present because some documentation only states that the widget tracks visited pages among other information.

After evaluating the market we can see a small niche that is not well explored. The niche is the VoIP calls, which is one of the main features to develop in the internship. This could be a huge opportunity to explore that can bring value and be a point of differentiation.

2.2.2 Indirect Competitors

In this section will be analyzed the indirect competitors, those who in some way offer a possible solution to the customers even if that it is not their purpose.

The indirect competitors were divided in two groups: context analysis software and Customer relationship management software.

a) Context analysis software

In this subsection will be analyzed context analysis software. This software collects a set of data, useful from the website proprietary point of view, which allows the user to access their websites stats about visits.

The set of companies that were chosen was based on the number of references presented in the search.

Google Analytics [9]

In 2005 Google Inc. bought Urchin. This move led to the creation of Google Analytics. Its success was huge, which made new requests to the service to stay suspended after the first week. Since that, new requests were attended in a lottery way, until 2006. Since September 2006 the system became available to all users. Nowadays, Google Analytics is used over 55% of the 10000 more popular sites and over 49.95% of the 1000000 first sites from the Alexa rank [10].

Adobe Marketing Cloud [11]

Omniture was founded by Josh James and James Pestana, and financed by venture capitalists. Becoming one of the greatest powers, among the 500 private companies with the biggest growth rate and nominated by Inc. Magazine [12]. The company was acquired by Adobe Systems in 2009, and until 2011 operated as a company inside Adobe. In 2012 Adobe started to remove the name Omniture and its products started to be integrated on the Adobe Marketing Cloud.

Kissmetrics [13]

Founded by Neil Patel and Hiten Shah in 2008, it is based in San Francisco, California.

The company offers a set of tools to collect data about how the customers interact with websites, web apps and mobile products.

b) CRM Software

According to Philip Kotler, conquering new clients can cost 5 to 7 times more than maintaining the existing ones [14]. For that reason companies should have strategies that allow loyalty from their clients.

Customer relationship management is a strategy that focus on understanding and anticipating the client needs. CRM is used to help in this strategy, which allows:

- Client management
- Identify and define client profiles
- Manage communication with client
- Follow orders
- Anticipate the market evolution
- Organize a personalize technical assistance

This section will analyze the four CRM software products with the bigger market share [15].

Salesforce [16]

Salesforce.com is a cloud computing company, based in San Francisco, California, USA. Founded by Marc Benioff, Parker Harris, Dave Moellenhoff and Frank Dominguez in 1999. Today it is one of most valuables cloud computing companies, with an estimated value of 50 billion dollars [17].

The company splits its products into six areas: sales, marketing, community, analysis and applications. Its products are used by over 100000.

Market Share: 18.6%

Growth: 28.2%

SAP CRM [18]

SAP SE is a German software company that makes enterprise software to manage business operations and customer relationships. Its CRM software targets midsize and large organizations in all industries.

Its solutions includes the modules of sales, marketing, services, analytics, interaction center and web channel.

Market share: 12.1%

Growth: 7.2%

Oracle CRM [19]

Oracle started in this market space when they purchased Siebel Systems in 2005, a company focused on developing software to support CRM. Later, they bought Upshot CRM, which brought a more robust interface.

The company split its products in six areas: marketing, sales, commerce, social, services and CPQ (Configure, Price and Quote). Its customers can either subscribe to a single module or a combination of modules.

Market share: 9.2%

Growth: 2.6%

Microsoft Dynamics CRM [20]

Microsoft Dynamics CRM is part of the Microsft Dynamics package of CRMs and Entity Resource Planning (ERP). Launched in 2003, it is a client-server accessable apllication from both browser and a plugin for Microsoft Outlook. The current version count with sales, marketing, services and social modules. It counts about 40,000 customers.

Market share: 6.2%

Growth: 21.7%

c) Summary

After the analysis of the indirect competitors, neither the Relationship management and context analysis software nor the CRM software are real menaces to the internship product, because its focus is not the one to track and assist. The first group is focused on data collection to context from the visits done to the website, however they cannot give assistance to the users. The CRM group, can give some assistance in real-time but its real focus is to collect and manage relationships in order to improve the communication and assistance of the customers.

In spite of not being a big menace, the integration of CRM software in the internship product should be considered. This integration can be useful to provide a better experience and assistance to the customer. If so, the first CRM software to consider should be Salesforce due to its market share, and, still show a greater growth than its competitors. Due to their market share, the other CRM software products are not a priority.

2.3 Technologies

Choices of technologies can influence a project. Choosing an unfamiliar or untested technology can bring unwanted risks to the project. On the other hand, choosing the right ones can make the development process easier with much better final results.

In order to choose the right technologies the intern made an analysis for the main components developed: Online Context for Voice Communications (OCVC) Server and Back Office (BO) Web App. Besides that, the database selection was also an important decision.

The intern had full liberty to analyze and choose the technologies. After making his analysis the intern presented his choices to Internship's Tutor Luís Matos who accepted the proposed technologies.

This section will be show the technology choices to develop the internship product. First to be presented is the list of the backend technologies, and second is a subsection of the frontend technologies.

The third subsection will present the database selected.

Finally a section about WebRTC is presented.

For a full technologies analysis, please refer to **Annex A, chapter 4 – Technologies**.

2.3.1 Backend Technologies

JavaScript on Node.js [21]

Node.js is a recent cross-platform runtime environment used for server-side applications written in JavaScript. It provides an event-driven architecture and a non-blocking I/O Application Programming Interface (API) ideal for real-time web applications. Besides its recent creation, in 2009 by Ryan Dahl and Joyent, it is already in use by some great companies such as LinkedIn, IBM or Microsoft.

Since the goal of the internship product is to build a web application that allows real-time communication between clients and operators without demanding Central Processing Unit (CPU) processing. The internship product will be developed as a multi-tier chat application – two different web clients (clients and operators) and a web server – where the clients' application will communicate between themselves through the server. So, the server will be a lightweight application that only needs to forward information every time a new request arrives. What is needed is an event-driven architecture with a fast non-blocking API, which is what Node.js offers.

Applications that run on Node.js are written in JavaScript, the same language can be used both on the client side and on server side.

The use of Node.js allows the use of several modules from the package modules repository – npm [22] – such as:

- Socket.io [23]: is cross-platform module that allows real-time bidirectional event based communication, which is great to establish communications between users (clients and operators) and server. This is an ideal module for real-time analytics and chat application. Besides that, it allows communication both via sockets or polling.
- Express [24]: is a Node.js framework that provides a robust set of features for web applications development, designed to quickly build a single-page, multiple-page or hybrid web applications.

2.3.2 Frontend Technologies

AngularJS [25]

AngularJS is a JavaScript framework, written in JavaScript that extends Hypertext Markup Language (HTML) with directives. It enables the creation of dynamic views in web applications, through a two-way-data-binding, which automatically synchronize both models and views. This abstraction from the Document Object Model (DOM) leads to better code decoupling (view separated from model and controller).

AngularJS enables the creation of dynamic applications with several views without too much complexity.

2.3.3 Database

Database choice is a critical decision to make in every project. Every database has its pros and cons and they should be taken into account when making this choice.

However, database dependency can be minimized with the right architectural choices. In the current project server architecture follows a component-based development pattern (see subsection 4.1.4). This pattern makes it easier to replace the database by only changing the database connector, which is an abstraction module to communicate with the database.

This does not mean that the database choice was made without any analysis. Below are the main reasons for the database choices.

a) NoSQL Databases

NoSQL (“non SQL” or “Not only SQL”) provides a storage mechanism modeled as a means other than well-defined schemas. Their data can be persisted in a collection of key-value pair, graph model or wide-columns stores which do not have a schema defined. NoSQL databases are horizontal scalable which means that they scale if more nodes are added.

Since they use a dynamic schema, they are well-suited for problems with unstructured data, hierarchical data or big data.

b) Analysis

Both SQL and NoSQL databases have its advantages.

In this project a little amount of relationships are needed, since only operators, visitors, conversations and tracked data entities exist. This kind of relationship would indicate the use of a SQL database. However, the structure of the entities is uncertain. More tracked fields could be added, visitors and operators profile could change when more features are added to the product. That is the most important reason to use a NoSQL database. This free-schema paradigm allows a fast development, which is critical to this internship. Another good reason to use NoSQL is the huge need for scalability. Tracked data and conversations will make the database size grow fast, so a database that scale horizontal is better. Besides that, no complex queries are needed, so NoSQL is suited to this case.

MongoDB [26] was the database selected. It is a NoSQL database that favors a Javascript Object Notation (JSON)-like documents structure with dynamic schemas. This allows for faster and easier integration of new data. MongoDB also allows a high insert load, which is great to persist data in the database. Since the application is always recording data from visitors and chats can scale quickly, a huge amount of inserts will be done.

Besides its technical advantages, MongoDB and Node.js have a mature connection very tested, and used, which means that they are two technologies that work very well together.

2.4 WebRTC

WebRTC is an open project that tries to standardize the support for real-time communications via simple APIs, allowing web applications to send data between devices over Internet Protocol (IP). Its mission is to offer rich and high-quality applications the power to be developed for devices and allow them all to communicate via a common set of protocols, using JavaScript APIs and HTML5.

Since WebRTC allows real-time communication into websites, sending data and media streams in a peer-to-peer way, it means it can be used for voice and video calls directly from a web browser without the need of plugins. Because of this, many communication solutions can be developed, solutions that can be innovative in the communication area.

Some of the main benefits of WebRTC are related to improved productivity and team collaboration using clientless web-based video inside and outside companies, enhanced flexible work by enabling internet calling and improved relationships with customers with web-based video and video communication.

We can sum up the benefits of WebRTC in five different scenarios:

- Consumer to Consumer (common users call to people they know)
- Consumer to Business
- Business to Consumer
- Within business organizations
- Business to Business

3 Approach

The internship followed a software methodology already adopted by the company, which was ideal to the kind of project. As in any project, there are risks that may affect its outcome, it is important to identify them and create a mitigation plans in order to minimize or eliminate the risks.

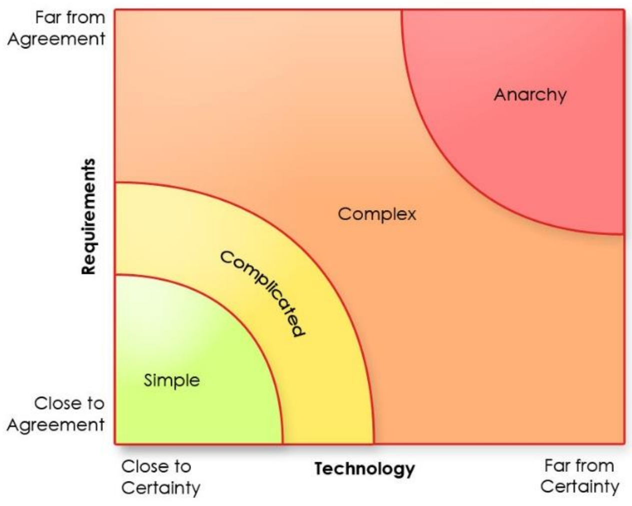
This chapter presents the approach followed; the planning process and the risks and mitigation plans.

3.1 Methodology

This section provides an overview on Scrum, the methodology used during the internship, its roles, events, artifacts and definition of done.

According to figure, there are four kinds of projects:

- Simple: projects with low complexity, where we know both the set of requirements and technologies to use.
- Complicated: projects where there is still some degree of certainty about the requirements and technologies, but with a bit more complexity than the simple projects.
- Complex: projects with a great degree of complexity where there are many possible requirements that demand a huge amount of study in order to get them defined, and there are a huge amount of technologies that may be used.
- Anarchy: projects where there is neither a definition of the requirements or the technologies to be used.



[27]

Figure 3.1. Projects complexity

The solution to the first two kinds of problems can easily be achieved with waterfall-like methodologies. The anarchy kind due to all the uncertainties are hardly named projects. The complex projects, which are the area where the internship problem inserts into, are problems where a fixed planning will not help due to its degree of uncertainty, but are feasible if an agile methodology is used.

“Scrum is a framework for developing and sustaining complex products. A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value” [28]. Scrum provides a flexible and holistic product development strategy with the necessary agility for big projects where a wide set of requirements and technologies is available.

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

3.2 Planning

In the internship, supervisor Jorge Sousa has the scrum master role, being responsible to manage all meetings and solve possible impediments.

Tutor Luís Matos took the product owner role with product backlog management. One of the responsibilities of the product owner is the definition of the requirements, this task was assigned to the intern in order to let him learn the processes of scrum.

The scrum team is composed by the intern José Grilo.

The project planning, like in any other methodology, starts with the requirements definition, however in scrum a requirement has a different name and motivation.

3.2.1 Scope

Defining the project scope is one of the most critical steps in a project. Without knowing what you are supposed to be delivering at the end to the client and what the boundaries of the project are, there is a little chance for the project to success.

A poorly defined scope definition will lead to an impossible management during the project execution.

The main purpose of the scope is to clearly describe the boundaries of the project, according to the client's agreement. The elements within the scope and out of the scope must be well defined in order to clearly understand the area under the project control.

This section will be divided in two areas: The elements within the scope, the project objectives and its goals and the elements out of the scope.

a) Within the scope

Besides the elements that are included within the scope, this section will expose both internship and project goals and the project objective.

From the internship view, the goals are to consolidate knowledge about Software Engineering and gain experience in developing software in a corporate environment where commitment and team work are essential to produce a high quality piece of software. At a technical level the goal is to learn and master the use of web frameworks, such as Node.js and develop a stable and high quality software.

On the other hand, regarding the project, the main goal is to contribute with a proof of concept that can be used for conferences and exhibitions and a posteriorly development of a product. To accomplish this, the following features must be fully implemented and tested to guarantee the existence of zero bugs in the product:

- Injection of a widget: a code snippet must be provided and the client can copy and paste it in his website providing it functionalities such as chat, VoIP and real user tracking.
- Chat: allows both clients and operators to start a conversation between them.
- VoIP: allows the client to start a voice conversation with an operator without the need to install any plugins, using the WebRTC capabilities.

- Real User tracking: the widget must do a full track of the clients' path in real time, tracking his viewed pages and referrer, browser details and geolocation.
- Customers' analysis and browser control: allows the operators to see the clients' information and push pages to their browsers.
- Remote Assistant: WIT's demo of Remote Assistant must be integrated with the developed product as part of its functionalities. This feature was added on second semester in order to bring more value to the developed project. This was a calculated risk which led to small backlog, requirements and architecture changes. However competitors' analysis was not revisited due to time constraints.

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

- Requirements analysis: identify, discuss and prioritize all the requirements.
- Technologies analysis**:** identify possible technologies to use, discuss the pros and cons and choose the one to use.
- Architecture definition: define how the features will work internally, understand which components must be created and how will they communicate with each other within the application.
- Risk analysis: identify possible risks and problems that may arise and trace a mitigation plan that overcomes them.

b) Out of the scope

This section will be specify the elements that were analyzed and are interesting to the project, but will not be implemented in this internship.

- Operators' profile management: allows the operator to manage his personal information such as name or password.
- Customize widget: allows the user to customize his widget appearance (colors, logo, *etc*) before the snippet generation.
- VoIP with Flash or plugins: allows the client to perform a VoIP call in browsers where WebRTC capabilities are not supported.

3.2.2 User Stories

A user story consists in one or more sentences written on small pieces of paper in everyday language that capture the intentions of the user that the program does as part as its job functions. It is used in scrum as the requirements definition because it is an easy and understandable way of handling the requirements without the formal formulation of a document. User stories follow the terminology below:

As a <role>, I want <goal/desire> so that <benefit>

Typically, the user stories are created and managed by the product owner, however, as already stated, the scrum team does the task as a learning process.

Usually, user stories are associated with a category, mainly if there are many features to implement. At the internship seven categories were created:

- **Widget**: consists in the development of features that allow the tracking of the client's information.
- **Widget's owner**: corresponds to the requirements of add and remove operators and widget's injection.

- **Client**: This category relates to all the functionalities a client can do while browsing in a page with the product
- **Dashboard**: consists in the development of the features that allow the operator to communicate, and retrieve all users' information from the server
- **Operator**: This category relates to all the functionalities an operator can do while using the dashboard
- **Server**: corresponds to the development of a server-side application that establishes all the communication with the dashboard, widget and database.
- **Documentation**: consists in the construction of all the documents related to the state of the art, requirements, architecture, demos and internship documentation.

3.2.3 Product Backlog

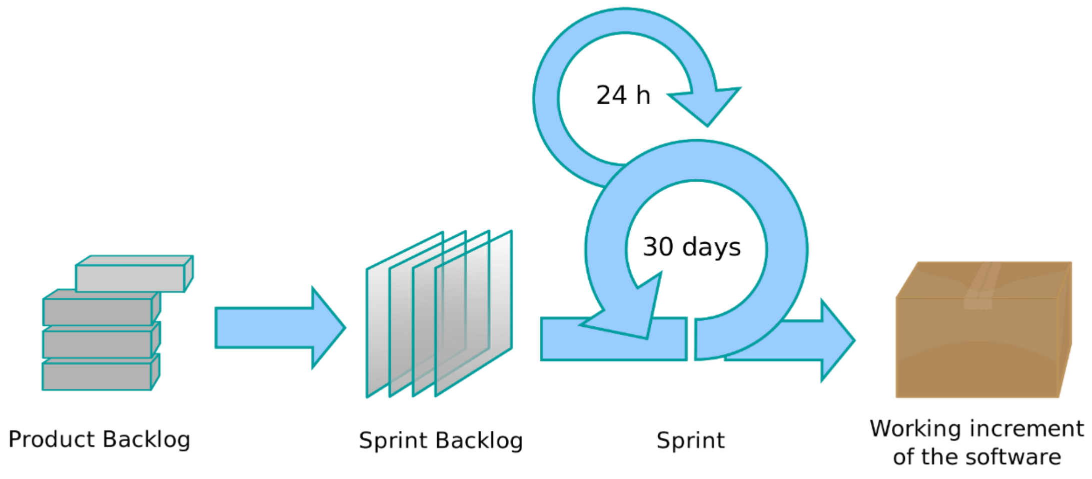
Each of the created user stories corresponds to an entry in the product backlog. Furthermore, each user story was assigned with a priority and its difficulty was calculated using planning poker. Planning poker is a gamification technique that consists in using a deck of cards where each card is assigned with a number from the Fibonacci sequence from 1 to 13 (100 and "?" are also used). It is assigned the value of "2" to make an easier user story, then, the other stories are valued based on the cards played by each participant relative to the first story ("?" means that a value cannot be assigned or more information is needed).

Based on the priorities and difficulties assigned an ordered list was formed. That list was the first version of the product backlog.

To see the current product backlog please refer to **Annex B, chapter 3, section 3 – Product Backlog**.

3.2.4 Sprints

After the project is minimally defined, it is time for the scrum team to work on the product. In an initial stage of the project was done, larger sprints (four weeks), then the sprints started to have a two-week duration.



[29]

Figure 3.2. Scrum Process

Presented here is a short summary of the sprints.

- **Sprint #0**
  Sprint Planning: 14-10-2015
  This was the warm up sprint. It was assigned to the scrum team to do the competitors' analysis, technologies' analysis, formulate the user stories and the "Definition of Done".
- **Sprint #1**
  Sprint Planning: 17-11-2015
  In this sprint it was assigned to the scrum team to design the database structure and install, and setup the database. It was also required to write the requirements document in order to start the User Interface (UI) design.
- **Sprint #2**
  Sprint Planning: 01-12-2015
  This sprint marked the development's beginning. It was assigned to set the working environment (install dependencies and Integrated Development Environment (IDE)) and create the connections between widget/back office and server.
- **Sprint #3**
  Sprint Planning: 16-12-2015
  In this sprint it was assigned to the team the operators' login and logout features and the UI implementation. The widget should also be loaded with the snippet.
- **Sprint #4**
  Sprint Planning: 04-01-2016
  It was assigned to start the context collection features.
- **Sprint #5**
  Sprint Planning: 18-01-2016
  This sprint marked the beginning of BO's development. Features regarding to queues loading and display and customers engaging were implemented. Midterm report was also written during this sprint.
- **Sprint #6**
  Sprint Planning: 02-02-2016
  In this sprint filters on customers' queues were implemented. Operator's engaging feature was finished in this sprint, showing to the customer that he had been selected. Returning customers were also identified on this sprint. This last feature marked the finish of context collection feature set.
- **Sprint #7**
  Sprint Planning: 16-02-2016
  This sprint was dedicated to chat functionalities such as messages exchanging and "is typing" notifications.
- **Sprint #8**
  Sprint Planning: 01-03-2016
  In this sprint URL push was implemented alongside offline form request for customers.
- **Sprint #9**
  Sprint Planning: 21-03-2016
  This sprint served to implement and integrate Remote Assistant demonstration on OCVC project.
- **Sprint #10**
  Sprint Planning: 04-04-2016

This sprint marked the beginning of VoIP calls implementation. Developed here was all the needed modules to make VoIP call for browsers.

- **Sprint #11**
  Sprint Planning: 19-04-2016
  In this sprint the product backlog was revisited and some flows were changed. Also in this sprint mail sending features was implemented.
- **Sprint #12**
  Sprint Planning: 03-05-2016
  This sprint was used to implement VoIP calls for devices.
- **Sprint #13**
  Sprint Planning: 17-05-2016
  This sprint was used to test all the implemented features and bug solving. During this sprint some UI changes were taught and WIT designer Elizabeth Pereira started to work on the project again.
- **Sprint #14**
  Sprint Planning: 31-05-2016
  This sprint was mainly used to final the report writing.

3.2.5 Definition of Done

In order to have shared understanding of "done" a definition was created, both for user story creation and development. Besides that a definition of done to the increment was created too.

The creation of a new user story is stated as "done" if:

1. The user story follow the notation "As a <role>, I want <goal/desire> so that <benefit>";
2. Story sized with thirteen story points or less;
3. Story is divided in tasks and each task has a duration in hours;
4. Story necessity is explained and agreed by all.

A user story development is stated as "done" if:

1. All the tasks related to development are coded;
2. Code commented and meeting company's development standards;
3. Builds without errors;
4. Acceptance tests are written and passing;
5. Code committed on server;
6. Relevant documentation is produced or updated;
7. Remaining hours for story set to zero and story closed.

An increment is stated as "done" if:

1. All modules developed during the sprint are integrated with the previous release;
2. The increment build as no errors;
3. It is ready for demo;
4. Final version is updated to server;
5. A presentation is prepared to present the increment.

3.3 Risks and mitigation plans

There are always risks in every project. Each risk has a probability to contribute to the failure of the project. Identify the risks, its sources and create mitigation plans are the best actions to take. These strategies should be reviewed periodically because the risk probability can grow and new mitigation plan may be needed.

The risk identification is based in factors that can cause project failure and their analysis is based on the probability of occurrence presented on Table 3.1, the impact on project if the risk occurs presented on Table 3.2. It is also important to take into account when the possibility of the risk itself.

Percentage	**<30%**	**30% - 50%**	**50% - 75%**	**> 75%**
Probability	Low	Medium	High	Very high

Table 3.1. Risk occurrence probability

Impact	**Description**
Low	Project success is not compromised
Medium	Project success is not compromised, however small adjustments are required so that risks do not evolve
High	Project success can be compromised if no adjustment and additional effort is done
Very High	Project success can be seriously compromised

Table 3.2. Risk associated impact

Occurrence forecast	**Description**
Short-term	Risks can occur in an initial project phase, during the first development weeks
Mid-term	Risks can occur in an intermediate project phase, during the development
Long-term	Risks can occur in a final project phase, after the development

Table 3.3. Risk occurrence forecast time

This is an up-to-date living list with the risks identified, their probabilities, impacts, occurrence forecasts, consequences and mitigation plans.

ID	**RK_01**
Name	Frameworks updates
Probability	Low
Impact	High
Occurrence forecast	Medium-term
Consequence	Deprecated code and/or system failures
Mitigation plan	Regular code reviews in order to look for deprecated code and bugs

Table 3.4. Risk 01 - Frameworks updates

ID	**RK_02**
Name	Unreachable frameworks' servers
Probability	Low
Impact	High
Occurrence forecast	Medium-term
Consequence	System failures
Mitigation plan	Frameworks should be ready to be served from internship server

Table 3.5. Risk 02 - Unreachable frameworks' servers

ID	**RK_03**
Name	Poorly defined requirements
Probability	Low
Impact	Medium
Occurrence forecast	Short-term
Consequence	Delay or failure in the requirements
Mitigation plan	Requirements documents should be reviewed and approved by scrum master

Table 3.6. Risk 03 - Poorly defined requirements

ID	RK_04
Name	Requirements’ changes
Probability	Medium
Impact	Medium
Occurrence forecast	Medium-term
Consequence	Delay or failure in the requirements
Mitigation plan	Stakeholders should be present regularly at sprint review to approve the work done so far

Table 3.7. Risk 04 - Requirements' changes

ID	RK_05
Name	Bad planning
Probability	Medium
Impact	Very High
Occurrence forecast	Medium-term
Consequence	Delay or failure in the requirements
Mitigation plan	Standup meetings (daily scrum) should be performed every day in order to see the current difficulties

Table 3.8. Risk 05 - Bad planning

ID	RK_06
Name	Not meeting stakeholders’ expectations
Probability	Low
Impact	Very High
Occurrence forecast	Long-term
Consequence	Delay or failure in the requirements
Mitigation plan	Stakeholders should be present regularly at sprint review to approve the work done so far

Table 3.9. Risk 06 - Not meeting stakeholders' expectations

ID	**RK_07**
Name	Technologies learning curve
Probability	Medium
Impact	High
Occurrence forecast	Short-term
Consequence	Delay or failure in the requirements
Mitigation plan	At planning execution time should be allocated by taking into account the time needed to learn the technologies to use

Table 3.10. Risk 07 - Technologies learning curve

ID	**RK_08**
Name	Strong market competition
Probability	High
Impact	Medium
Occurrence forecast	Long-term
Consequence	Project failure
Mitigation plan	Constant market analysis in order to know the competitors and how can the internship product differentiates from them

Table 3.11. Risk 08 - Strong market competition

To prioritize the identified risks and know which are the most urgent to fix was used Pareto's Top N strategy [30] based on the occurrence, impact and occurrence forecast of each risk.

The urgency level of each risk was determined with the analysis of Table 3.12.

		Impact			
		Low	**Medium**	**High**	**Very High**
Probability	**Very High**	🟨	🟧	🟧	🟥
	High	🟩	🟨 RK_08	🟧	🟧
	Medium	🟩	🟩 RK_04	🟨 RK_07	🟧 RK_05
	Low	🟩	🟩 RK_03	🟩 RK_01, RK_02	🟨 RK_06

Table 3.12. Risk exposure

Color	🟩	🟨	🟧	🟥
Risk Exposure	Low	Mediu	High	Very High

Table 3.13. Table 3.12's color code

From the table below it is possible to analyze the risk exposure to each risk. The green risks are the ones that will not jeopardize the project either because they have a low probability or a low impact. On the other hand, red risks are the most dangerous and will cause the project failure.

To prioritize the risk list it was taken into account both risk exposure and occurrence forecast to each risk.

Risk ID	Risk Exposure	Occurrence forecast
RK_05	High	Medium-term
RK_06	Medium	Long-term
RK_07	Medium	Short-term
RK_08	Medium	Long-term
RK_04	Low	Medium-term
RK_01	Low	Medium-term
RK_02	Low	Medium-term
RK_03	Low	Short-term

Table 3.14. Risk prioritization based on exposure and occurrence forecast

Lastly, it is important to know which risks are important to mitigate. Those are found based with their probability and impact. The risks that are important to mitigate are those who have a high or very high probability and a high or very high impact, which means that only RK_05 needs to mitigate.

4 Solution Architecture

The architecture is an essential component of any software project. It defines the structure and behavior of the components, how they interact between themselves, while helps to hold the non-functional requirements and defines a guideline to implementation.

The current chapter is divided into two sections. Section one is the system architecture, that the architecture is designed and justified. Section two is the data model, with the document references for the documents presented at the database.

4.1 System Architecture

The model used to represent the system architecture was an adaptation of the C4 Model by Simon Brown [31]. The C4 Model [32] divides the architecture in four levels:

- System Context: The system plus users and system dependencies
- Containers: The overall shape of the architecture and technology choices
- Components: Logical components and their interactions within a container
- Classes: Component or pattern implementation details.

This architecture representation was chosen mainly because it was concluded to be complete enough to present the system to all involved clearly. Another reason was the limitations given by the technologies, it would be very hard to create an architecture representation using Unified Modeling Language (UML) diagrams while using Javascript language mainly.

C4 Model allow the creation of four levels of representation, each level could be used to present the architecture depending on the interested. Besides that, C4 uses a simple ubiquitous language that everyone can understand easily.

The first and second levels show who will use the product and how its components are used to construct it. These levels are the most abstract and easily give the overall shape and context of the product. They can be used to present the product to newly arrived stakeholders or potential interested customers.

Third and fourth levels are more technical and could be used by development team as a development guideline to ensure non-functional requirements. Here are presented the components inside each container, their interactions and the implementation patterns used.

4.1.1 System Context

System context is the highest level of abstraction and represents something that delivers value to somebody. A system is made up of a number of separate containers.

The system context is meant to answer three questions:

- What are we building?
- Who is using it?
- How does it fit into the existing environment?

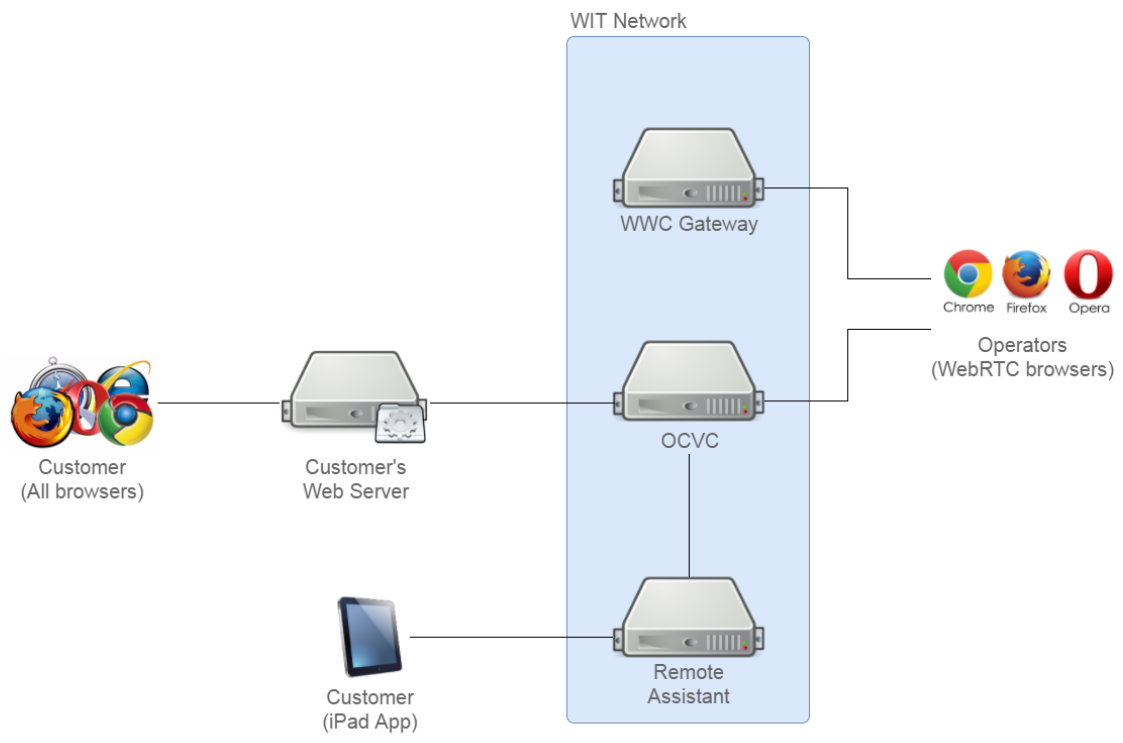


Figure 4.1. System context

With the previous diagram it is clear the kind of system that will be build and the actors who will use it.

This kind of view is suited for non-technical audience. It is clear to the ones who see this view that the system will be used, on a first-level by operators. On a second level, customers will interact with a website that connect with the system or with an iPad App connected to OCVC server by Remote Assistant server.

The OCVC system, will serve a widget to customer's website server. This widget will make the context collection and connection with the web server. On the other hand, it will serve a web application as well in order to operators to manage customers.

Regarding to actors there are two groups. One group of actors are the operators, they will interact with the system via a web application, pushing and pulling information from it that will enable them to see the customers that are using the websites, their tracked info and communicate with them. The other group of actors is the customers, which can be broken in two categories: web customers and iPad Customers. Web customers will not interact directly with the system. Instead, they will interact with the websites that get the widget from the system. The widget will track customers' information and allow them to communicate with operators. iPad customers interact with WIT's Remote Assistant application which allow them to call to operators at back office.

OCVC server is deployed in WIT's Network, alongside WWC Gateway and Remote Assistant server, which act as project's dependencies to implement breakout and iPad call respectively. Widget will be injected in customer's website.

4.1.2 Containers

If we lower one level in the C4 Model we are looking for:

- What are the high level technology decisions?
- How do containers communicate with one another?
- As developer, where do I need to write code?

A container represents something in which components are executed or where data resides. This could be anything from a web or application server through to a rich client application or database. Containers are typically executable that are started as a part of the overall system, but they don't have to be separate processes in their own right.

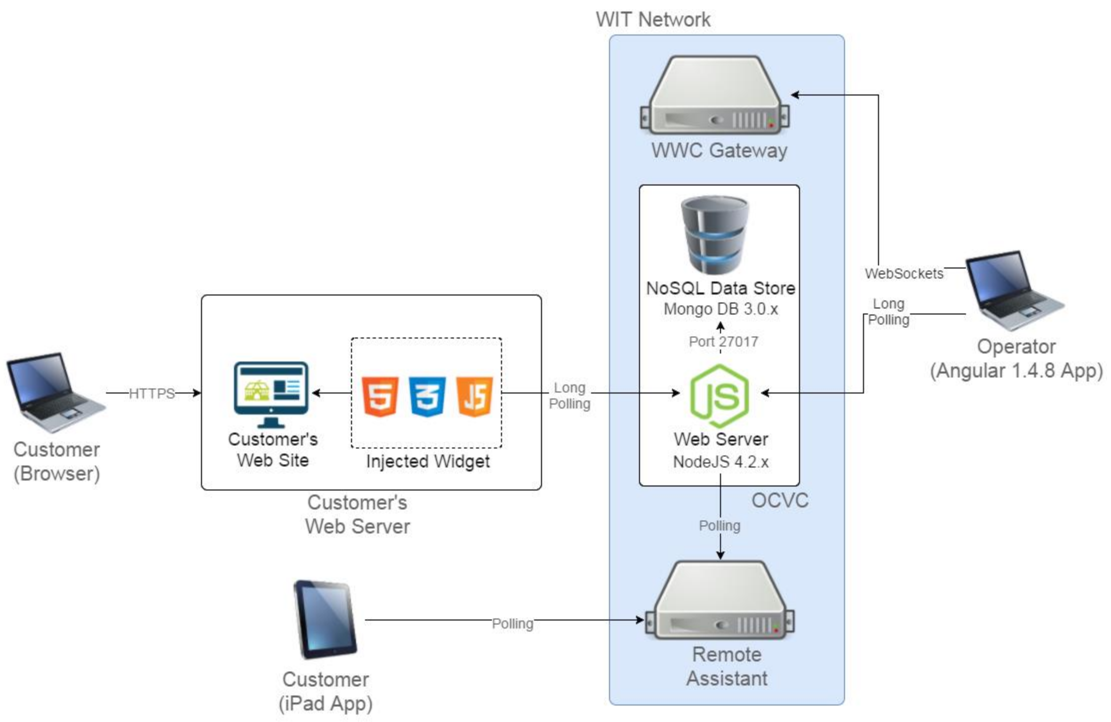


Figure 4.2. System containers

This view is suited for people that are semi-technical. Here they can see technologies choices, what containers will compose the system and how will they communicate between themselves.

At the technologies choices, the system will hold a Node.js server and a MongoDB.

The communication with the system will be done via HypertText Transfer Protocol Secure (HTTPS) using self-signed certificates. Inside the system the web server will communicate with the data store that will run at the port 27017.

After UI is served communications proceed by Long Polling between Widget/Web App and OCVC server. To communicate between OCVC server and Remote Assistant is used Polling. This communication method was not changed due to the complexity to change an existent server while there were another priority features to develop. Between Web App and WWC gateway communication is made by WebSockets.

The web server will hold all the logic. The web server will take the request to widgets and response with the desired one, and will take requests from operators to access and manage the back office application. Besides that, the web application will also write and read from the MongoDB, that will store operators and customers' information, tracked data and sent messages.

User Interfaces, customers and operators, will both use HTML, Cascading Style Sheets (CSS) and JS. However selected frameworks will be different. Will use jQuery on Widget, Socket.io and Bootstrap. These frameworks were chosen because it was to inject them at Customer's web pages without breaking them and would fast development process. On operator's back office, Bootstrap, AngularJS and Socket.io as working frameworks were mainly used.

4.1.3 Components

A component can be thought of as a logical grouping of one or more classes. Components are typically made up of a number of collaborating classes, all sitting behind a higher-level contract.

At components level we need to answer to:

- What components/services is the container made up off?
- Are the technology choices and responsibilities clear?

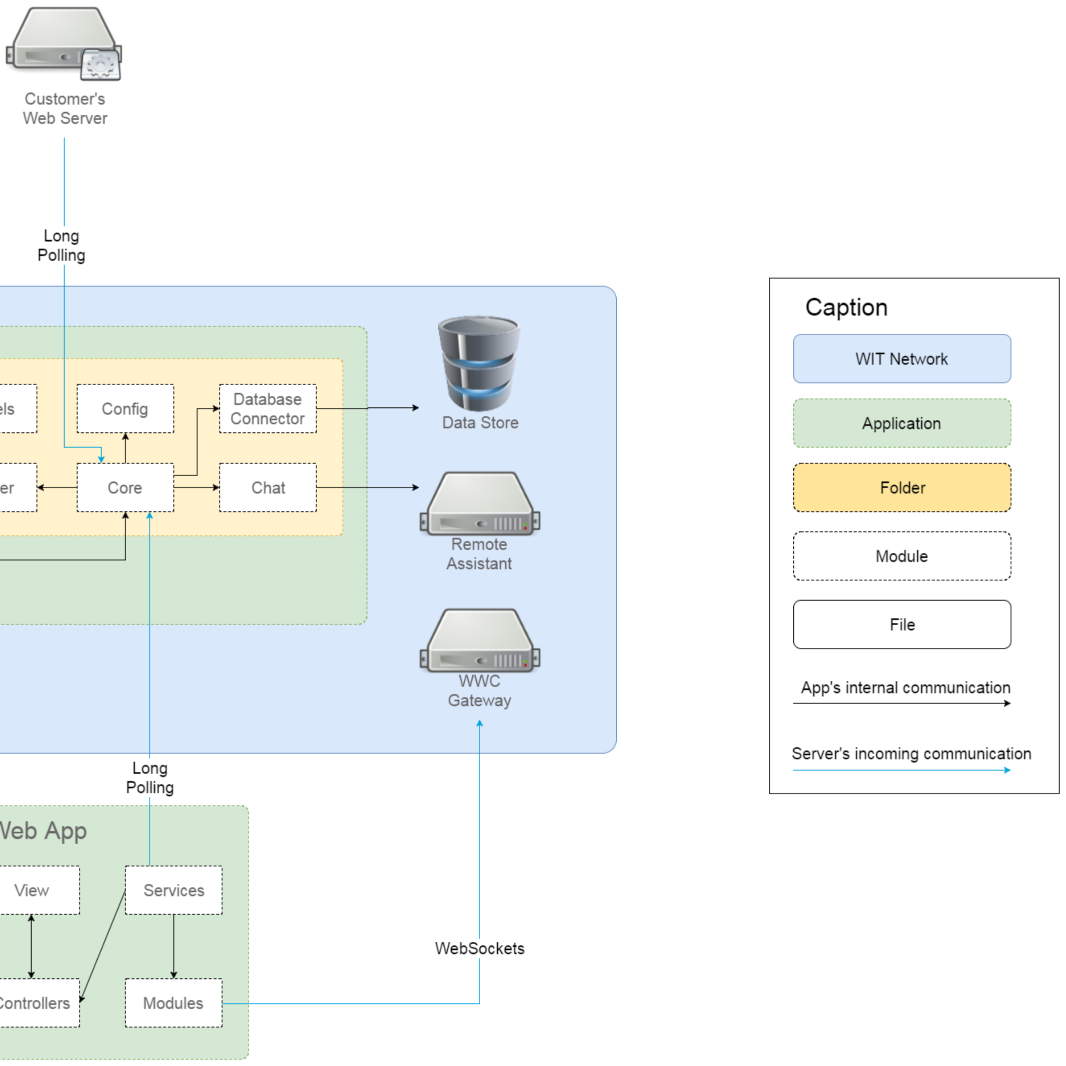


Figure 4.3. Components

As presented in Figure 4.3 there are 3 major containers being developed.

First there is a widget that will be injected on customers' web pages. Widget has 4 big components: Starter component, chat component, Context Collector UI constructor. Starter component is responsible for loading dependencies such as HTML, CSS and JS modules from server, Socket.io module and jQuery, Chat component is responsible for maintain communication with server, Context Collector as it states collects navigation history from the customer and UI constructor holds functions to construct UI as it is needed.

The biggest developed container is the OCVC server. OCVC server has 3 main folders: web, bin and lib. Web folder holds web components both for widget and BO applications, while bin folder holds installation scripts. Lib is the main component of the server. It has 6 modules, each with its own responsibility. Core module is started by index.js at root and it is responsible for starting the remaining modules. It also accepts HTTPS request requesting HTML, CSS and JS files. Chat module is responsible for serving widget and BO information request. Database connector is an abstraction layer between OCVC server and database with the purpose to manage communication between them. Logger is responsible to log all important events. Config holds configuration files needed to run the application. Finally, Models module hold the object models used in the application.

Last container is the BO web app. This container has four components: view holding the HTML files and CSS to create the UI, controller who are responsible to manage UI aspect, modules used through the application and Services responsible for data and communication management.

4.1.4 Classes

For most of us in an Oriented-Object (OO) world, classes are the smallest building blocks of our software systems.

This is an optional level with two purposes:

- Detail big components.
- Describe any particular pattern.

Since JS is not an OO language UML will not be used. However there is a need to draw this level in order to describe in more detail OCVC server (big component with a component-based development [33]) and BO web app (particular pattern).

OCVC Server

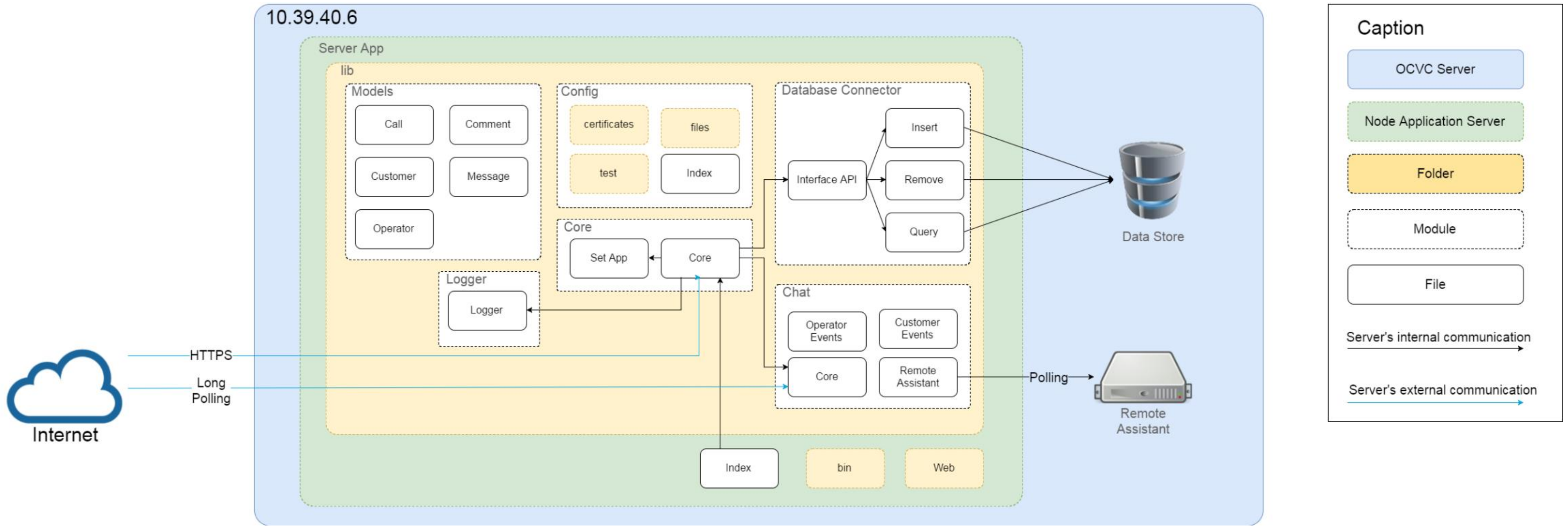


Figure 4.4. Server classes

OCVC Server's Index at root is the file that starts the server, however all logic is under lib folder.

As stated before, lib has six modules: Core, Chat, Database connector, Config, Logger and Models. Server's Component-Based Development (CBD) pattern help to keep responsibilities strict and changes on code easy to maintain, this means that the components are loosely coupled. This is particularly helpful for module maintenance as long as the communication standards do not change. The introducing of new features inside each module or new modules with new responsibilities is also a very easy operation. This architecture pattern was chosen because it kept the system easy to maintain, easy to improve and due to the communication type with customers (Event-Driven [34]).

Core module is the starter. Its main file starts all the remaining modules with the configurations presented at Config folder. It also processes HTTPS requests to serve HTML, CSS and JS files both to widget and BO.

Chat module is one of the most important server modules. It receives the long polling requests and serve the answer to both widget and BO. This module has a main file, Index is where Socket.IO requests arrive and responses leave and where all the functionalities are registered. Other three files have the implementation. So, a Socket.IO request arrive and is redirected to the file holding the implementation needed. After a response is produced it is then sent in Index.

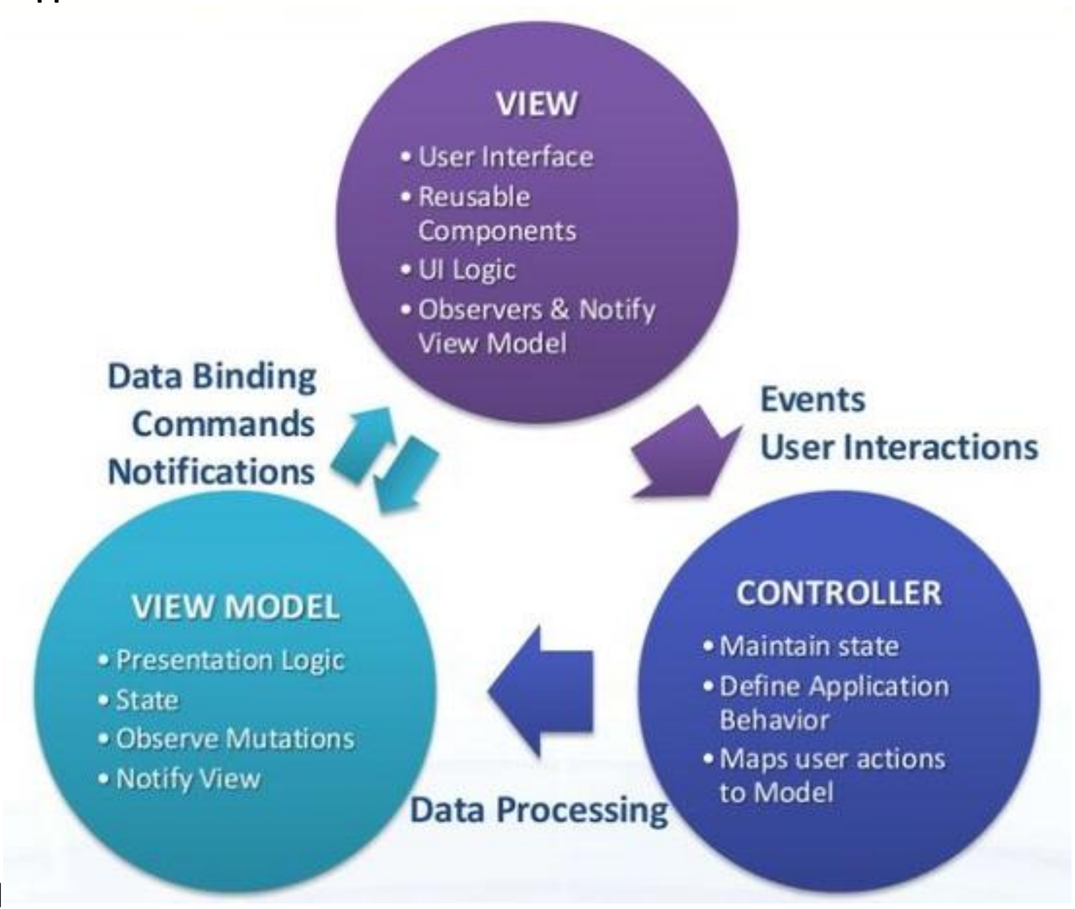
Database connector classes can also be divided into two categories: Interface and queries. Interface class is the Interface API, this class is an abstraction class called by the other components in order to store, read, update or delete elements from the Data Base. The remaining files have the queries implementation calls.

Config file holds configuration files, certificates and tests.

Logger is the module responsible for log the important events that occur on server.

Models module holds the JSON structure for the objects used on server for the remaining modules.

Web App



[35]

Figure 4.5. Angular MVVM model

The frontend application uses AngularJS framework Model-View-View-Model (MVVM).

Model (M) component acts as data access layer. Services presented in model are singletons always in execution.

The view (V) is the application's UI and includes several partial HTML and CSS files that compose the frontend.

The controller (VM) is an abstraction of the view exposing properties and commands. The controllers will register the callbacks that they expect at the service.

View and View-Model have a bidirectional data binder, that allow changes in the model to propagate to the matching views, and changes made in the views are also propagated to the model. When the application data changes, so does the UI, and vice-versa.

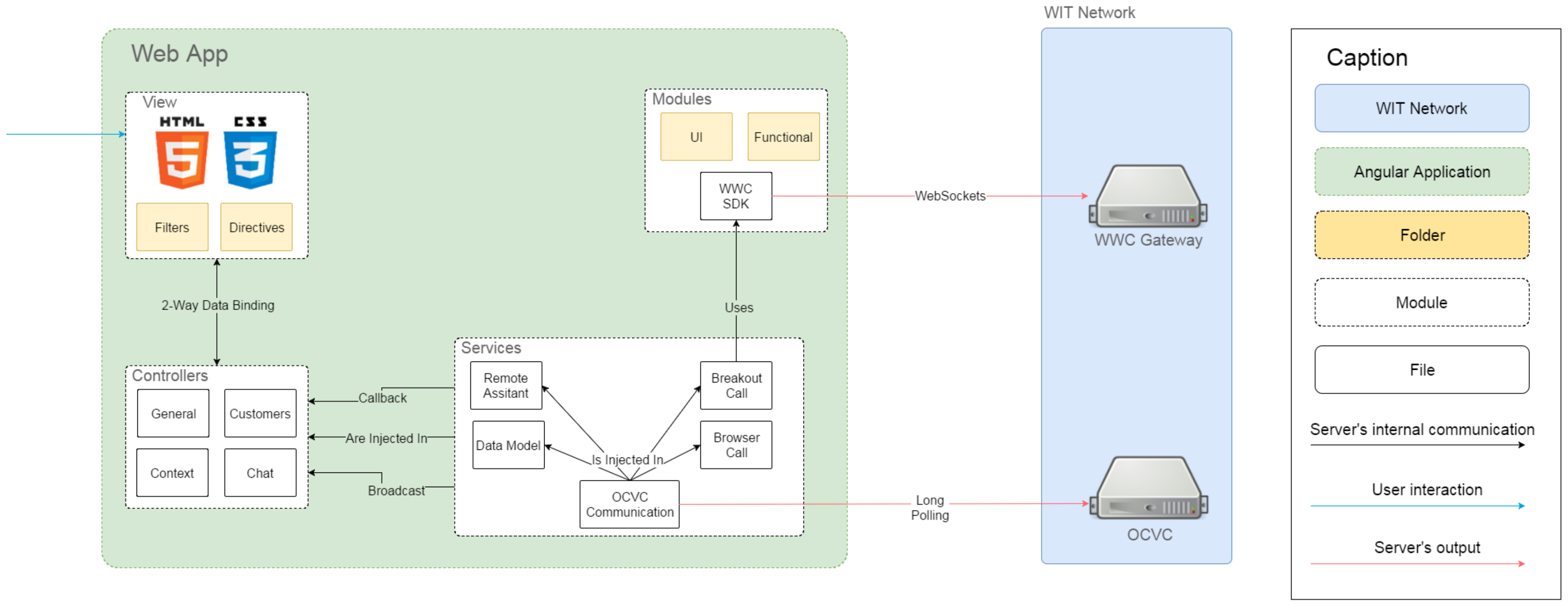


Figure 4.6. Web App classes

As described above, BO web app follow a MVVM pattern. This pattern is ensured using Angular JS framework. Other frameworks such as scroll glue were used in order to help UI management.

The web app is divided is four components: View, Controllers, Services and Modules.

Modules hold all the frameworks needed in the app.

Users interact with the view, composed by several partial HTML and CSS files using Bootstrap framework.

Controllers manage UI behavior, map users' actions and maintain states. Controller were divided in four groups of controller. General controllers manage global aspects of the BO such as window resizing, login and logout. Customers manage all customers' tabs and are responsible for managing features such as select a customer, answering or closing a conversation. Context manage the collected context and comments tabs and are responsible for editing customers' information, adding new comments or open URLs. Chat manage all chat's UI such as sending new messages or making calls.

Services are singletons injected in other components of the web app. After a service is injected in a component it can use service's exposed functions. Services were divided in OCVC Communication, Data Model, Remote Assistant, Browser Call and Breakout Call. If any component needs to communicate with OCVC server it injects OCVC Communication service and the communication is done by it, to receive information callbacks are registered in the same service that will notify the components. Data model holds the needed data received from server. The remaining services are used for call management. Browser call manage call between BO and widget by WebRTC. Breakout calls manage call between BO and a device. In this case WWC Software Development Kit (SDK) is used, the SDK allow communication with WWC Gateway. In order to do so, SDK gives a Communication Interface to call server methods and events registration methods to receive answers from server. Remote Assistant manage calls between BO and iPad's Remote Assistant app and is an encapsulation of the lib developed implementing an Interface that allow the communication between BO and Remote Assistant Server using OCVC server as proxy.

4.2 Data model

Data in MongoDB, and in NoSQL databases in general, has a flexible schema. Unlike SQL databases where a declared tables' schema is defined before inserting data, in MongoDB no schema definition is needed. Instead of tables are used collections of documents where each document as the flexibility of mapping different data between themselves even if the data fields have substantial variations. In practice, however, the documents inside a collection share a similar structure.

This section presents the collections used in the internship, and the similar structure definition of their objects. Note that the presented representation of the documents is a general one. Each document inside the collections could have more or less fields than the presented. This is not a schema definition, instead is just a similar structure definition of the documents.

In MongoDB are used two tools to represent relationships between data:

- References: when documents store an ID from one document to another.
- Embedded documents: when a document structure is stored inside a field or array.

References are used when the same document must be stored several times inside a document or a collection. Instead of store the whole document, an ID to it is stored and data redundancy is kept at minimum. Embedded documents are used when the document to save does not justify the creation of a new document (e.g.: date/geographical structures).

Above are the collections created and their reference documents. A description of the collection and document is written, as well as all decisions are justified.

4.2.1 Customers' Collection

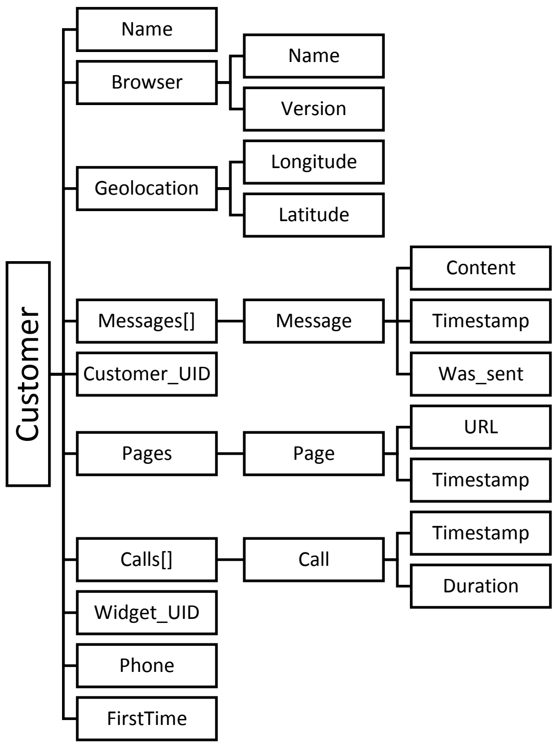


Figure 4.7. Customer document reference representation

Summary

Customers' collection hold the documents that reference each customer. Each customer will be stored in a document.

Figure 4.7 represents the reference document for the customers. Some of the fields will always exist every time a new document is created at the collection. This fields are Customer_UID and Pages. The other fields can or cannot be created depending on customer's actions during his visit. For instance, if a customer never chat with an operator the Messages[] is not needed.

For the geolocation field it was opted to use embedded documents instead of references, since every document can have very different geolocation values it was unnecessary to create a collection to hold every geolocation. The same goes to the Messages and Calls fields.

In the cases of the Widget_ID it was used references. It would make no sense to use embedded objects and increment the data redundancy on the database. The use of the references allow the documents to share the same Widget document without the need to store it in every Customer document.

4.2.2 Operators Collection

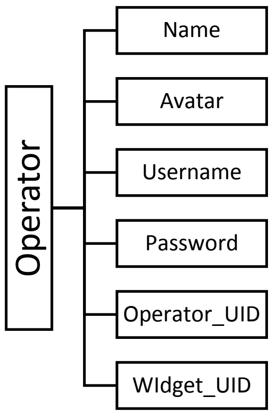


Figure 4.8. Operator document reference representation

Summary

This collection stores the operators' information. Each operator will be stored in one document. Figure 4.8 represents the reference document to the collection.

Since no composed fields were needed it was not used embedded objects. However, it was needed a reference to the widget the operator is associated with. The other fields are only used to store operator's profile information.

4.2.3 Online List Collection

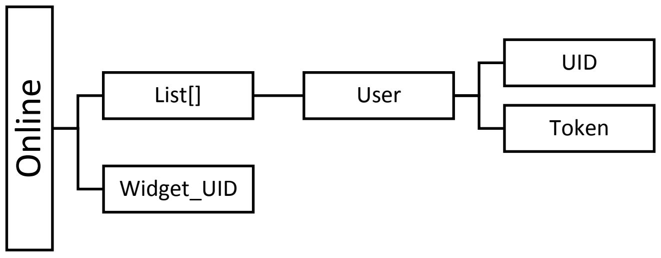


Figure 4.9. Online list document reference representation

Summary

Collection of documents that store the list of online users per widget. In the internship, this collection will only hold one object since the use case of serve multiple websites is out of scope. However, the document reference is already prepared to store multiple widgets.

In this case it was opted to mix references and embedded documents.

To store the online users it was used embedded objects with a reference to the user. To store the widget it was used a reference.

4.2.4 Widget Collection

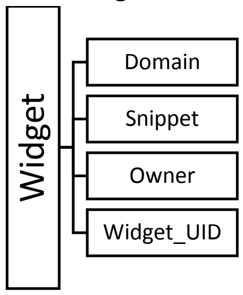


Figure 4.10. Widget document reference representation

Summary

Like the previous case, in spite of being a collection of documents that store the widget documents, this collection will only hold one document.

This collection stores widget's information and widget's owner information.

5 Development

This chapter refers to the development done over the project.

There are always challenges when developing software. This chapter will explain the implementation process, the evaluation of the solution and at the end of the chapter is presented in the future work.

5.1 Developed Work

A good quality planning and architecture design help to predict and overcome most of the challenges that will appear during the development. However, sometimes if the planned process is not the clear unexpected problems can appear.

During the internship planning most of the problems were predicted and contingency plans were created in order to minimize its risks, still, some were impossible to predict and there was a need to create strategies to resolve them.

This section highlights the major features implemented. Here are the functionalities described and the sequence diagrams. To see all the features in more detail please refer to **Annex D, chapter 2 – Developed Work**.

5.1.1 UI specification and implementation

For the UI definition the intern worked with WIT designer Elizabeth Pereira.

The first step was the definition of the requirements and user stories. With that document ready, the designer drew the mockups, which were evaluated by all the team members (scrum master, product owner and scrum team) as well as the stakeholder.

The process of defining an appealing UI that would give to the final user a great experience was progressive and done in several iterations. Elizabeth was in charge of drawing the mockups taking into account the documents of the internship.

Once a satisfactory result was achieved from all involved in the project, which is presented at Figure 5.1 and Figure 5.2, the UI implementation began.

The UI was implemented using Bootstrap 3.3.5 framework that helped to make the back office responsive. For the back office AngularJS framework was used, in order to implement an MVVM model.

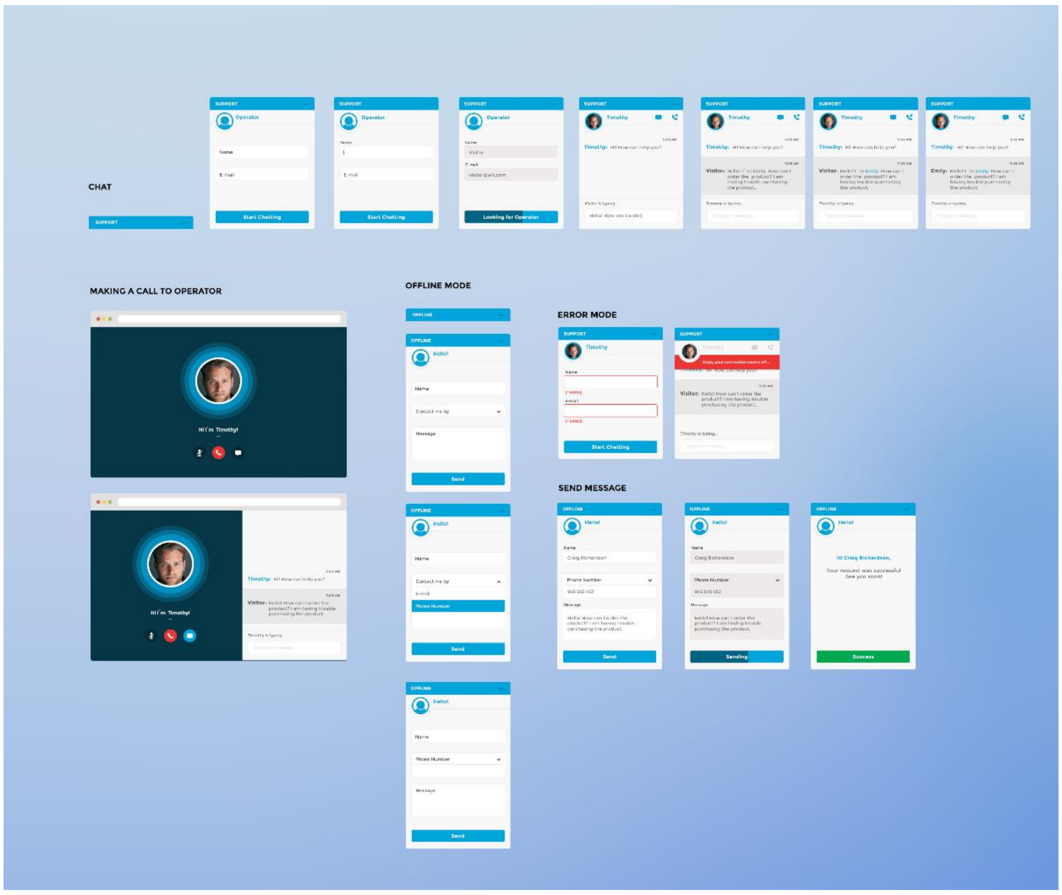
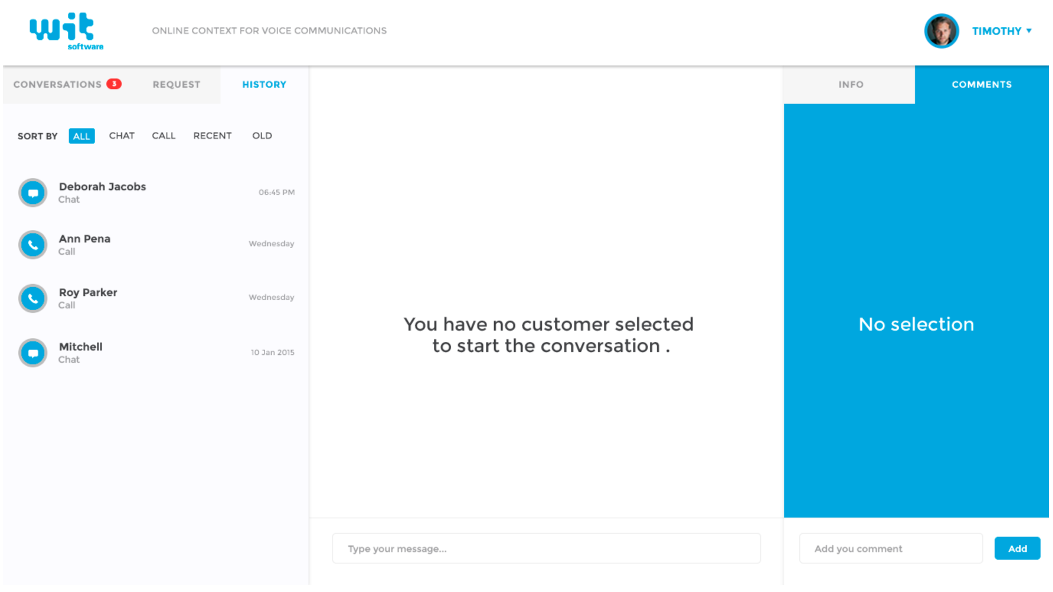


Figure 5.1. First design for widget's UI



ONLINE CONTEXT FOR VOICE COMMUNICATIONS

TIMOTHY ▾

CONVERSATIONS 3 REQUEST HISTORY

SORT BY ALL CHAT CALL RECENT OLD

Deborah Jacobs
Chat
06:45 PM

Ann Pena
Call
Wednesday

Roy Parker
Call
Wednesday

Mitchell
Chat
10 Jan 2015

You have no customer selected to start the conversation .

INFO COMMENTS

No selection

Type your message...

Add you comment

Add

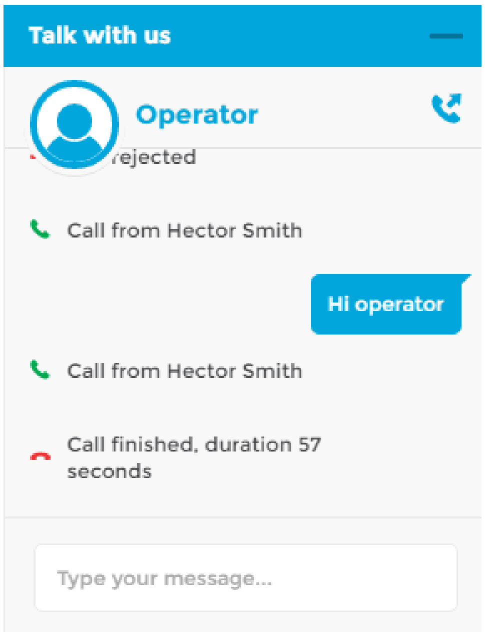
Figure 5.2. First design for back office's UI

The UI implementation was one of the biggest challenges in the project.

It was very hard to find a UI that was appealing to everyone involved in the project, and that suited both operators and customers' purposes in terms of user experience. Another major factor that influenced the UI implementation was the find of new features to implement. These requirement changes were taken into account during project's planning; therefore development was not affected with these changes. However, this led to a major change of the UI.

From the first UI mockups to the last, there were several changes. One of the main changes was the widget's communication flow – in order to attract more customers the identification form was removed and costumers can start a conversation without identify themselves. Messages displayed suffered several changes on development both on BO and the widget – messages are now grouped if they were received at the exact same minute. In the BO some paddings were adjusted between elements to refine the UI. Beside that the contextual information, the column changed a bit along with the communication buttons used to start and answer calls.

The aspect of the UI is presented on Figure 5.3 to Figure 5.7.


Talk with us

Operator

rejected

Call from Hector Smith

Hi operator

Call from Hector Smith

Call finished, duration 57 seconds

Type your message...

Figure 5.3. Final Widget UI

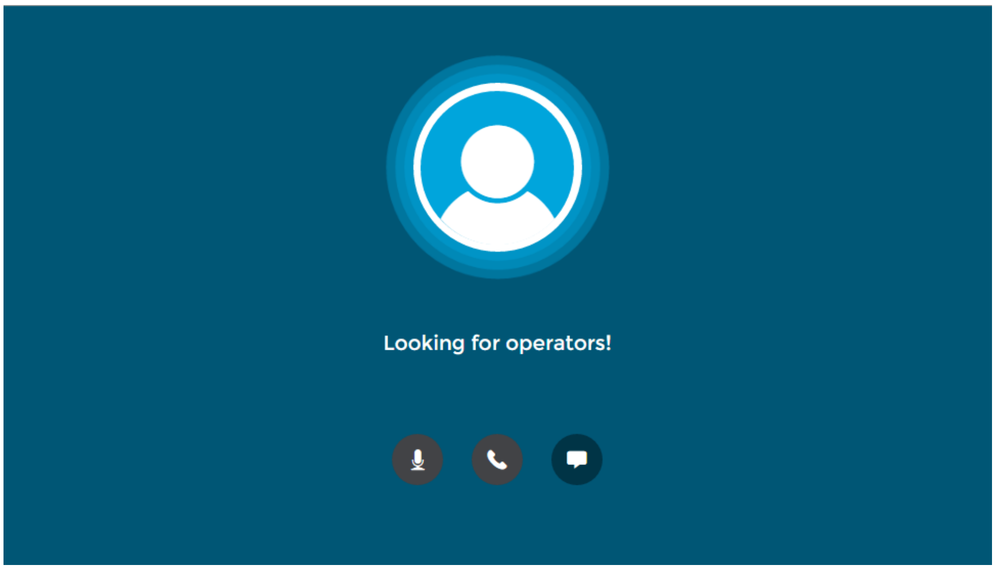
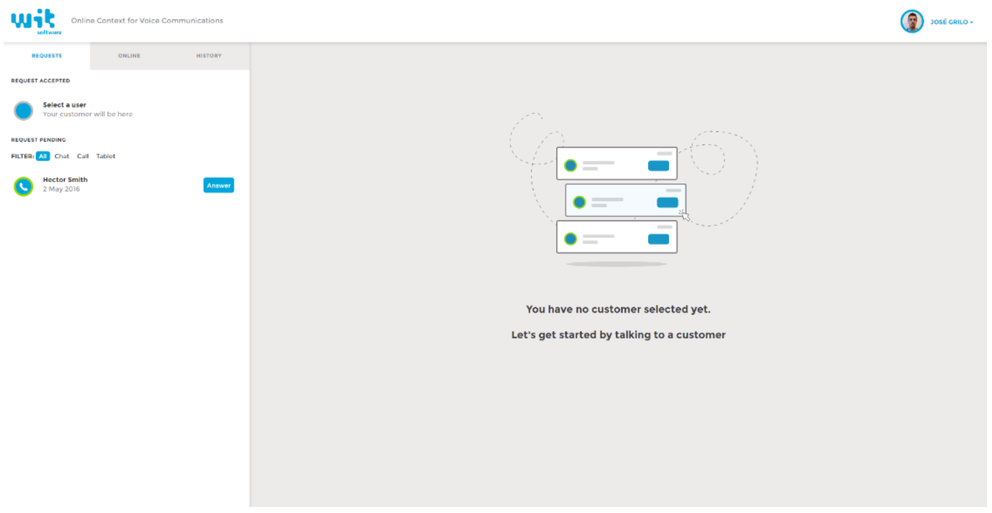

Looking for operators!

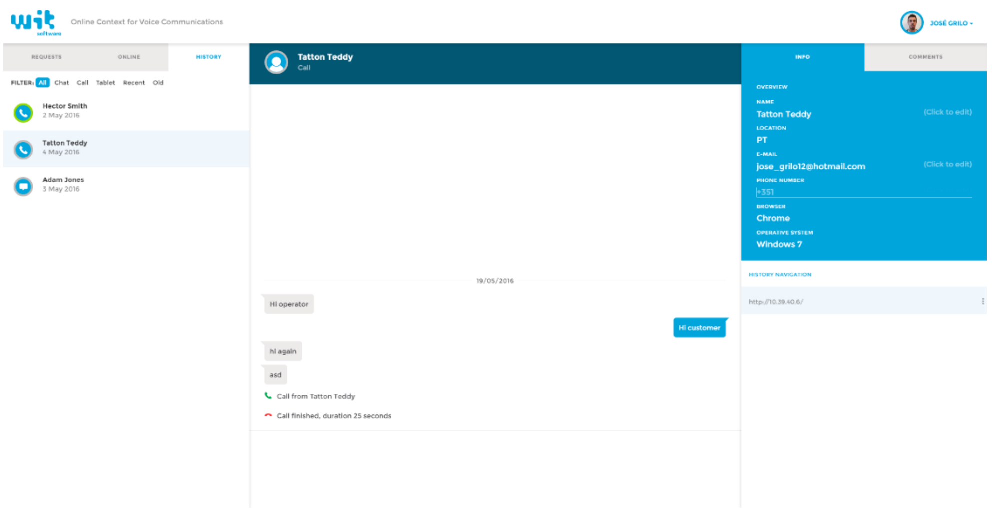
Figure 5.4. Customer's call window UI



You have no customer selected yet.

Let's get started by talking to a customer

Figure 5.5. Final back office UI - Entrance screen



NAME
Tatton Teddy (Click to edit)
LOCATION
PT
E-MAIL
jose_grilo12@hotmail.com (Click to edit)
PHONE NUMBER
+351
BROWSER
Chrome
OPERATIVE SYSTEM
Windows 7

HISTORY NAVIGATION
http://10.39.40.6/

19/05/2016

Hi operator

Hi customer

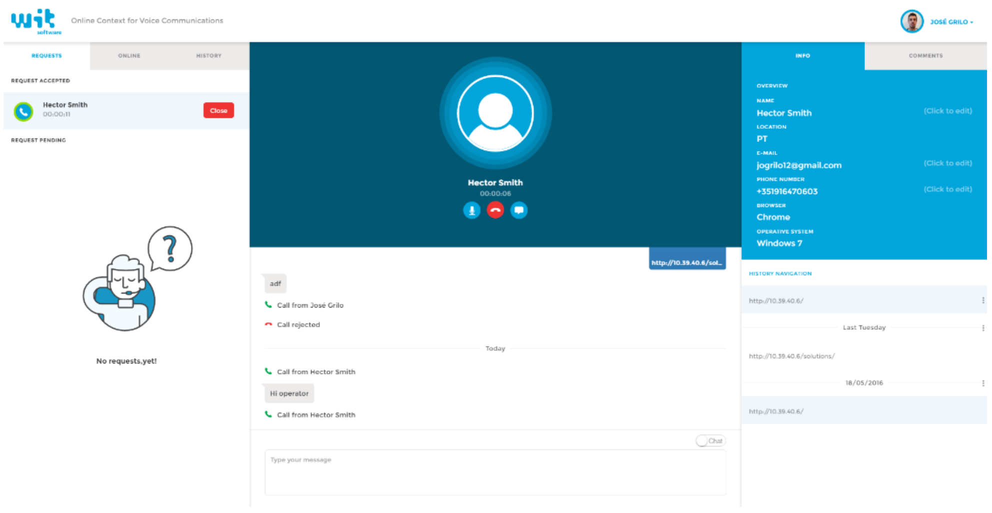
hi again

asd

Call from Tatton Teddy

Call finished, duration 25 seconds

Figure 5.6. Final back office UI - Customer selected



NAME
Hector Smith (Click to edit)
LOCATION
PT
E-MAIL
jogrilo12@gmail.com (Click to edit)
PHONE NUMBER
+351916470603 (Click to edit)
BROWSER
Chrome
OPERATIVE SYSTEM
Windows 7

HISTORY NAVIGATION
http://10.39.40.6/
Last Tuesday
http://10.39.40.6/solutions/
18/05/2016
http://10.39.40.6/

http://10.39.40.6/sol...

adf

Call from Jose Grilo

Call rejected

Today

Call from Hector Smith

Hi operator

Call from Hector Smith

Figure 5.7. Final back office UI - On call

5.1.2 Operator's Login and Socket Disconnection

Not everyone can use the back office to answer customers' requests. Since that was a major requirement an authentication window was created. This window pops-up before the user enters the back office administration window. Here the user fills in his login information and it is sent to the backend using HTTPS. Once the credentials have been authenticated, the back office will be displayed and a connection to the backend is created.

It's as equally important to have a logout action, as it is to have a login action in place, which allows the operation to close the working session. In this case the connection is broken and the operator automatically returns to the login page.

5.1.3 Customer's socket connection and disconnection

These were one of the most challenging features to implement due to its complexity.

To encourage customers to start a conversation it was required that a conversation could start without a previous login. It was also required to identify the customer that was sending the requests, in order to keep that track a Universal Unique Identifier (UUID) was given so that it appended to every request and was used to unequivocally identify a customer, since that UUID was passed as an HTTPOnly cookie that can only be altered on server.

The implementation challenge came from the requirement that required that a customer would be able to navigate the site on multiple tabs on the same browser. This brought problems, especially to differentiate when a customer left the site from when he refreshed or navigated through the browser. To overcome this challenge, the solution found was to create a timeout every time the last socket from a client broke, and clean the timeout if the customer reconnects to the site.

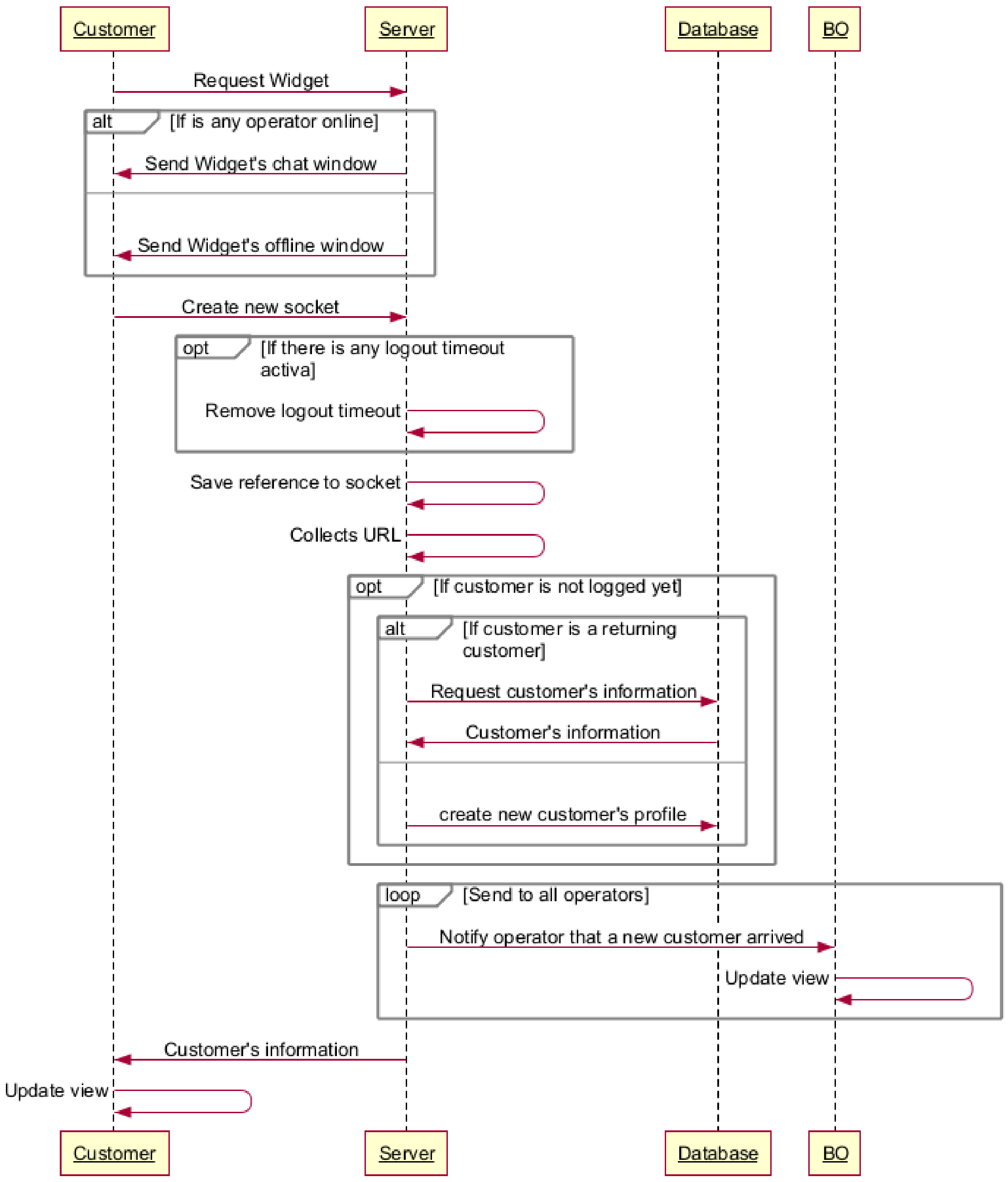

Customer
Server
Database
BO
Request Widget
alt
[If is any operator online]
Send Widget's chat window
Send Widget's offline window
Create new socket
opt
[If there is any logout timeout activa]
Remove logout timeout
Save reference to socket
Collects URL
opt
[If customer is not logged yet]
alt
[If customer is a returning customer]
Request customer's information
Customer's information
create new customer's profile
loop
[Send to all operators]
Notify operator that a new customer arrived
Update view
Customer's information
Update view
Customer
Server
Database
BO

Figure 5.8. Customer's socket connection sequence diagram

Unlike the typical chat system, in the product developed the customers don't need to input any personal information to start a chat. This decision was made to reduce the probability of a customer leaving the site because he was forced to share his name or email.

To use the product the customer only needs to access the site and the widget assists automatically. If operators are not available an offline window is displayed, otherwise an online window is shown. After, the socket is created and personal information is sent over. The socket is then used to send and receive chat messages, call events, and contextual information is collected.

Finally, the operators are notified that a costumer entered the site.

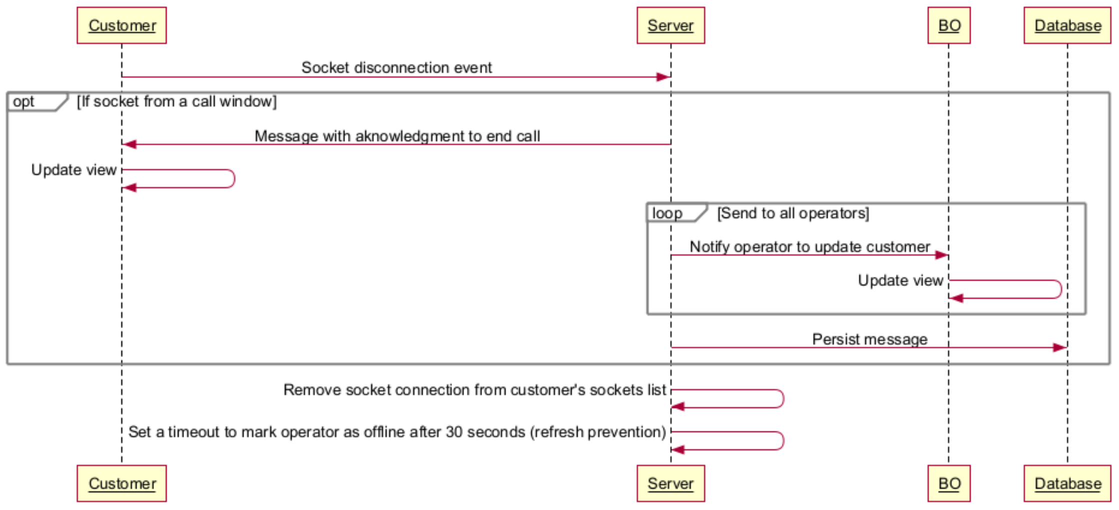

Customer
Server
BO
Database
Socket disconnection event
opt
[If socket from a call window]
Message with aknowledgment to end call
Update view
loop
[Send to all operators]
Notify operator to update customer
Update view
Persist message
Remove socket connection from customer's sockets list
Set a timeout to mark operator as offline after 30 seconds (refresh prevention)
Customer
Server
BO
Database

Figure 5.9. Customer's socket disconnection sequence diagram

The socket disconnection event is very similar to the operator's socket disconnection event.

The socket disconnection event has two meanings: the customer is refreshing the site or the customer has left the site. When this event happens it is verified for ongoing calls with the customer and they are finished and the operator is notified. Since the event is the same for both page refresh and leaving the page a timeout is set. If the customer creates a new socket before the timeout fires the same is dismissed. If the timeout is triggered then the socket is removed from the customer's sockets list and if the list turns empty then the customer is marked as offline and the operators are notified.

5.1.4 Context Collection

The context collection feature refers to all the information that can be collected that can help operators understand the reasons behind the navigation and questions from the customer and provide a better service.

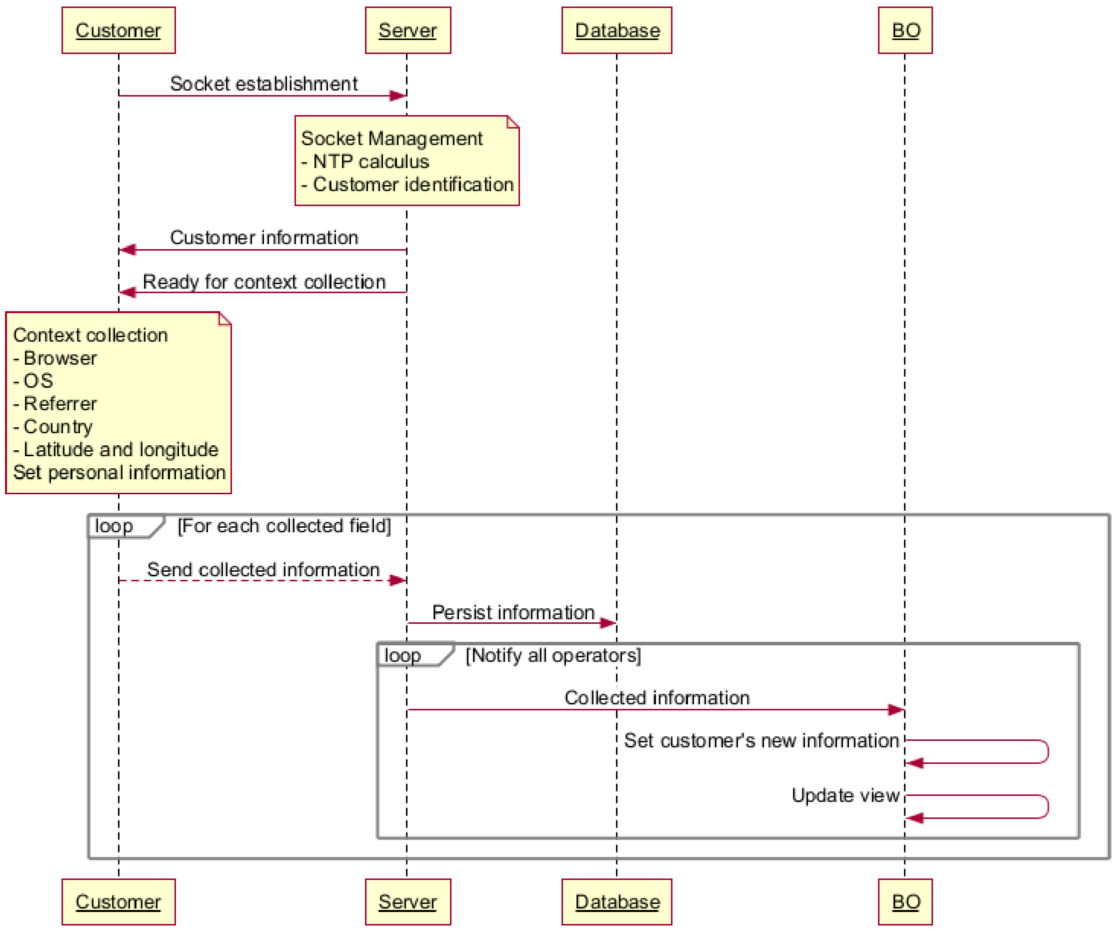


Figure 5.10. Context collection sequence diagram

To collect both OS and browser's information, the *window.navigator.userAgent* property from the browser was used. Every time that the widget starts, it analyses and parses the string returned by the property, and returns the OS and browser's information, which are sent to the server via Socket.io.

To collect the country code it is called http://ipinfo.io [35] API that returns a JSON with several fields, among them is the country code.

The latitude and longitude values are obtained with the Geolocation API from the HTML5. This information is then translated on server side, using Google's Geocode API location.

For the visited URLs, every time a new Socket.io connection is created, its header is parsed. If the origin matches the server URL, the URL is ignored because it means that the connection is from an operator. Otherwise, the referrer field is parsed to get the URL where the customer sent his request.

After it has been parsed and validated on server side, the collected information is then sent to the operators in real time in order to allow for easy assistance of the customers.

5.1.5 Start and End a conversation

There are two ways to start a conversation. The most common is when a customer makes a request to be contacted, either by chat, mail or phone and an operator opts for answer and contact him.

The second way is when an operator chooses a customer from the online list and starts to talk to him without any previous request. This proactive contact can be used to get more insight on the customer's visit, to show the customer any special promotion or to force sales for instance.

Only operators can state the conversation as "closed". When a conversation is closed, it is passed onto the history. Besides that, the customer is removed from the pending queue and is notified. When a conversation is closed the customer has the option to get the conversation over email if he wants to.

5.1.6 Chat

After a conversation starts any of the actors can contact the other with instant messaging.

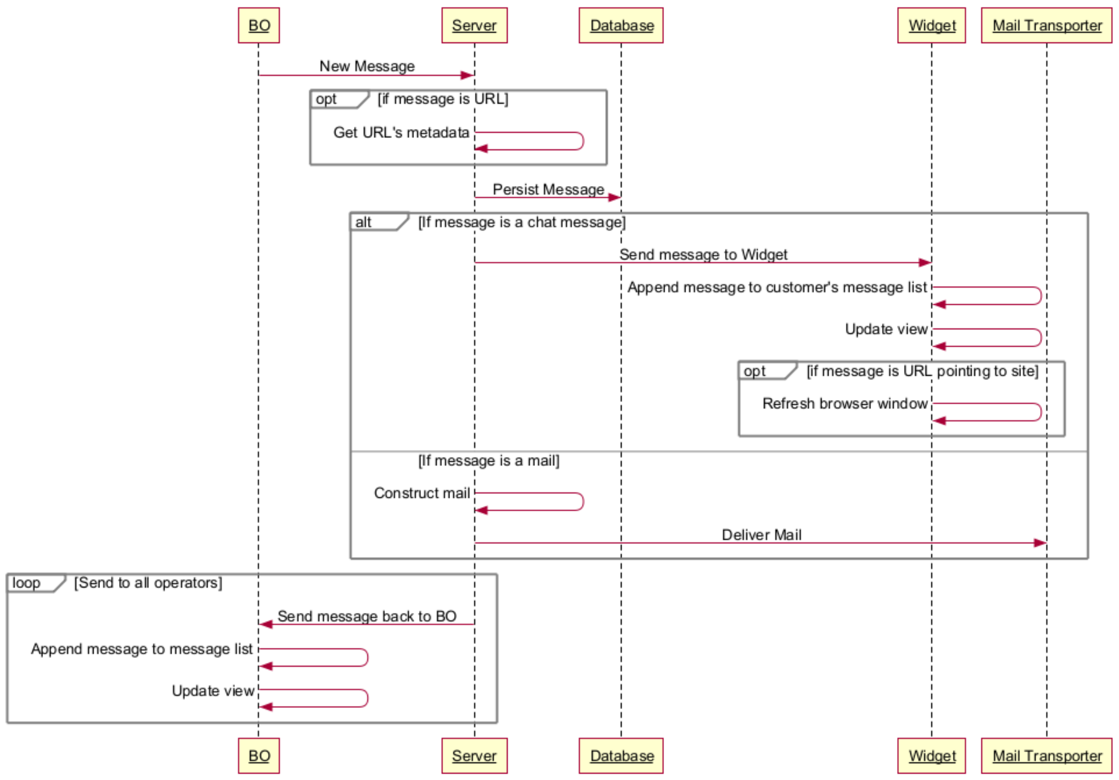


Figure 5.11. Operator's message sequence diagram

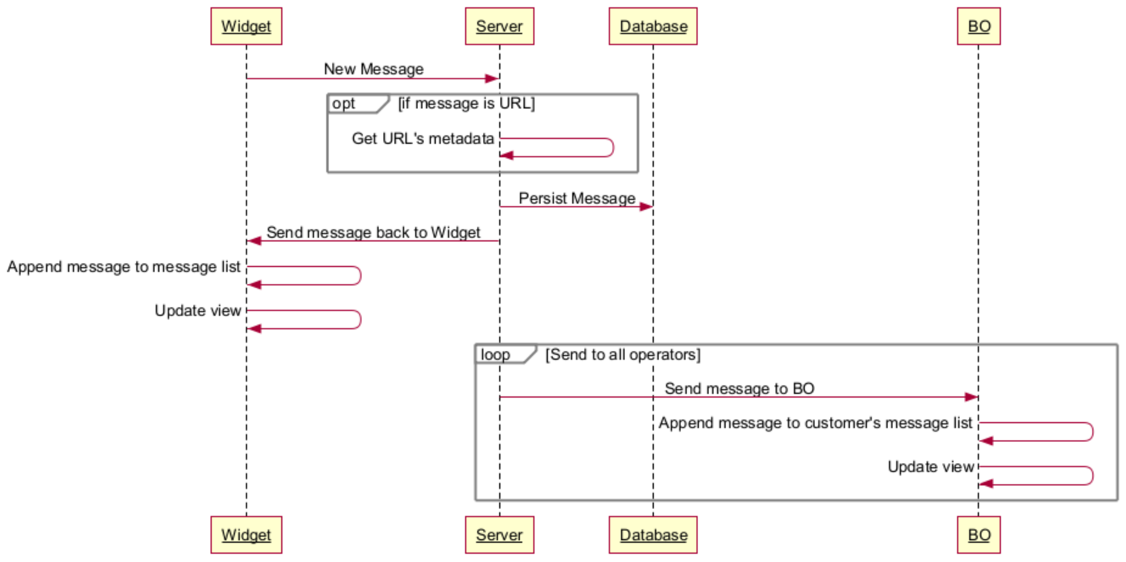


Figure 5.12. Customer's message sequence diagram

The chat flow is very similar for both actors. The message is redirected by the server to the other actor. All messages are sent to all operators. This allows all operators to follow the conversations in real time even if they are not answering the customer.

The major differences between customer and operator messaging is the way that messages are processed when received by the widget. A message from an operator received by the widget it is processed looking for a URL. If it contains an URL and points to the site domain, the customer is redirected to that location.

5.1.7 Calls

Another way to contact the customer is through a call, either by browser or device.

Browser calls were fully developed by the intern and breakout calls were developed using WWC gateway.

In the first case, the customer creates a call request that is sent to the operator through the server. Once the operator accepts, the customer makes the call request, this is sent to the operator through the server. When the server accepts the request, the client creates an offer which is then sent to the operator which in turn responds with an answer and the connection is created. On the other hand, the operator starts a breakout call. The flow is the same as the browser call, however the WWC gateway makes the data transformation in order to pass the requests to their receivers.

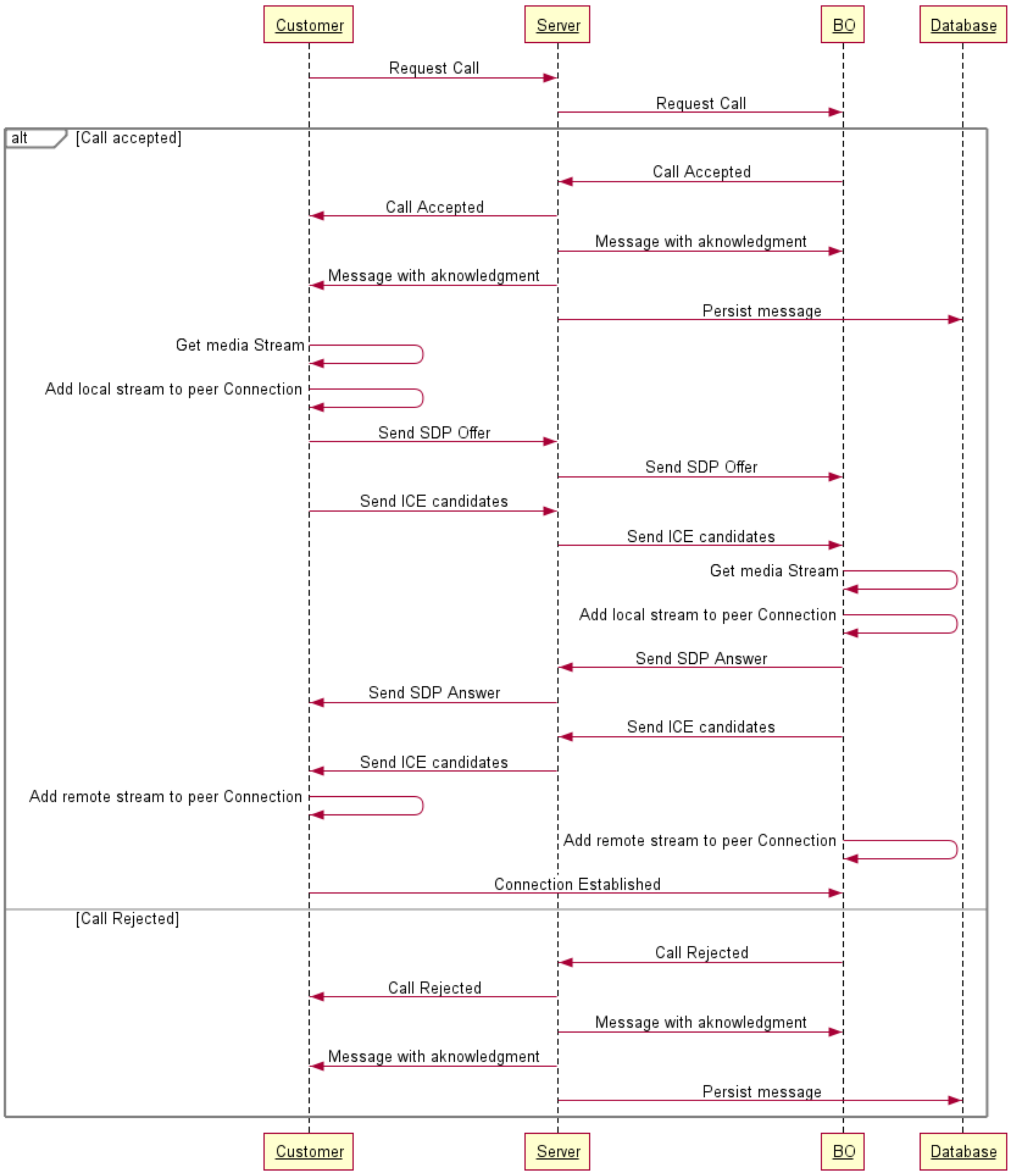


Figure 5.13. Browser call sequence diagram

The sequence presented above describes the browser call sequence. The call is always asked by the customer and the operators needs to accept it, in order to start the call. After all the signaling the actors create a peer connection, gets access to media and an exchange session descriptions – both the customer and operator generate a session description – and Interactive Connectivity Establishment (ICE) candidates. This exchange is asynchronous.

Once both actors have each other's stream a new peer to peer connection is established and the call starts.

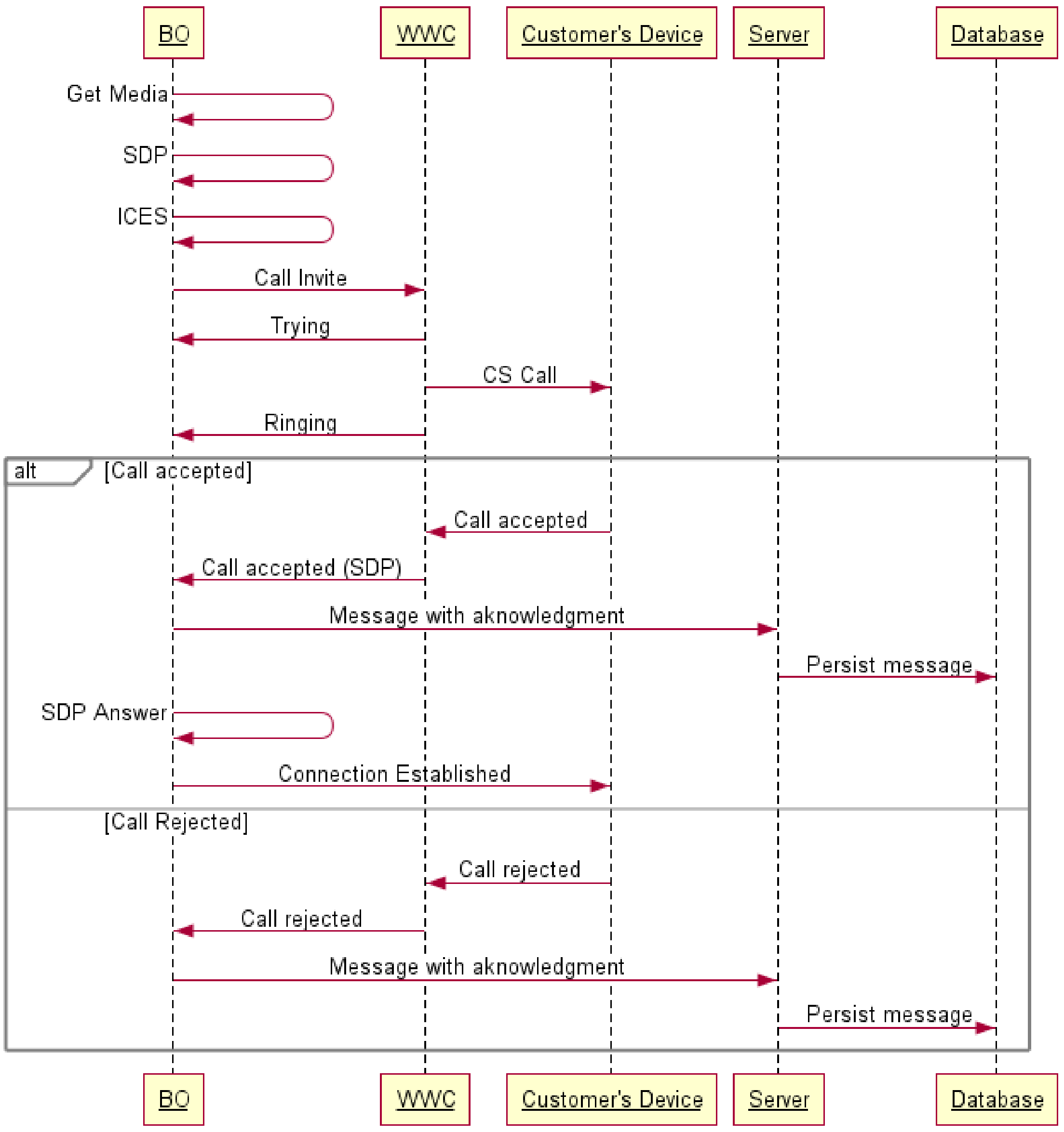


Figure 5.14. Breakout call sequence diagram

Breakout calls and browser calls have a very similar flow, however WWC gateway serves as the middle man to manage and translate packages from the browser to the device.

WWC SDK was used to communicate with WWC gateway. This SDK offers a communication API to make requests and handle responses. After login, WWC SDK registers at WWC using a previously configured account and an API key. After that, the SDK is ready to make requests for the calls. These requests act as a browser call request – SDP and ICE candidates are exchanged, and WWC gateway translates the requests to be interpreted by both the device and browser. Answers are served as events and WWC SDK offer callback functions to handle those events.

5.1.8 Remote Assistant

Calls received from the remote assistant app were a different issue. Since this demo was already part of WIT's portfolio it was not an implementation challenge, but rather an integration challenge. The architecture was already flexible enough to integrate with remote assistant server, however some changes needed to be made. Since the communication between customer and server was done by HTTP, the OCVC server was used as proxy so a new service was implemented inside the Chat module. The remote assistant back office (Figure 5.15) was also redone to meet Angular and UI standards (Figure 5.16).

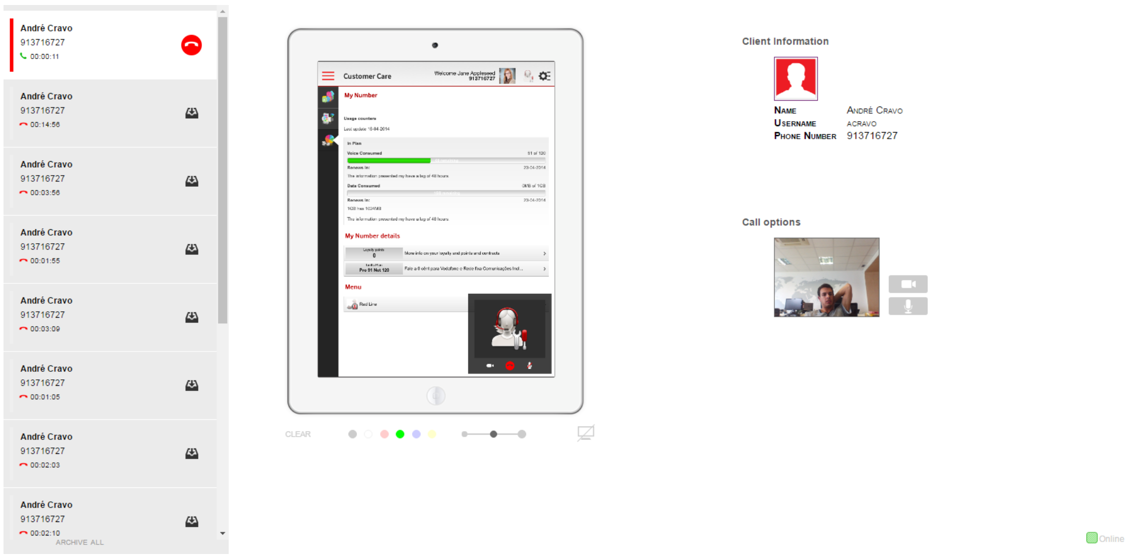


Figure 5.15. Remote assistant's back office old UI

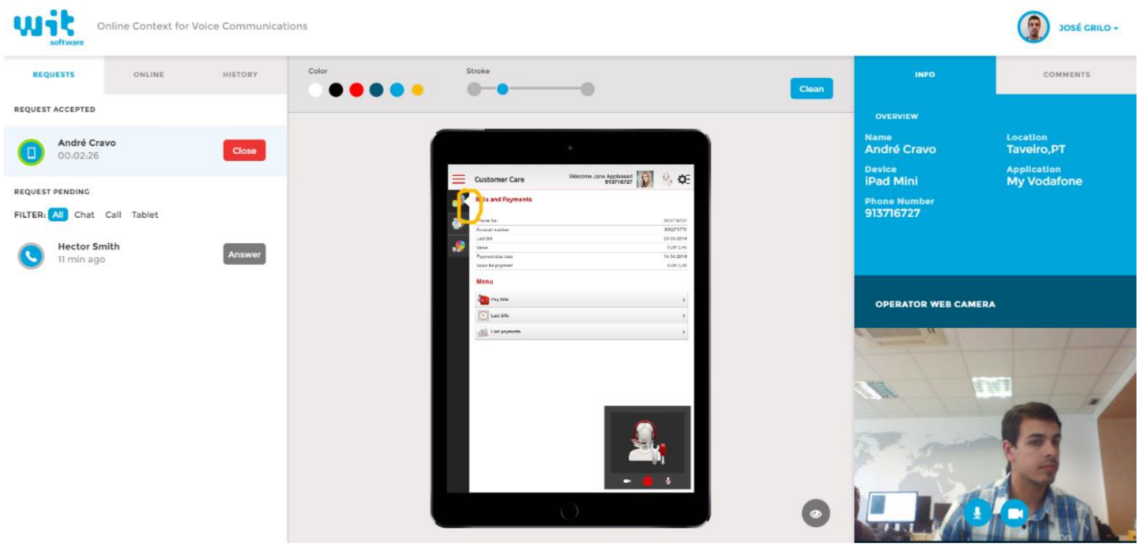


Figure 5.16. Remote assistant's back office new UI

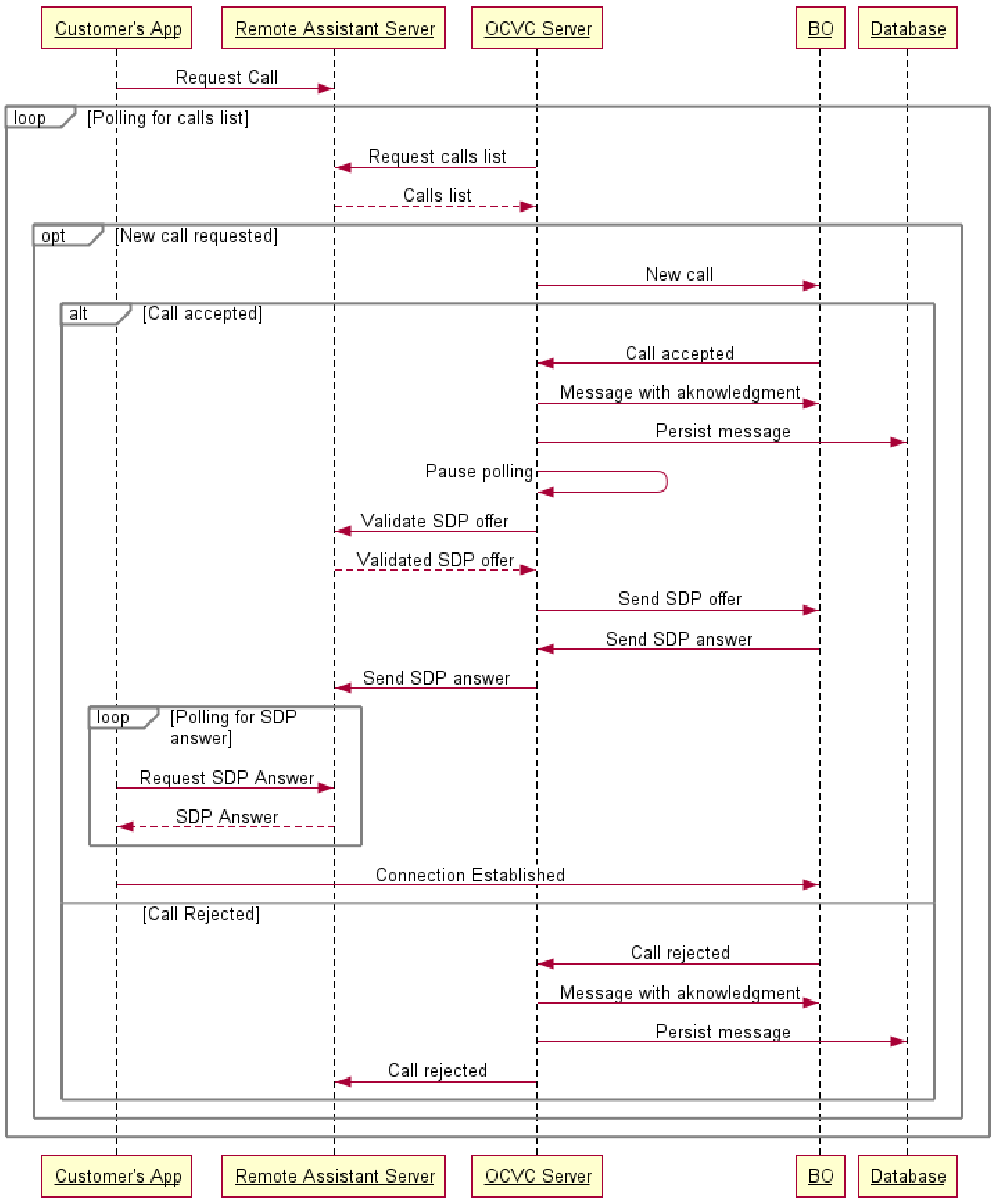


Figure 5.17. App call sequence diagram

As in breakout calls, there is a server in the middle between browser and device, but in this case server implements a REST API which is called every second (polling) to check for call updates.

All communication between actors was processed by OCVC server, which served as proxy due to Remote Assistant HTTP implementation. Requests are posted by the device on Remote Assistant server. OCVC is polling information every second until a change occurs. At this moment the offer is read and an answer is produced and posted. As previously stated, the device is also polling for changes and when it find an answer to its offer a connection is created.

The challenge here was in terms of integration. Besides knowing that there are better communication forms instead of polling (ex.: websockets or long polling) it was opted to not change the Remote Assistant server due to the complexity and time constraints.

The biggest challenge was to handle the communication between BO and the device. On the existing remote assistant demo the communication was by HTTP. On OCVC implementation, BO is served by HTTPS, so HTTP communications are not allowed due to Same Origin Policy. The solution found was to put OCVC server as a proxy, so the Remote Assistant communication lib presented on BO was merged to OCVC server.

5.2 Tests

Testing the product was an important part of the project. Since an agile methodology was used testing was a continuous process which means that at the end of each sprint the implemented features and integration were fully tested by the ones presented at the sprint meeting.

This testing methodology allowed a fast way of tracking and fixing bugs during the development at the same time that a new increment was produced.

In this section, will be presented the set of tests used to tests the features and the results obtained.

5.2.1 Set of Tests

Sets of tests were created to test all features to ensure the project's product quality, reliability and good experience.

Functional tests served to prove the quality and reliability of the application. These allowed to find unexpected behaviors or potential crashes on unexpected situations. The requirements were used to produce the set of tests created. Features were grouped in five sets: authentication, context collection, back office features, widget features and call window features. Since most of the features produce a visual response it is important to guarantee that the developed features produce the expected results.

Functional tests follow WIT's Test Guidelines. For every test produced a unique name was given with a description, conditions to test, requisites and steps needed. Each test was classified in terms of importance in a scale from Low to High – a low important test is a test that verifies a feature that is not very relevant to the product functionality, a high important test is test that verifies a feature that is critical to the product functionality.

The sets of tests produced is as an appendix due to its extensiveness. To see the sets produced please refer to subsection **Annex D – Development, 6.1 – Test Sets**.

In spite of not having a set of tests to test the usability, there was a constant topic discussion over the project. As stated on subsection 5.1.1 the UI suffered great changes over the project result of a continuous improvement to the user experience. Everyone involved in this discussion actively participated.

The main reason why there were no usability tests was due to the fact that this product is still under development. Since it will be used as POC and a demonstration product, if any potential interested appears the UI must be rethought to fulfill the interested needs.

5.2.2 Results

At the end of each sprint the corresponding tests were run in order to validate the feature and the results were used to fix bugs.

After the features were implemented from the backlog, sets of tests were run. Full test result are presented at **Annex D – Development, 3.2 – Results**.

From a total of 82 tests only 2 not passed which led to a 98% of acceptance.

Test Set	# of tests	Passed	Failed	% Acceptance	% Failure
Authentication	4	4	0	100%	0%
Context Collection	7	5	2	71%	29%
BO's Features	41	41	0	100%	0%
Widget's Features	19	19	0	100%	0%
Call Window Features	11	11	0	100%	0%
Total	82	80	2	98%	2%

Table 5.1. Tests results

From the table analysis it is easy to see that the tests that failed both belong to the context collection group. In the beginning those features weren't thought to be implemented, and the UI was not prepared to display those values, however they were tracked and saved because they seemed relevant for future work on the project.

To analyze WebRTC calls it was used WebRTC Internals [36] from Google Chrome.

In addition to Chrome's WebRTC Internals, WWC SDK also offers an event to analyze connectivity status with *window.getStats* API if available. The returned results are:

- Available bandwidth;
- Input level;
- Packets lost;
- RTT;
- Packets sent;
- Bytes sent.

5.3 Future Work

All the proposed scope was implemented and the project was successful.

The developed product will be used for demos and as a proof of concept either for its technologies, experience and product concept, which means that this POC will be used in the future and can be modified or evolve to a product if any potential interest appears.

As stated at subsection 3.2.1 some features were left out of the scope, but those features were not forgotten and can be used to improve the project's concept.

This section presents the reader some features that were analyzed, but were not implemented due to time or complexity reasons.

Multiple operators

Change server in order to work with multiple registered operators. Furthermore, a new page can be created in order to add and remove operators.

Priority queues for customers and operators teams

Different request types can be added to different queues with different priorities. For instance, clients that requested a call could be all on the same queue, and the same operators team would answer them all.

Operators team could be divided by request type (chat, call, tablet), page type (products, sales, ...) and others.

Automatic customers' engagement

Currently, operators without any request can engage customers. This approach could be done automatically using rules to approach the customers.

For instance, if the same user visits the same page a certain number of times an automatic message could be sent in order to approach him. The set of rules could be customizable by operator.

Develop promotion feature

The reserved keyword :promo: is used to open a promotion div on the customer's browser. This feature could be improved by adding a list of promotions on the BO so that the operator could choose one and send it to the customer.

Collect/show more context

The context collection can be expanded. For instance, the first time the customer visits the hosting site or the page referrer are collected but not displayed. Besides that, the last conversation time, last visit, time between visits, visit duration or page visit duration can also be tracked.

CRM integration

If the hosting site holds a CRM, integration could be possible in order to enrich the customers' information provided to the operator.

Multiple active chats

The number of active conversations at the same time could be customizable.

Automatically infer customers' relationships

The BO could help the operator to answer the customers' needs. For instance, if the hosting site is a sell retailer website if the customer search for an amount of the same type of products the operator could suggest some related.

Chat transfer
Chat transfer could be possible in order to transfer the conversation to a more suited operator.

VoIP to all browsers
A huge improvement would be the development of VoIP calls to all browsers instead of the current solution that only support browsers with WebRTC technology.

Security Issues

Security issues such as data privacy and encryption and the use of cookies to track customers should be studied before this POC is used as a product.

UI Redefinition

A new UI is being prepared in order to meet the new features proposed and companies UI standards. This UI is currently being evaluated to decide if it will be rethought or implemented.

Message and comment edition
After sending a message, the same could be passive of edition or deletion.

Operators' metrics
From time to time (each week or month) a report with the operators' metrics could be generated. The report could contain information such as conversation mean duration, number of clients answered among others specific to some hosting sites such number of customers converted into clients or number of sales. This reports would help to improve the operators' approach and to collect the most active and efficient operators/teams.

Conversation qualification
In order to improve the customers' attendance and operators' quality, at the end of a conversation the customer could qualify (star rating for instance) the quality of his conversation.

Canned responses to FAQs
To help the operators to respond faster to their clients, a list of canned responses to frequently asked questions could be available on the BO. The operator would select and send the response immediately.

Widget Personalization
A page to customize the widget could be added in order to change the widget colors, depending on the widget owner preferences.

Operators internal chat
Operators could have an internal chat network in order to help themselves to answer their customers.

Profiles
Both customers and operators could have a profile. On the customer profile all of his information could be displayed and passive of edition. On the operator profile the operator could be able to edit his information (password, name, ...) and edit his engagement rules.

Find less intrusive ways to get users' information
The information on the BO could be autocompleted during the conversation. For instance, if the customer writes his name on the conversation the BO would identify it and set it to the database.

6 Conclusions

This document reflects the work done over the internship concerning the planning, development and testing of the Online Context for Voice Communications project. The developed solution comprises an Injectable JavaScript Widget, an Angular app for the Back Office and a Node.js backend.

During first semester was done the major analysis and planning. Competitors were analyzed as well as the differentiation points between them and the project's product. The project's scope was defined, technologies choices were made and the architecture was planned. Development was also started on the first semester with the UI implementation and Context Collection.

During second semester the feature set was improved with the addition of new features which led to new challenges and architecture changes. During this semester development was finished with the implementation of chat, VoIP calls and the integration of Remote Assistant. The product was tested during development due to the methodology followed.

From a goal perspective is safe to say that the goals defined were not only achieved but were surpassed as well. Customers can easily use the injected widget to talk by chat, VoIP calls between browsers or breakout calls with operators and operators can see the context from each conversation, which were the primary defined goals for the project. Besides that, Remote Assistant demonstration was integrated to the project which enriched the final result and gave a major contribution to the product differentiation from competitors.

At the time of writing the report project's product reached a very stable release ready to be used for demonstration. In spite of that is still room for improvement both on the developed feature as by adding the new proposed features as well. The future work analysis was not left out of the project and was done, so the beginning of new features can start as soon as it is intended to.

From a personal point of view, this internship was, without any doubts, a great contribution in my education on Software Engineering Area. Every day I had to make my own decisions, face challenges and learn from my mistakes. In spite of having the help from my tutors, I always had the chance to express my opinions regarding all the options and that helped me to grow, keep focused and motivated.

At the end of this internship I am happy to say that with this internship I became a better software engineer who understands what the business reality is about. I can consider my performance a success because I helped the company by delivering to them a finished product that can be used to create valor on their portfolio.

7 References

[1] "WebRTC," [Online]. Available: https://webrtc.org/. [Accessed 14 November 2015].

[2] Vivocha, "Vivocha," [Online]. Available: http://www.vivocha.com/. [Accessed 05 October 2015].

[3] Livezilla, "Livezilla," [Online]. Available: http://www.livezilla.net/home/en/. [Accessed 05 October 2015].

[4] ClickDesk, "ClickDesk," [Online]. [Accessed 05 October 2015].

[5] ParkerSoftware, "WhosOn," [Online]. Available: http://www.whoson.com/. [Accessed 05 October 2015].

[6] Parker Software, "Parker Software," [Online]. Available: http://www.parkersoftware.com/. [Accessed 06 October 2015].

[7] Netop, "Live Guide," [Online]. Available: http://www.netop.com/live-guide/what-is-live-guide.htm. [Accessed 05 October 2015].

[8] Netop, "Netop," [Online]. Available: http://www.netop.com/. [Accessed 06 October 2015].

[9] Google Inc., "Google Analytics," [Online]. Available: https://www.google.com/analytics/. [Accessed 16 October 2015].

[10] Wikipedia, "Google Analytics," [Online]. Available: https://en.wikipedia.org/wiki/Google_Analytics. [Accessed 2015 October 2015].

[11] Adobe Systems, "Adobe Marketing Cloud," [Online]. Available: http://www.adobe.com/marketing-cloud.html. [Accessed 16 October 2015].

[12] Wikipedia, "Omniture," [Online]. Available: https://en.wikipedia.org/wiki/Omniture. [Accessed 16 October 2015].

[13] Kissmetrics, "Kissmetrics," [Online]. Available: https://kissmetrics.com/. [Accessed 16 October 2015].

[14] Wikipedia, "Customer Relationship Management," [Online]. Available: https://pt.wikipedia.org/wiki/Customer_relationship_management. [Accessed 16 2015 16].

[15] Forbes, 22 May 2015. [Online]. Available: http://www.forbes.com/sites/louiscolumbus/2015/05/22/gartner-crm-market-share-update-47-of-all-crm-systems-are-saas-based-salesforce-accelerates-lead/#2715e4857a0b515edaed4e6e. [Accessed 22 October 2015].

[16] Salesforce, "Salesforce," [Online]. Available: http://www.salesforce.com/eu/?ir=1. [Accessed 22 Ocotber 2015].

[17] Nasdq, "Salesforce.com Inc Stock Quote & Summary Data," [Online]. Available: http://www.nasdaq.com/symbol/crm. [Accessed 2015 October 2015].

[18] SAP, "SAP CRM," [Online]. Available: http://go.sap.com/solution/customer-engagement.html. [Accessed 2015 October 2015].

[19] Oracle, "Oracle CRM," [Online]. Available: https://www.oracle.com/applications/customer-experience/crm/index.html. [Accessed 2015 October 2015].

[20] Microsoft, "Microsoft Dynamics CRM," [Online]. Available: https://www.microsoft.com/pt-pt/dynamics/default.aspx. [Accessed 22 October 2015].

[21] nodejs.org, "NodeJS," [Online]. Available: https://nodejs.org/en/. [Accessed 28 October 2015].

[22] npm, Inc., "npm," [Online]. Available: https://www.npmjs.com/. [Accessed 2015 October 2015].

[23] socket.io, "Socket.io," [Online]. Available: http://socket.io/. [Accessed 28 October 2015].

[24] Express, "Express," [Online]. Available: http://expressjs.com/. [Accessed 28 October 2015].

[25] Google .Inc, "AngularJS," [Online]. Available: https://angularjs.org/. [Accessed 15 November 2015].

[26] MongoDB, "MongoDB," [Online]. Available: https://www.mongodb.org/. [Accessed 23 October 2015].

[27] Noop, "Simple vs. Complicated vs. Complex vs. Chaotic," 20 August 2008. [Online]. Available: http://noop.nl/2008/08/simple-vs-complicated-vs-complex-vs-chaotic.html. [Accessed 12 October 2015].

[28] "Scrum Guide," 2014. [Online]. Available: http://www.scrumguides.org/scrum-guide.html. [Accessed 12 October 2015].

[29] [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/5/58/Scrum_process.svg/2000px-Scrum_process.svg.png. [Accessed 12 October 2015].

[30] J. C. Helm, "Risk Management - Methods ans Tools," 1 January 2006. [Online]. Available: http://sce.uhcl.edu/helm/BB-TestRiskMan/my_files/module6/topn.htm. [Accessed 14 November 2015].

[31] S. Brown, "Simon Brown," [Online]. Available: http://www.codingthearchitecture.com/authors/sbrown/. [Accessed 12 November 2015].

[32] S. Brown, "C4 model poster," [Online]. Available: http://www.codingthearchitecture.com/2014/08/24/c4_model_poster.html. [Accessed 12 November 2015].

[33] Wikipedia, "Component-Based Software," 7 June 2016. [Online]. Available: https://en.wikipedia.org/wiki/Component-based_software_engineering. [Accessed 15 June 2016].

[34] Wikipedia, "Event-Driven Architecture," 7 June 2016. [Online]. Available: https://en.wikipedia.org/wiki/Event-driven_architecture. [Accessed 15 June 2016].

[35] "ipinfo.io," [Online]. Available: https://ipinfo.io/.

[36] Google Inc., "WebRTC internals," [Online]. Available: chrome://webrtc-internals/. [Accessed 03 June 2016].

[37] Wikipedia, "Live support software," [Online]. Available: https://en.wikipedia.org/wiki/Live_support_software. [Accessed 02 October 2015].

[38] Wikipedia, "Customer Relationship Management," [Online]. Available: https://en.wikipedia.org/wiki/Customer_relationship_management. [Accessed 2015 October 2015].

[39] Wikipedia, "NodeJS," [Online]. Available: https://en.wikipedia.org/wiki/Node.js. [Accessed 28 October 2015].

[40] Socket.io, "Get Started: Chat application," [Online]. Available: http://socket.io/get-started/chat/. [Accessed 20 October 2015].

[41] tutorialPoint, "AngularJS - MVC Architecture," [Online]. Available: http://www.tutorialspoint.com/angularjs/angularjs_mvc_architecture.htm. [Accessed 17 November 2015].

[42] codecademy, "Learn AngularJS," [Online]. Available: https://www.codecademy.com/learn/learn-angularjs. [Accessed 17 November 2015].

[43] E. Mcnulty, "SQL vs NoSQL - What you need to know," 1 July 2014. [Online]. Available: http://dataconomy.com/sql-vs-nosql-need-know/. [Accessed 23 October 2015].

[44] C. Buckler, "SQL vs NoSQL: The Differences," 18 September 2015. [Online]. Available: http://www.sitepoint.com/sql-vs-nosql-differences/. [Accessed 25 October 2015].

[45] "Is WebRTC ready yet?," [Online]. Available: http://iswebrtcreadyyet.com/. [Accessed 27 October 2015].

[46] "Can I use WebRTC?," 12 May 2015. [Online]. Available: http://caniuse.com/#search=webrtc. [Accessed 27 October 2015].

[47] W3 School, "Browser Statistics," [Online]. Available: http://www.w3schools.com/browsers/browsers_stats.asp. [Accessed 27 October 2015].

[48] S. Dutton, "Getting Started with WebRTC," 23 July 2012. [Online]. Available: http://www.html5rocks.com/en/tutorials/webrtc/basics/. [Accessed 30 October 2015].

[49] Github, "Socket.io," [Online]. Available: https://github.com/socketio/engine.io. [Accessed 04 December 2015].

[50] Strongloop, "Writing Modular Node.js Projects for Express and Beyond," 2014 October 2014. [Online]. Available: https://strongloop.com/strongblog/modular-node-js-express/. [Accessed 26 November 2015].

[51] "Securing Node.js and Express with SSL Client-Authentication," 10 June 2012. [Online]. Available: http://www.gettingcirrius.com/2012/06/securing-nodejs-and-express-with-ssl.html. [Accessed 30 November 2015].

[52] A. Marandon, "How to build a web widget (using jQuery)," 15 June 2010. [Online]. Available: http://alexmarandon.com/articles/web_widget_jquery/. [Accessed 26 November 2015].

[53] M. Vieira, *Project Management, Module 3: Planning & Tracking - Risk Management,* October, 2014.

[54] "IP Info," [Online]. Available: https://ipinfo.io/.

Masters in Informatics Engineering

Internship 2015/2016
Final Report

Online context for voice communications

Annex A – State of the Art

José Manuel Marques Grilo

jgrilo@student.dei.uc.pt

Supervisors:

Jorge Sousa
Luís Matos

Carlos Bento

July, 1st 2016

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

Index

Index of Tables

Index of Figures

1 Introduction

This document intends to expose detailed information about the competitors analyzed and about the technologies that were considered. Decisions made are also justified in this document.

The document will be divided in three big sections. The first one will give the reader bases about Live Support Online including its goals and main functionalities.

In the section 3 is performed a market analysis about competitors of the internship software, whether direct or indirect. This analysis aims to study the market's saturation level, possibilities of success, impact level and inherent risks.

Lastly, a technologies analysis was performed and is shown at section 4. Like section 3, this section aims to study the technologies available and evaluate their advantages and limitations in order to select the ones to use.

For both market and technologies analyses a brief description of the services/technologies, relevant features and main limitations was studied. Each section contains all the studied services or technologies, sorted in alphabetical order.

2 Live Support Online

Stated as an evolution from the helpdesk systems, the Live Support Online systems allow to the customer an immediate personalized service while he keep browsing the website. Unlike helpdesk systems where the service is not immediate because the customer needs to go to an assistance store or has to wait for a callback to be made or an email to be replied, in the Live Support Online this attendance is made through a popup window that allows the customer to chat or make a Voice over Internet Protocol (VoIP) call by the time the problem arise.

This system has two components:

- A chat window through which the customer communicates.
- A dashboard which allows the operator to talk back to the customers.

The chat window is usually injected through a snippet (e.g.: JavaScript (JS) code) which is pasted at the website source code and the dashboard can either be native or web developed.

The start of the conversation can be done following two approaches:

- Broadcast: the conversation is initiated by the customer that sends a message to the server and then it broadcast the message to all the available operators, which then one of them is selected to answer the call.
- Proactive engagement: in this approach the operator is who starts the conversation with a selected customer. Once the operator sends a message it will be delivered to the customer and then he opts to answer or not. The beginning of the conversation is usually defined by a customer achievement, which was specified by the operator (e.g.: number of clicks in the website). This approach is especially effective in sales due to its brute force approach.

More innovator approaches, besides providing an immediate attendance allows a personalized attendance as well. They use techniques such as Real User Monitoring (RUM), which allow them to collect data about their customers and customers’ navigation. This approach makes the service more efficient avoiding the context exchange between customer and operator.

3 Competitors

The competitors against the internship product will be described in this section. Two groups can be created:

- Direct competitors: it was stated as a direct competitor the company that presents a service or application with at least some of the features that the internship product aims to develop.
- Indirect competitors: it was stated as an indirect competitor the company that presents a service or application with different features from the ones that the internship product aims to develop, yet can still offer a service that can compete against the internship

3.1 Direct Competitors

3.1.1 Competitors Analysis

Nowadays, there is a set of companies offering a competitor service like the one proposed in the internship. For a matter of convenience, and since that there are many similarities in the services, only the most popular companies were selected. The companies were selected based on the amount of references in the search.

Below there is an analysis for each one of the competitors. Their comparison is done at the next section.

LiveHelpNow [1]

LiveHelpNow was founded in 2003 by Michael Kansky, current company's CTO. The company's headquarters are in Willow Grove, Pennsylvania, USA.

It already counts with more than 120000 deployments and it has customers like Discovery Education, NBC Sports or Dell among others. With more than 65% of its personnel focused in the development area the company provides, apart from its live chat solution, a call and ticket management solution and an information and knowledge database.

The live chat solution allows real time monitoring over customers' actions to increase sales. Beyond that, operators and communications are also monitored in order to know their efficiency.

Main features:

- Real time monitoring.
- Easily exportable customers list.
- Monitors information from the customers' social media profiles open in the browser.
- Returning customers identification.

Negatives:

- Do not allow voice calls.

LiveChat [2]

With offices in Boston, Massachusetts, USA and Wroclaw, Poland, the company was founded by Mariusz Cieply, current CEO, in 2002.

LiveChat provides a live support online through chat used by 12000 customers.

Main features:

- Real time monitoring.

Negatives:

- Both customer and operator's applications must be downloaded and installed.

Zopim [3] **by Zendesk** [4]

Zendesk has Mikkel Svane, company's CEO, Morten Primdahl, CTO, and Alexander Aghassipour as co-founders. The company was founded at Copenhagen, Denmark when they still used a kitchen door as desk. The company's purpose is to bring a little Zen to the chaotic world of help support, with appellative and simple software.
Since its foundation, the company already have its headquarters at San Francisco, California, USA. Nowadays is based at Tenderloin, California, USA.
In its set of customers there are some relevant names such as L'Oréal, Foursquare, Xerox and Vodafone.
Zendesk offers a set of solutions like help center, chat and voice, which can be integrated between them.
Main Features:

- Real time monitoring.
- Multiples CRM and salesforce software integrations available.
- Allow multiple operators and chat transfer.

Negatives:

- Do not allow voice calls.
- Do not allow URL push to the client.
- The widget needs to be downloaded and installed.

Olark [5]

Olark was founded in 2009 at Ann Arbor, Michigan, USA. Its founders, Bem Congleton, Matt Pizzimenti, Roland Osborne and Zach Steindler, started the company at Y Combinator, a seed accelerator, and ended up creating a successful company. Their purpose is help to solve customers' communication problems in an immediate way and gain information about what they want. The company is currently based in San Francisco, California, USA and has employers dispersed in areas like USA, Canada and UK.

Its service allows an easy installation of a customizable widget and a great set of CRM integrations.

Main features:

- Real time monitoring.
- Multiples CRM and salesforce software integrations available.
- The widget is easily installed, through copy & paste of a snippet.

Negatives:

- Do not allow voice calls.

Vivocha [6]

Vivocha is a self-called startup with offices in San Francisco, California, USA, Milan and Cagliari, Italy. It was founded by Gianluca Ferranti and Federico Pinna, currently the company's CEO and CTO.

Beyond its monitoring features, a large set of communications combinations are offered with the possibility of chat, voice, video and callbacks.

Main features:

- Real time monitoring.
- Web dashboard to monitor and assist multiple customers.
- Multiple communication options: chat, voice, video and callbacks.

Livezilla [7]

Livezilla GmbH was founded in 2009 and is based at Singen, Germany.

The company offers a set of integrated solutions of customer support. This set includes a live chat for communication between customers and operators with the possibility of proactive engagement by the operators, real time monitoring and a tickets system to allow offline contact and customers support. Besides that, operators can communicate with each other, both from their dashboard and mobile app.

Main features:

- Real time monitoring.
- Helpdesk and ticket system for offline service.

Negatives:

- The widget needs to be downloaded and installed.
- Do not allow voice calls.

BoldChat [8] **by LogMeIn** [9]

BOLDCHAT

LogMeIn was founded by Michael Simon, current CEO, in 2003 and it is based in Boston, Massachusetts, USA. Beside its headquarters, the company has offices at the USA, India and Australia.

The company offers a set of products with four different focus areas: collaboration services, IT management services, live support and services about the Internet of Things. Some of its products include the LogMeIn Hamachi for net virtualization and VPN services, and the join.me software, an online collaboration tool.

Boldachat is the product which gives the users a chat system with clients monitoring and the ability of proactive engagement from the operators.

Main features:

- Real time monitoring.
- Integration with salesforce software

Negatives

- Do not allow voice calls.
- Do not allow URL push to the client.
- Both customer and operator's applications must be downloaded and installed.
- Do not allow offline conversations.

Provide Support [10]

Based in New York, USA, Provide Support has the purpose of rise the companies' power that use its service with communication software that help to develop communication relationships between customers and providers and raise the sales numbers.

Company's solution offers a customizable chat, easy to integrate, and capable of monitor clients and offer a personalized attendance.

Main features:

- Real time monitoring.
- Easy integration of the widget and web dashboard available
- Online clients list.

Negatives

- Do not allow voice calls.
- Do not recognize returning customers.

Tawk.To [11]

The company was founded in 2010 and is based at Riga, Latvia, however it assumes a distributed policy where its employees are out of the headquarters.

Its chat is highly customizable, since its color, position and even language. In terms of monitoring, it is possible to collect all the tracked pages and geolocation in real time, which can be seen in the dashboards developed for Windows, OS X, Android and iOS.

Main features:

- Real time monitoring.
- Multiple operators over multiple departments with the possibility of client routing.
- Conversation history.
- Highly customizable widget.

Negatives:

- Do not allow voice calls.
- Do not allow URL push to the client.
- Both customer and operator's applications must be downloaded and installed.
- Do not allow offline conversations.

Kayako [12]

Founded in 2001 by the current CEO, Varun Shoor, the company is based in London, UK. Kayako is one of the greatest Live Support Online companies, with clients such as Peugeot, Nasa and Avast.

Beyond its real time monitoring capabilities and native applications for mobile operating systems, Kayako allows voice calls using an installed service, as well as callbacks from operators.

Main features:

- Real time monitoring.
- Possibility of voice calls.

Negatives

- The dashboard needs to be downloaded and installed.

ClickDesk [13]

ClickDesk is a self-funded company based in Silicon Valley. The company was founded after a search made for a live support chat with the ability to receive phone calls instantly.

Besides the chat and phone abilities, the company intended to keep the service cloud-based.

Main features:

- Multiple communication options: chat, voice.
- Multiples CRM and salesforce software integrations available.

Negatives:

- Do not track customers' heat map.
- Do not allow URL push to the client.

Comm100 [14]

Comm100 has the motto "100% communication, 100% success". They believe that communication is the best way to make the difference and create great relationships so, they have a set of solutions communication-oriented: live chat, email marketing, ticket/email, helpdesk, knowledge base and forum. Some of its customers are Whirlpool, sears and Stanford University.

Main features:

- Real time monitoring.
- Possibility of screen sharing.

Negatives:

- Do not allow voice calls.
- Do not allow URL push to the client.

WhosOn [15] **by Parker Software** [16]

Parker Software was founded in 2003 and it is a company based on real-time communication. Its main product is WhosOn, first launched in 2002 with the purpose to track and analyse customer's journey across a website. In 2007, WhosOn was developed into a live chat and tracking solution, used to analyze, engage and chat with customers.

Main features:

- Real time monitoring.
- Possibility of screen sharing.
- Possibility of voice calls.

Negatives:

- Do not allow URL push to the client.

Live Guide [17] **by Netop** [18]

Netop employs 130 people and has subsidiaries in the USA, Great Britain, China Romania and Switzerland and it is headquartered in Denmark.

The company believe in a world where people can connect with anyone, anywhere, anytime. For that reason they have three products oriented to online connection: Live Chat with chat, voice and video call, Secure Room Control for remote access and Classroom Management with screen sharing and student supervision.

Main features:

- Real time monitoring.
- Possibility of screen sharing.
- Possibility of voice calls.

Negatives:

- Do not allow URL push to the client.

Website Alive [19]

AliveChat started as an experiment in Austin, Texas, USA. The concept evolved and now is present in more than 11000 web and mobile sites over 120 countries. Since 2004 the company has won some awards with its products.

Main features:

- Real time monitoring.
- Possibility of screen sharing.

Negatives:

- Do not allow voice calls.

3.1.2 Relevant features

Live support online systems allow communication between a customer and operator in a way that the customer can clear his doubts and questions immediately. However, with the technology evolution and the rise of new concepts to satisfy the customers' demand, new features need to be added. This will allow for a simpler, faster and more efficient attendance system that requires less time and effort both from customers and operators.

To make a complete analysis from the competitors a features' survey was performed. It was taken in count all the features that not only allow the customer to communicate with an operator but also the features that allow the operator to collect navigation context among others. However, many of the analyzed features will be out of the scope of the internship, so a small set of features was selected that allows a comparison between the competitors and the internship product.

The set of features created is a set that was considered to be the minimal acceptable set of features to create a product that allowed costumers to ask for help to operators while operators could see the navigation context. This minimal acceptable set must have at least communication features such as chat and VoIP and context collection features in Real Time. Besides that, the customers' widget injection and a web back office dashboard for operator are also requirements.

The selected set was:

- **Web dashboard**: A web dashboard that allows the operator to communicate with the customers and see their navigation data.
- **Proactive engagement**: A chat conversation may be started every time a defined rule is achieved by a customer; for example if the number of page visited is equal to α.
- **Prioritize chats**: The displayed customers assigned to each operator should be listed in a First-In-First-Served way.
- **Page tracking**: The pages visited inside the website must be saved in order to create a heat map for each customer.
- **Referrer tracking**: The referrer to the website must be saved for each visit.
- **Browser Tracking**: The browser version used in every visit must be saved.
- **Geolocation Tracking**: The operator must see the customer's geolocation if they are allowed.
- **Real time monitoring**: The monitoring of the tracked data must be done in real time during the customer's visit.
- **Online customers list**: An online customers list should be provided to every online operator.
- **Recognize customers**: Returning customers must be identified and their previous information should be loaded to the dashboard.
- **Chat**: Customers should be able to start a chat conversation with an online operator.
- **VoIP without plugins**: Customers should be able to start a voice call with an online operator without the need to install any plugins.
- **Operators login/logout**: Registered operators should be able to login and log out at the dashboard.
- **URL Push**: Operators should be able to push pages to their customers' browser and force them to load that page.
- **Multiple chat windows**: Operators should be able to have multiple conversations at the same time.
- **Chat transfer**: Operators should be able to transfer chat conversations between themselves.
- **Proactive engagement rules**: Operators should be able to configure a set of rules to start proactive engagement with their customers.
- **Widget injection**: The widget should be injected in the websites through a snippet, without the need of software installation.

3.1.3 Comparative analysis

In order to easily compare direct competitors, first a table with their features developed by each company is presented. After that, a written summary will highlight the main competitors from the ones analyzed and how the internship product can stand in the market.

The table below compares the services of the companies. The check sign (✓) indicates that the feature is supported, on the other hand, the cross sign (✗) indicates that the feature is not supported. Since it is very hard to collect all the data from the companies' products due to the inability to test all software, the circle sign (●) is used every time that no information is supported or available. For a more complete table, with all the features analyzed, please refer to chapter 6, section 6.1 – **Full Competitors Comparative Table**.

	Livezilla	Tawk.to	Vivocha	Provide Sup-	Zopim	BoldChat	Live Help Now	Kayako	Olark	LiveChat	ClickDesk	Comm100	Netop Live	WhosOn	website Alive
Dashboard															
Responsive dashboard	✗	●	✓	●	●	✗	●	✗	✗	●	✓	●	✗	●	●
Proactive engagement	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Prioritize chats	✓	✗	●	●	●	✗	●	●	●	●	●	●	●	●	●
Widget															
Page tracking	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Referrer tracking	✓	●	✓	✓	●	✓	✓	✓	●	✓	✓	✓	✓	✓	✓
Browser tracking	✓	●	✓	✓	●	✓	✓	✓	●	●	✓	●	●	●	●
Geolocation tracking	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Real time monitoring	✓	✓	✓	✓	✓	✓	✓	✓	✗	●	✓	✓	✓	✓	✓
Online customers list	✓	✓	●	✓	✓	●	●	●	✓	●	✓	●	✓	●	●
Recognize customers	✓	✓	●	✗	●	✓	✓	●	●	✓	●	✓	✓	●	✓
Customer															
Chat	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
VoIP without plugins	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	●	✓	✗
Operator															
Operators login/logout	●	✓	✓	●	✓	●	●	✓	●	●	●	✓	●	✓	●
URL push	✓	✗	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	●
Multiple chat windows	✗	●	✓	✓	✓	●	●	●	✓	✓	✓	✓	●	✓	✓
Chat transfer	✓	✓	✓	✓	✓	●	✓	✓	●	✓	✓	✓	✓	✗	✓
Proactive engagement rules	✓	✓	✓	✗	✓	✓	●	●	✓	●	●	✓	✓	✓	✓
Owner															
Widget injection	✗	✗	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 3.1. Direct competitors analysis

Among the features between them, the final result presented is similar when compared, for example the proactive engagement allows operators to start a chat conversation without the customer allow to, through operator's initiative or the achievement of some kind of metric.

As stated previously, it is difficult to know exactly the features that are presented in each competitor due to the difficulty to test all services. Some features like the tracking ones are difficult to know if they are present because some documentation only states that the widget tracks visited pages among other information.

After evaluating the market we can see a small niche that is not well explored. The niche is the VoIP calls, which is one of the main features to develop in the internship. This could be a huge opportunity to explore that can bring value and be a point of differentiation.

3.2 Indirect Competitors

In this section will be analyzed the indirect competitors, those who in some way offer a possible solution to the customers even if that it is not their purpose.

The indirect competitors were divided in two groups: relationship management and context analysis software and Customer relationship management software.

3.2.1 Relationship management and context analysis

In this subsection will be analyzed relationship management and context analysis software. This software collects a set of data, useful from the website proprietary point of view, which allows the user to access their websites stats about visits.

The set of companies that were chosen was based on the number of references presented in the search.

Google Analytics [20]

In 2005 Google Inc. bought Urchin. This move led to the creation of Google Analytics. Its success was huge, which made new requests to the service to stay suspended after the first week. Since that, new requests were attended in a lottery way, until 2006. Since September 2006 the system became available to all users. Nowadays, Google Analytics is used over 55% of the 10000 more popular sites and over 49.95% of the 1000000 first sites from the Alexa rank.

Adobe Marketing Cloud [21]

The company was founded by Josh James and James Pestana, and financed by venture capitalists. Becoming one of the greatest powers, among the 500 private companies with the biggest growth rate and nominated by Inc. Magazine [22]. The company was acquired by Adobe Systems in 2009, and until 2011 operated as a company inside Adobe. In 2012 Adobe started to remove the name Omniture and its products started to be integrated on the Adobe Marketing Cloud.

Kissmetrics [23]

Founded by Neil Patel and Hiten Shah in 2008, it is based in San Francisco, California.

The company offers a set of tools to collect data about how the customers interact with websites, web apps and mobile products.

3.2.2 CRM Software

According to Philip Kotler, conquering new clients can cost 5 to 7 times more than maintaining the existing ones [24]. For that reason companies should have strategies that allow loyalty from their clients.

Customer relationship management is a strategy that focus on understanding and anticipating the client needs. CRM is used to help in this strategy, which allows:

- Client management
- Identify and define clients profile
- Manage communication with client
- Follow orders
- Anticipate the market evolution
- Organize a personalize technical assistance

This section will analyze the four CRM software with the bigger market share [25].

Salesforce [26]

Salesforce.com is a cloud computing company, based in San Francisco, California, USA. It was founded in 1999 by Marc Benioff, Parker Harris, Dave Moellenhoff and Frank Dominguez. Nowadays is one of most valuables cloud computing companies, evaluated around 50 thousand million dollars.

The company splits its products over six areas: sales, marketing, community, analysis and applications. Its products are used for over 100000.

Market Share: 18.6%

Growth: 28.2%

Main features:

- Offers a fast and personalized service to the customers
- Create and manage personalized marketing campaigns
- Allow the approach of customers and clients independently their places
- Collect and display tracked data in a dashboard

SAP CRM [27]

SAP SE is a German software company that makes enterprise software to manage business operations and customer relationships. Its CRM software targets midsize and large organizations in all industries.

Its solutions includes the modules of sales, marketing, services, analytics, interaction center and web channel.

Market share: 12.1%

Growth: 7.2%

Oracle CRM [28]

Oracle started in this market space when they purchased Siebel Systems in 2005, a company focused on developing software to support CRM. Later, they bought Upshot CRM, which brought a more robust interface.

The company split its products in six areas: marketing, sales, commerce, social, services and CPQ (Configure, Price and Quote). Its customers can either subscribe to a single module or a combination of modules.

Market share: 9.2%

Growth: 2.6%

Main features:

- Manage individual experiences over a set of channels.
- Allow personalized experiences to its customers.
- Collect data from customers.
- Easy communication between customers and operators.

Microsoft Dynamics CRM [29]

Microsoft Dynamics CRM is part of the Microsft Dynamics package of CRMs and Entity Resource Planning (ERP). Launched in 2003, it is a client-server accessable apllication from both browser and a plugin for Microsoft Outlook. The current version count with sales, marketing, services and social modules. It counts about 40,000 customers.

Market share: 6.8%

Growth: 21.7%

Main Features:

- Sales analysis
- Account management
- Multichannel assistance
- Personalized assistance and marketing design

3.2.3 Summary

After the analysis of the indirect competitors, neither the Relationship management and context analysis software nor the CRM software are real menaces to the internship product, because its focus is not the one to track and assist. The first group is focused on data collection to context from the visits done to the website, however they cannot give assistance to the users. The CRM group, can give some assistance in real-time but its real focus is to collect and manage relationships in order to improve the communication and assistance of the customers.

In spite of not being a big menace, the integration of CRM software in the internship product should be considered. This integration can be useful to provide a better experience and assistance to the customer. If so, the first CRM software to consider should be Salesforce due to its market share, and, still show a greater growth than its competitors. Due to their market share, the other CRM software products are not a priority.

4 Technologies

Choices of technologies can influence a project. Choosing an unfamiliar or untested technology can bring unwanted risks to the project. On the other hand, choosing the right ones can make the development process easier with much better final results.

In order to choose the right technologies the intern made an analysis for the main components developed: Online Context for Voice Communications (OCVC) Server and Back Office (BO) Web App. Besides that, the database selection was also an important decision.

The intern had full liberty to analyze and choose the technologies. After making his analysis the intern presented his choices to Internship's Tutor Luís Matos who accepted the proposed technologies.

This section will be show the technology choices to develop the internship product. First to be presented is the list of the backend technologies, and second is a subsection of the frontend technologies.

The third subsection will present the database selected.

Finally a section about WebRTC is presented.

4.1 Backend Technologies

Java [30]

Java is a general-purpose computer programming language that is concurrent, class-based and specifically designated to have as few implementation dependencies as possible. Developed in 1991 by James Gosling at Sun Microsystems (acquired by Oracle in 2010), it was released in 1995 and nowadays is one of the most popular programming languages in use, particularly for client-server web applications. Java was projected to be:

- An objected oriented programming language.
- Portable.

- A language with an extensive network of resources.
- Secure.

PHP [31]

PHP is a server side scripting language designed for web development. It is one of the most web languages used for websites and web servers. It was created in 1994 by Rasmus Lerdorf and originally stood for Personal Home Page, it now stands for Hypertext Preprocessor.

PHP is an imperative, object-oriented language which can mix HTML code with its own and send the result to the client usually in the form of a web page. Until 2014 it had a *de facto* standard, since then there is an ongoing work on creating a formal PHP specification.

JavaScript on Node.js [32]

Node.js is a recent cross-platform runtime environment used for server-side applications written in JavaScript. It provides an event-driven architecture and a non-blocking I/O Application Programming Interface (API) ideal for real-time web applications. Besides its recent creation, in 2009 by Ryan Dahl and Joyent, it is already in use by some great companies such as LinkedIn, IBM or Microsoft.

4.1.1 Summary

Since the goal of the internship product is to build a web application that allows real-time communication between clients and operators without demanding Central Processing Unit (CPU) processing. The internship product will be developed as a multi-tier chat application – two different web clients (clients and operators) and a web server – where the clients' application will communicate between themselves through the server. So, the server will be a lightweight application that only needs to forward information every time a new request arrives. What is needed is an event-driven architecture with a fast non-blocking API, which is what Node.js offers.

Applications that run on Node.js are written in JavaScript, the same language can be used both on the client side and on server side.

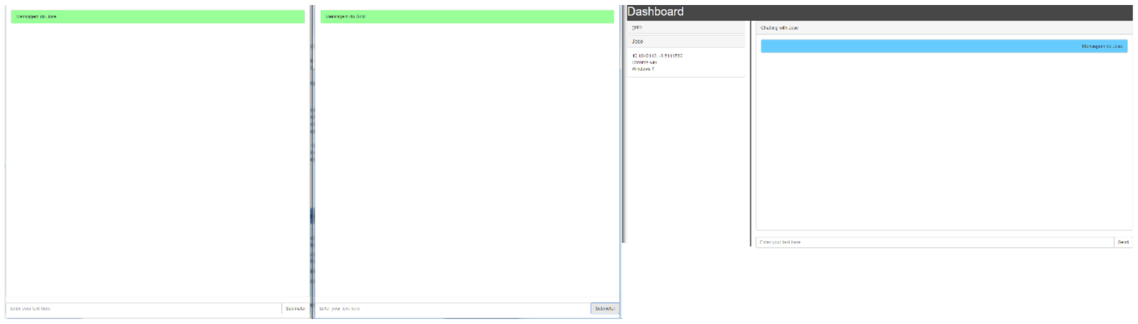
The use of Node.js allows the use of several modules from the package modules repository – npm [33] – such as:

- Socket.io [34]: is cross-platform module that allows real-time bidirectional event based communication, which is great to establish communications between users (clients and operators) and server. This is an ideal module for real-time analytics and chat application. Besides that, it allows communication both via sockets or polling.
- Express [35]: is a Node.js framework that provides a robust set of features for web applications development, designed to quickly build a single-page, multiple-page or hybrid web applications.

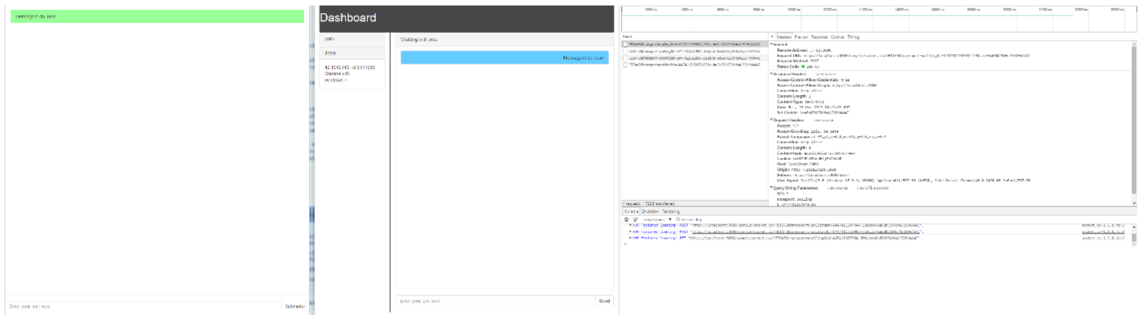
4.1.2 Study Case

Due to the lack of good documentation, a study needed to be performed in order to prove that both Node.js and Express allow a one-to-one chat implementation, and that Socket.io allows the creation of connections other than web sockets.

In the first step a tutorial present at [36] allowed the development of a chat application, with Express and Socket.io. Since Socket.io framework uses an event driven architecture the code was easily changed to create an application that would chat with a single operator.



In a second step, it was necessary to change the communication from web sockets to long polling. Socket.io allows this configuration when a new connection is created.



This small proof of concept allowed to validate the technology selected.

4.2 Frontend Technologies

AngularJS [37]

AngularJS is a JavaScript framework, written in JavaScript that extends HTML with directives. It enables the creation of dynamic views in web applications, through a two-way-data-binding, which automatically synchronize both models and views. This abstraction from the DOM leads to better code decoupling (view separated from model and controller).

AngularJS enables the creation of dynamic applications with several views without too much complexity.

4.3 Database

Database choice is a critical decision to make in every project. Every database has its pros and cons and they should be taken into account when making this choice.

However, database dependency can be minimized with the right architectural choices. In the current project server architecture follows a component-based development pattern (see **Annex C – Architecture**). This pattern makes it easier to replace the database by only changing the database connector, which is an abstraction module to communicate with the database.

Both SQL and NoSQL databases were considered. In this section they will both be analyzed and a choice will be made.

4.3.1 SQL Databases

Structured Query Language is a special programming language designed for managing data held in relational database management systems (RDBMS). SQL databases are table based, which means that their data is stored in a predefined set of tables with a predefined number of rows (schema). Since a predefined schema is needed this kind of databases are best for well-defined problems where all the data needed is known and it will not change quickly. SQL databases scale vertically, more resources means more scalability. They are best suited for heavy transactional work and complex queries.

4.3.2 NoSQL Databases

On the other hand, NoSQL (“non SQL” or “Not only SQL”) provides a storage mechanism modeled as a means other than well-defined schemas. Their data can be persisted in a collection of key-value pair, graph model or wide-columns stores which do not have a schema defined. NoSQL databases are horizontal scalable, which means that they scale if more nodes are added.

Since they use a dynamic schema, they are well-suited for problems with unstructured data, hierarchical data, or big data.

4.3.3 Analysis

Both SQL and NoSQL databases have its advantages.

In this project a little amount of relationships are needed, since only operators, visitors, conversations and tracked data entities exist. This kind of relationship would indicate the use of a SQL database. However, the structure of the entities is uncertain. More tracked fields could be added, visitors and operators profile could change when more features are added to the product. That is the most important reason to use a NoSQL database. This free-schema paradigm allows a fast development, which is critical to this internship. Another good reason to use NoSQL is the huge need for scalability. Tracked data and conversations will make the database size grow fast, so a database that scale horizontal is better. Besides that, no complex queries are needed, so NoSQL is suited to this case.

MongoDB [38] was the database selected. It is a NoSQL database that favors a JSON-like documents structure with dynamic schemas. This allows for faster and easier integration of new data. MongoDB also allows a high insert load, which is great to persist data in the database. Since the application is always recording data from visitors and chats can scale quickly, a huge amount of inserts will be done.

Besides its technical advantages, MongoDB and Node.js have a mature connection very tested, and used, which means that they are two technologies that work very well together.

4.4 WebRTC [39]

4.4.1 What is it?

WebRTC is an open project that tries to standardize the support for real-time communications via simple APIs, allowing web applications to send data between devices over IP. Its mission is to offer rich and high-quality applications the power to be developed for devices and allow them all to communicate via a common set of protocols, using Javascript APIs and HTML5.

Since WebRTC allows real-time communication into websites, sending data and media streams in a peer-to-peer way, it means it can be used for voice and video calls directly from a web browser without the need of plugins. Because of this, many communication solutions can be developed, solutions that can be innovative in the communication area.

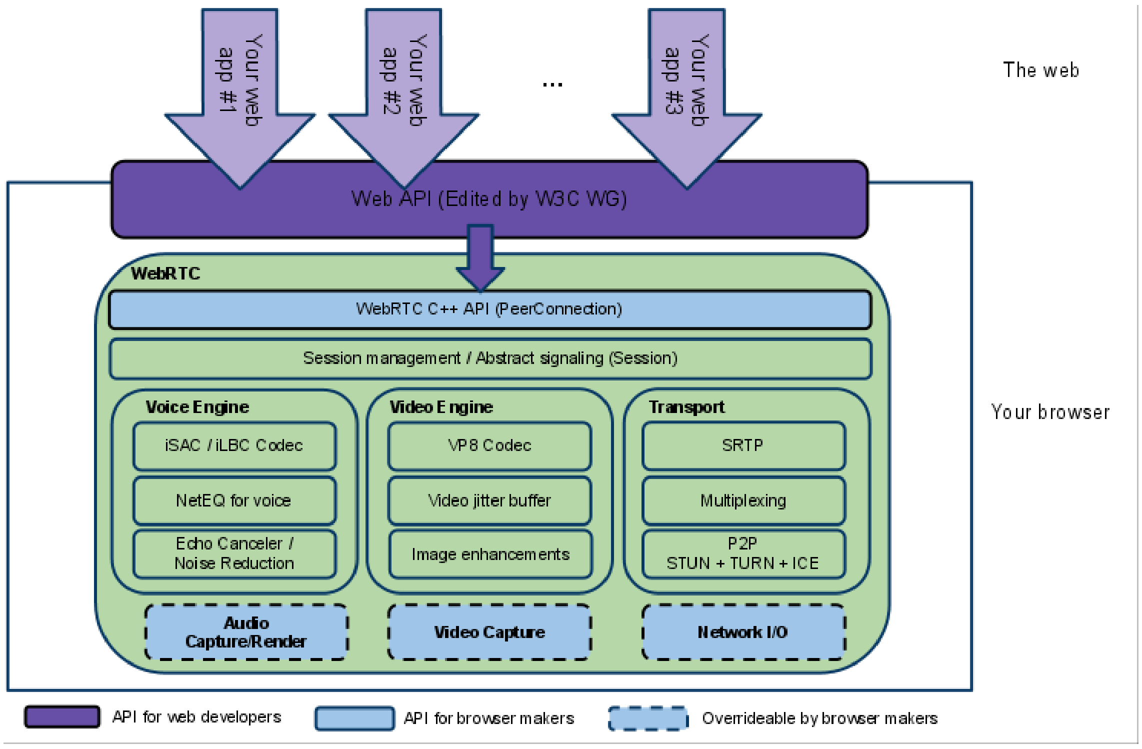


Figure 4.1. WebRTC Architecture [40]

WebRTC architecture can be divided in two layers:

- WebRTC C++ API and the capture/render hooks for browser developers
- Web API for Web App developers.

Inside browser implementation there is WebRTC C++ API enables browser developers to implement WebRTC API proposal. Then there is an abstraction layer for communication management for call setup and session management. Bellow, are three big modules. Voice and video engines are responsible for voice and video collection and presentation, they are also responsible to sending them to Transport module who is responsible for RTP stack management and STUN/ICE mechanisms to guarantee connectivity across different networks. On the last layer are the hooks that are overridden by browsers to access the device's audio and video.

WebRTC API is the part that concerns the internship. It allows the third party web based application to use WebRTC C++ API on browser.

WebRTC simplifies developers's work on three points:

- Transport layer;
- Codecs agreement;
- Media Engine.

4.4.2 Benefits

Some of the main benefits of WebRTC are related to improved productivity and team collaboration using clientless web-based video inside and outside companies, enhanced flexible work by enabling internet calling and improved relationships with customers with web-based video and video communication.

We can sum up the benefits of WebRTC in five different scenarios:

- Consumer to Consumer (common users call to people they know)

- Consumer to Business
- Business to Consumer
- Within business organizations
- Business to Business

4.4.3 WebRTC, Flash and ORTC

Before WebRTC browser's real-time communication were usually done using plugins such as Flash Player.

Flash is ubiquitously available for all browsers and acts like a patch on them to allow communication in real-time. The ubiquity of Flash is its greatest advantage over WebRTC. However, WebRTC's abstraction for developers, high quality codecs and built-in browsers API make WebRTC a much better choice.

WebRTC and ORTC are very similar between themselves. They both work over IP for web and mobile devices with the intention of improve real-time communications. The main differences between them is the level of abstraction of the offered API. ORTC claims that WebRTC API is too much high level and do not offer enough control to developers to work. Besides that, ORTC has an undefined signaling methodology in contrast with WebRTC's slightly defined SDP signaling.

4.4.4 Codecs

WebRTC supports a very limited set of codecs. Unfortunately, there is no "legal" way to choose the desired codec in a certain application. This choice is done by browsers agreement using SDP. SDP packets can be hacked and played to choose the desired codec, however that is not a good practice.

Audio Codecs

OPUS

OPUS is the codec used by default for encoding audio stream. With a constant and variable bitrate, from 6kbit/s to 510kbit and sampling rates from 8 kHz to 48 kHz is the best for high-quality audio. It implements lossy audio compression.

iSAC

This codec is well suited for voice data and streaming audio, but is not for high quality audio streams. It has an adaptative and variable bitrate, from 10kbits/s to 52kbits/s and supports 32 kHz sampling rate.

iLBC

It is the codec that is used on bad channels and low bandwidth, with a fixed bitrate of 15.2kbit/s or 13.33kbit/s and a sampling rate of 8 kHz.

G.711

It is the standard codec for audio companding and is primarily used in telephony, with a sampling rate of 8 kHz and a bitrate of 64kbit/s.

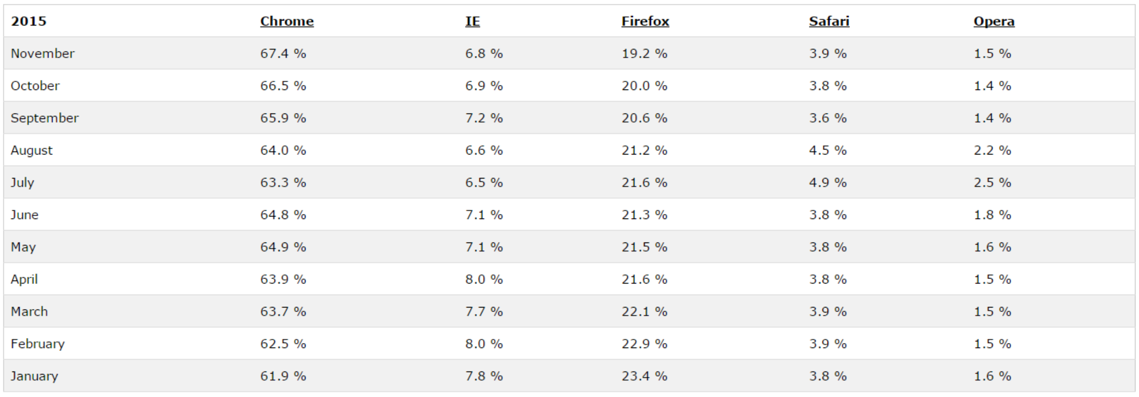
Video Codes

VP8

VP8 implements a high-efficient video compression technology. While only Firefox supports H.264 natively and VP9 is supported by Chrome and Firefox only, currently VP8 is the only video codec supported by all browsers.

4.4.5 Browsers Support

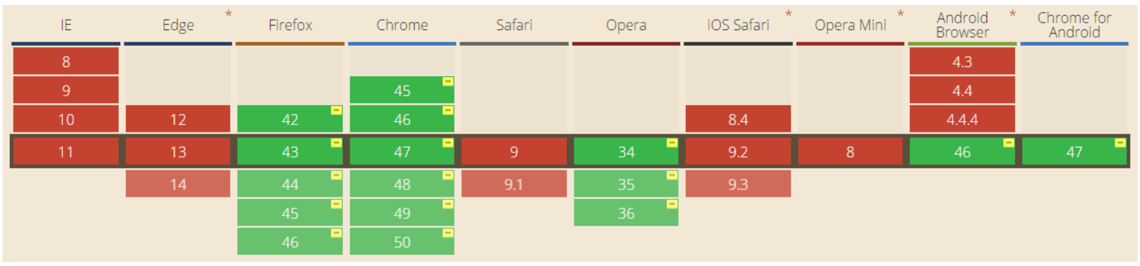
In terms of browser support, not all browser support WebRTC, however WebRTC is supported by the browsers with the biggest market share, Chrome and Firefox.



2015	Chrome	IE	Firefox	Safari	Opera
November	67.4 %	6.8 %	19.2 %	3.9 %	1.5 %
October	66.5 %	6.9 %	20.0 %	3.8 %	1.4 %
September	65.9 %	7.2 %	20.6 %	3.6 %	1.4 %
August	64.0 %	6.6 %	21.2 %	4.5 %	2.2 %
July	63.3 %	6.5 %	21.6 %	4.9 %	2.5 %
June	64.8 %	7.1 %	21.3 %	3.8 %	1.8 %
May	64.9 %	7.1 %	21.5 %	3.8 %	1.6 %
April	63.9 %	8.0 %	21.6 %	3.8 %	1.5 %
March	63.7 %	7.7 %	22.1 %	3.9 %	1.5 %
February	62.5 %	8.0 %	22.9 %	3.9 %	1.5 %
January	61.9 %	7.8 %	23.4 %	3.8 %	1.6 %

Figure 4.2. Market share for the main browsers

[41]

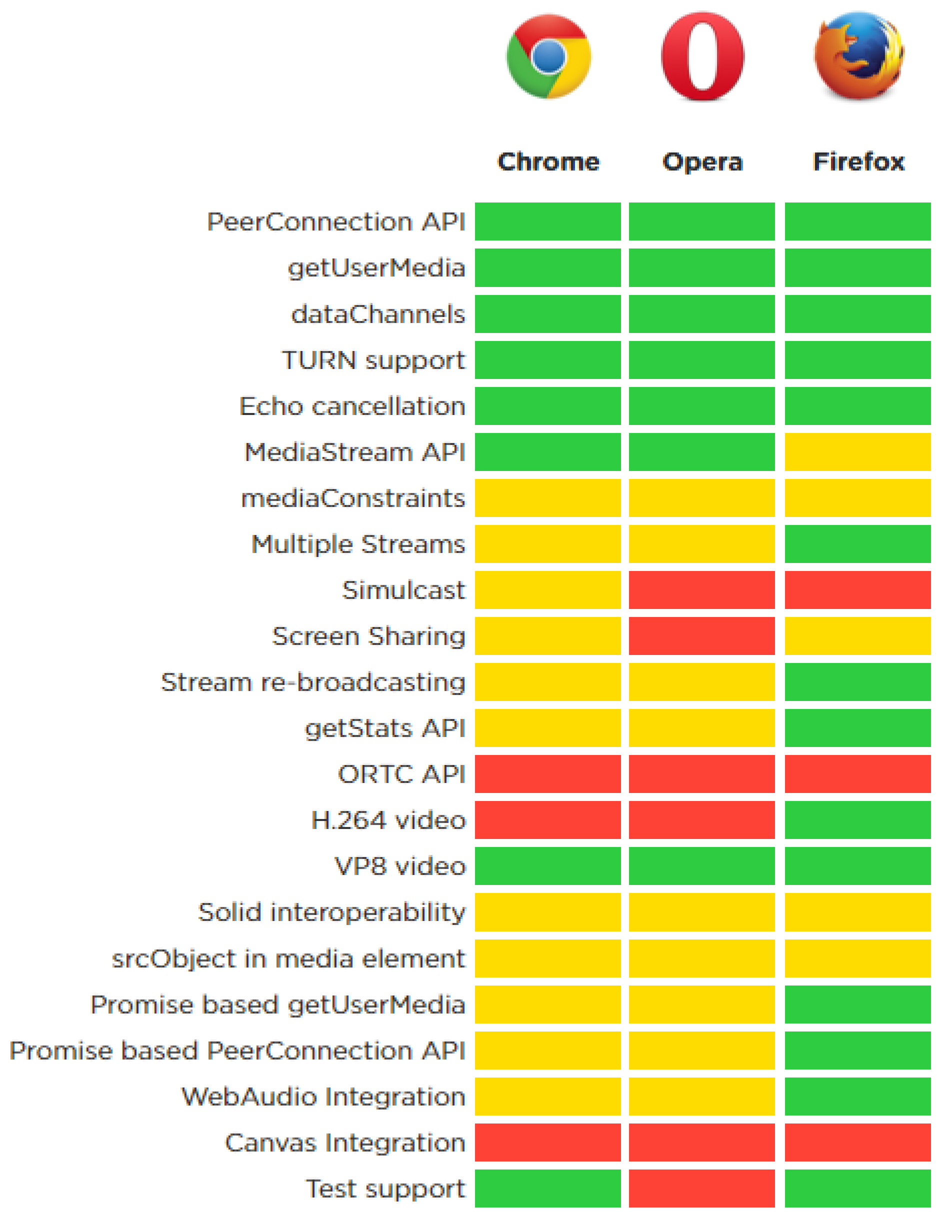


[42]

Figure 4.3. Browser support for WebRTC

As we can see on Figure 4.3, only destop/laptop browsers support WebRTC. However, as Figure 4.2 shows, Firefox and Chrome dominate the market. After them, IE is the most used browser, however its market share is decreasing rapidly over time.

On the other hand, the WebRTC project does not offer the same capabilities to all browsers. As we see in the image bellow, Firefox is the browser with the most capabilities, followed by Chrome and Opera.



	Chrome	Opera	Firefox
PeerConnection API	🟩	🟩	🟩
getUserMedia	🟩	🟩	🟩
dataChannels	🟩	🟩	🟩
TURN support	🟩	🟩	🟩
Echo cancellation	🟩	🟩	🟩
MediaStream API	🟩	🟩	🟨
mediaConstraints	🟨	🟨	🟨
Multiple Streams	🟨	🟨	🟩
Simulcast	🟨	🟥	🟥
Screen Sharing	🟨	🟥	🟨
Stream re-broadcasting	🟨	🟨	🟩
getStats API	🟨	🟨	🟩
ORTC API	🟥	🟥	🟥
H.264 video	🟥	🟥	🟩
VP8 video	🟩	🟩	🟩
Solid interoperability	🟨	🟨	🟨
srcObject in media element	🟨	🟨	🟨
Promise based getUserMedia	🟨	🟨	🟩
Promise based PeerConnection API	🟨	🟨	🟩
WebAudio Integration	🟨	🟨	🟩
Canvas Integration	🟥	🟥	🟥
Test support	🟩	🟥	🟩

[43]

Figure 4.4. WebRTC support for the main browsers

In terms of capabilities offered on the three browsers that support WebRTC, is fair to say that Firefox is the most complete one. However, all of them offer the needed capabilities to implement voice communication.

5 References

[1] LiveHelpNow, “Featured Benefits,” [Online]. Available: http://www.livehelpnow.net/help-desk-software. [Acedido em 06 October 2015].

[2] LiveChat, Inc., “LiveChat,” [Online]. Available: https://www.livechatinc.com/. [Acedido em 06 October 2015].

[3] Zendesk, “Zopim,” [Online]. Available: https://www.zopim.com/. [Acedido em 06 Ocotber 2015].

[4] Zendesk, “Zendesk,” [Online]. Available: https://www.zendesk.com/. [Acedido em 06 October 2015].

[5] Olark, “Olark,” [Online]. Available: https://www.olark.com/. [Acedido em 06 October 2015].

[6] Vivocha, “Vivocha,” [Online]. Available: http://www.vivocha.com/. [Acedido em 05 October 2015].

[7] Livezilla, “Livezilla,” [Online]. Available: http://www.livezilla.net/home/en/. [Acedido em 05 October 2015].

[8] LogMeIn, “Boldchat,” [Online]. Available: https://www.boldchat.com/. [Acedido em 06 October 2015].

[9] LogMeIn, “LogMeIn,” [Online]. Available: https://secure.logmein.com/PT/. [Acedido em 06 October 2015].

[10] Provide Support, “Provide Support,” [Online]. Available: http://www.providesupport.com/. [Acedido em 06 October 2015].

[11] Tawk.to, “Tawk.to,” [Online]. Available: https://www.tawk.to/. [Acedido em 06 October 2015].

[12] Kayako, “Kayako,” [Online]. Available: http://www.kayako.com/. [Acedido em 06 October 2015].

[13] ClickDesk, “ClickDesk,” [Online]. [Acedido em 05 October 2015].

[14] Comm100, “Comm100,” [Online]. Available: http://www.comm100.com/. [Acedido em 06 October 2015].

[15] ParkerSoftware, “WhosOn,” [Online]. Available: http://www.whoson.com/. [Acedido em 05 October 2015].

[16] Parker Software, “Parker Software,” [Online]. Available: http://www.parkersoftware.com/. [Acedido em 06 October 2015].

[17] Netop, “Live Guide,” [Online]. Available: http://www.netop.com/live-guide/what-is-live-guide.htm. [Acedido em 05 October 2015].

[18] Netop, “Netop,” [Online]. Available: http://www.netop.com/. [Acedido em 06 October 2015].

[19] Website Alive, “Website Alive,” [Online]. Available: https://www.websitealive.com/. [Acedido em 06 October 2015].

[20] Google Inc., “Google Analytics,” [Online]. Available: https://www.google.com/analytics/. [Acedido em 16 October 2015].

[21] Adobe Systems, “Adobe Marketing Cloud,” [Online]. Available: http://www.adobe.com/marketing-cloud.html. [Acedido em 16 October 2015].

[22] Wikipedia, “Omniture,” [Online]. Available: https://en.wikipedia.org/wiki/Omniture. [Acedido em 16 October 2015].

[23] Kissmetrics, “Kissmetrics,” [Online]. Available: https://kissmetrics.com/. [Acedido em 16 October 2015].

[24] Wikipedia, “Customer Relationship Management,” [Online]. Available: https://pt.wikipedia.org/wiki/Customer_relationship_management. [Acedido em 16 2015 16].

[25] Forbes, 22 May 2015. [Online]. Available: http://www.forbes.com/sites/louiscolumbus/2015/05/22/gartner-crm-market-share-update-47-of-all-crm-systems-are-saas-based-salesforce-accelerates-lead/#2715e4857a0b515edaed4e6e. [Acedido em 22 October 2015].

[26] Salesforce, “Salesforce,” [Online]. Available: http://www.salesforce.com/eu/?ir=1. [Acedido em 22 Ocotber 2015].

[27] SAP, “SAP CRM,” [Online]. Available: http://go.sap.com/solution/customer-engagement.html. [Acedido em 2015 October 2015].

[28] Oracle, “Oracle CRM,” [Online]. Available: https://www.oracle.com/applications/customer-experience/crm/index.html. [Acedido em 2015 October 2015].

[29] Microsoft, “Microsoft Dynamics CRM,” [Online]. Available: https://www.microsoft.com/pt-pt/dynamics/default.aspx. [Acedido em 22 October 2015].

[30] Java, “Java,” [Online]. Available: https://www.java.com/pt_BR/. [Acedido em 2015 October 15].

[31] PHP, “PHP,” [Online]. Available: https://secure.php.net/. [Acedido em 2015 October 2015].

[32] nodejs.org, “NodeJS,” [Online]. Available: https://nodejs.org/en/. [Acedido em 28 October 2015].

[33] npm, Inc., “npm,” [Online]. Available: https://www.npmjs.com/. [Acedido em 2015 October 2015].

[34] socket.io, “Socket.io,” [Online]. Available: http://socket.io/. [Acedido em 28 October 2015].

[35] Express, “Express,” [Online]. Available: http://expressjs.com/. [Acedido em 28 October 2015].

[36] Socket.io, “Get Started: Chat application,” [Online]. Available: http://socket.io/get-started/chat/. [Acedido em 20 October 2015].

[37] Google .Inc, “AngularJS,” [Online]. Available: https://angularjs.org/. [Acedido em 15 November 2015].

[38] MongoDB, “MongoDB,” [Online]. Available: https://www.mongodb.org/. [Acedido em 23 October 2015].

[39] "WebRTC," [Online]. Available: https://webrtc.org/. [Accessed 14 November 2015].

[40] “webrtc Public Diagram for Website,” [Online]. Available: https://webrtc.org/assets/images/webrtc-public-diagram-for-website.png. [Acedido em 20 June 2016].

[41] W3 School, “Browser Statistics,” [Online]. Available: http://www.w3schools.com/browsers/browsers_stats.asp. [Acedido em 27 October 2015].

[42] “Can I use WebRTC?,” 12 May 2015. [Online]. Available: http://caniuse.com/#search=webrtc. [Acedido em 27 October 2015].

[43] “Is WebRTC ready yet?,” [Online]. Available: http://iswebrtcreadyyet.com/. [Acedido em 27 October 2015].

[44] Wikipedia, “Customer Relationship Management,” [Online]. Available: https://en.wikipedia.org/wiki/Customer_relationship_management. [Acedido em 2015 October 2015].

[45] Wikipedia, “Google Analytics,” [Online]. Available: https://en.wikipedia.org/wiki/Google_Analytics. [Acedido em 2015 October 2015].

[46] Wikipedia, “Live support software,” [Online]. Available: https://en.wikipedia.org/wiki/Live_support_software. [Acedido em 02 October 2015].

[47] Wikipedia, “NodeJS,” [Online]. Available: https://en.wikipedia.org/wiki/Node.js. [Acedido em 28 October 2015].

[48] Nasdq, “Salesforce.com Inc Stock Quote & Summary Data,” [Online]. Available: http://www.nasdaq.com/symbol/crm. [Acedido em 2015 October 2015].

[49] C. Buckler, “SQL vs NoSQL: The Differences,” 18 September 2015. [Online]. Available: http://www.sitepoint.com/sql-vs-nosql-differences/. [Acedido em 25 October 2015].

[50] S. Dutton, “Getting Started with WebRTC,” 23 July 2012. [Online]. Available: http://www.html5rocks.com/en/tutorials/webrtc/basics/. [Acedido em 30 October 2015].

[51] E. Mcnulty, “SQL vs NoSQL - What you need to know,” 1 July 2014. [Online]. Available: http://dataconomy.com/sql-vs-nosql-need-know/. [Acedido em 23 October 2015].

[52] InfoWorld, “PHP vs. Node.js: An epic battle for developer mind share,” [Online]. Available: http://www.infoworld.com/article/2866712/php/php-vs-node-js-an-epic-battle-for-developer-mind-share.html. [Acedido em 15 November 2015].

[53] InfoWorld, “Java vs. Node.js: An epic battle for developer mind share,” [Online]. Available: http://www.infoworld.com/article/2883328/java/java-vs-nodejs-an-epic-battle-for-developer-mindshare.html. [Acedido em 15 November 2015].

[54] A. Sergiienko, “WebRTC Example,” 9 August 2015. [Online]. Available: https://www.webrtcexample.com/blog/?go=all/which-audio-and-video-codecs-can-be-used-in-a-webrtc-application/. [Acedido em June 2016].

[55] WebRTC.org, “WebRTC FAQ,” [Online]. Available: https://webrtc.org/faq/#what-is-the-vp8-video-codec. [Acedido em June 2016].

6 Appendix

6.1 Full Competitors Comparative Table

	Livezilla	Tawk.to	Vivocha	Provide Support	Zopim	BoldChat	Live Help Now	Kayako	Olark	LiveChat	ClickDesk	Comm100	WhosOn	Netop Live Guide	website Alive
Dashboard (RF_DB)															
Web Dashboard	✓	●	✓	✓	✓	✗	✓	✗	✓	✓	✓	✓	✗	✓	●
Responsive dashboard	✗	●	✓	●	●	✗	●	✗	✗	●	✓	●	✗	●	●
Mobile app	✓	✓	✓	✗	✓	✓	✓	✓	●	✓	✓	✓	●	✓	✓
Busy state while on voice call	✗	✗	●	✗	✗	✗	✗	✗	✗	●	●	●	✓	●	●
IDLE state detection	✓	●	●	✓	●	●	●	●	●	●	●	●	✓	●	●
Proactive engagement	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Prioritize chats	✓	✗	●	●	●	✗	●	●	●	●	●	●	●	●	●
Widget (RF_WI)															
Page Tracking	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Referrer tracking	✓	●	✓	✓	●	✓	✓	✓	●	✓	✓	✓	✓	✓	✓
Browser tracking	✓	●	✓	✓	●	✓	✓	✓	●	●	✓	●	●	●	●
Geolocation tracking	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Real time monitoring	✓	✓	✓	✓	✓	✓	✓	✓	✓	●	✓	✓	✓	✓	✓
Online customers list	✓	✓	●	✓	✓	●	●	●	✓	●	✓	●	✓	●	●
Change on operators state change	●	●	●	✓	●	●	●	●	✓	●	●	●	●	●	●
CRM integration	●	●	✓	✗	✓	●	✓	●	✓	✓	✓	✓	✓	✓	✓
Social media integration	✗	✗	✓	✗	✗	✗	✓	✗	✗	✓	●	✓	●	●	●
Recognize customers	✓	✓	●	✗	✓	✓	✓	●	●	✓	●	✓	✓	●	✓
Cliente (RF_CL)															
Chat	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Chat withou new window	✗	✓	●	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✓
Offlive conversation	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓

VoIP withou plugins	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	●	✓	✗
VoIP with plugins	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	●	✗	✗
Screen sharing	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓
Operador (RF_OP)															
Operators login	●	✓	✓	●	✓	●	●	✓	●	●	●	✓	●	✓	●
Operators logout	●	✓	✓	●	✓	●	●	✓	●	●	●	✓	●	✓	●
URL push	✓	✗	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	●
Multiple chat windows	✗	●	✓	✓	✓	●	●	●	✓	✓	✓	✓	●	✓	✓
Chat transfer	✓	✓	✓	✓	✓	●	✓	✓	●	✓	✓	✓	✓	✗	✓
Change operator state	●	●	✓	✓	●	●	●	●	●	●	●	●	✓	●	●
canned responses	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	●
Proactive engagement rules	✓	✓	✓	✗	✓	✓	●	●	✓	●	●	✓	✓	✓	✓
Owner (RF_OW)															
Customizable widget	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Widget injection	✗	✗	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Operators tracking	✗	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Masters in Informatics Engineering

Internship 2015/2016
Final Report

Online context for voice communications

Annex B – Approach

José Manuel Marques Grilo

jgrilo@student.dei.uc.pt

Supervisors:

Jorge Sousa
Luís Matos

Carlos Bento

July, 1st 2016

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

Index

Index of Tables

Index of Figures

1 Introduction

The internship followed a software methodology already adopted by the company, which was ideal to the kind of project. As in any project, there are risks that may affect its outcome, it is important to identify them and create mitigation plans in order to minimize or eliminate the risks.

This chapter presents the approach followed; the planning process and the risks and mitigation plans.

2 Methodology

This section provides an overview on Scrum, the methodology used during the internship, its roles, events, artifacts and definition of done.

2.1 Scrum Process

According to figure, there are four kinds of projects:

- Simple: projects with low complexity, where we know both the set of requirements and technologies to use.
- Complicated: projects where there is still some degree of certainty about the requirements and technologies, but with a bit more complexity than the simple projects.
- Complex: projects with a great degree of complexity where there are many possible requirements that demand a huge amount of study in order to get them defined, and there are a huge amount of technologies that may be used.
- Anarchy: projects where there is neither a definition of the requirements or the technologies to be used.

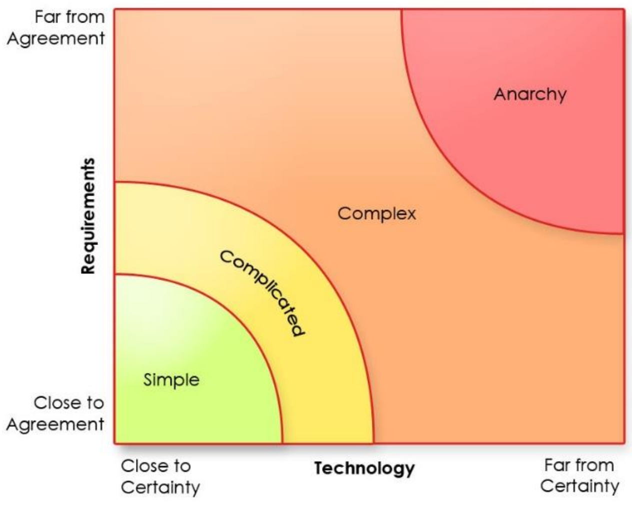


Figure 2.1. Projects complexity

[1]

The solution to the first two kinds of problems can easily be achieved with waterfall-like methodologies. The anarchy kind due to all the uncertainties are hardly named projects. The complex projects, which are the area where the internship problem inserts into, are problems where a fixed planning will not help due to its degree of uncertainty, but are feasible if an agile methodology is used.

"Scrum is a framework for developing and sustaining complex products. A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value" [1]. Scrum provides a flexible and holistic product development strategy whit the necessary agility for big projects where a wide set of requirements and technologies is available.

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

2.1.1 Scrum Roles

On essential component in Scrum is the work in teams. Teams in Scrum have three important roles:

- The **scrum master** ensures that the development process is moving forward and the values of Scrum stand while evaluates the scrum team's performance. Acts like a moderator who is responsible for creating a trustful environment to work in, facilitating team meetings, negotiating with product owner and removing impediments so that the **scrum team** can focus on development only.
- **Product owner** is the person responsible for maximize the value of the product developed. He manages the stakeholder's expectations, prioritizing the backlog and release planning. He acts like and man-in-the-middle, ensuring that the product developed by the scrum team is the one that the stakeholders are expecting.
- **Scrum team** is the set of people of different areas (Design, Development and Test) in charge of develop and deliver a potential releasable increment of the product at the end of each sprint. They must be self-organized and autonomous, in order to estimate the size of the requirements and making their own design and implementation decisions.

2.1.2 Scrum Events

Pre-established events are used in Scum in order to create a regular methodology and minimize the need of not defined meetings. All events are time-boxed with a maximum duration. Except sprint which has a fixed duration, all other events may end whenever intended to.

- **Sprint** is the container for all other events and the heart of Scrum. The sprint is used to develop a potentially releasable increment. It is a time-box with the fixed duration of one week to one month, and once the duration is fixed it cannot be shortened or lengthened.
- **Sprint planning** marks the beginning of each sprint and used to plan all the work that must be accomplished in the sprint in order to achieve a new increment. This planning is done in a meeting with no more than eight hours with the collaboration of all roles.
- **Daily scrum** is a stand-up meeting with fifteen minutes or less between the scum team members in order to reflect about the work done in the previous day, define a set of goals to achieve in the next day and see if any impediments could arise.
- **Sprint review and sprint retrospective** are prior meetings before sprint planning that occur at the end of each sprint. The sprint review is used by the scrum team and product owner to review the increment developed, the "done" and "not done" work and to present a demo to the stakeholders invited by the product owner.

2.1.3 Scrum Artifacts

The scrum artifacts represent work or value to provide opportunities for inspection and improvement in the methods used.

- **Product Backlog** is a living artifact that holds an ordered list of user stories that might be needed in the product as a source of requirements to develop. The product owner is the only responsible for the product backlog, including its content, availability and ordering, based on the scrum team analysis. The product backlog is never complete and it evolves alongside the project.
- The **Sprint Backlog** is a set of product backlog items selected at the sprint planning for the sprint. This subset is chosen based on the priority of each user story and in the time that each story takes to be "done". It is a forecast of what functionalities must be on the

next increment, in other words, represent the work that the scrum team must do in order to achieve the sprint goal.

- **Burndown Chart** is a displayed chart showing the remaining work in the sprint. It gives the sprint progress and must be updated every day. A burndown chart consists in an ideal burndown line over the sprint time and lines with the remaining tasks and effort until the end of the sprint that must be update every day.
- **Increment** is the sum of all sprint backlog items completed and the increments of previous sprints.

2.1.4 Definition of Done

"Done" is a subjective word to express the end or the accomplishment of something, so a shared understanding of what "done" means should be defined. This is a very import concept which allows the scrum team to assess when the work is complete on the product increment. Besides that, it also helps the team to select which sprint backlog items to choose in the sprint planning.

3 Planning

In the internship, supervisor Jorge Sousa has the scrum master role, being responsible to manage all meetings and solve possible impediments.

Tutor Luís Matos took the product owner role with product backlog management. One of the responsibilities of the product owner is the definition of the requirements, this task was assigned to the intern in order to let him learn the processes of scrum.

The scrum team is composed by the intern José Grilo.

The project planning, like in any other methodology, starts with the requirements definition, however in scrum a requirement has a different name and motivation.

3.1 Scope

Defining the project scope is one of the most critical steps in a project. Without knowing what you are supposed to be delivering at the end to the client and what the boundaries of the project are, there is a little chance for the project to success.

A poorly defined scope definition will lead to an impossible management during the project execution.

The main purpose of the scope is to clearly describe the boundaries of the project, according to the client's agreement. The elements within the scope and out of the scope must be well defined in order to clearly understand the area under the project control.

This section will be divided in two areas: The elements within the scope, the project objectives and its goals and the elements out of the scope.

3.1.1 Within the scope

Besides the elements that are included within the scope, this section will expose both internship and project goals and the project objective.

From the internship view, the goals are to consolidate knowledge about Software Engineering and gain experience in developing software in a corporate environment where commitment and team work are essential to produce a high quality piece of software. At a technical level the goal is to learn and master the use of web frameworks, such as Node.js and develop a stable and high quality software.

On the other hand, regarding the project, the main goal is to contribute with a proof of concept that can be used for conferences and exhibitions and a posteriorly development of a product. To accomplish this, the following features must be fully implemented and tested to guarantee the existence of zero bugs in the product:

- Injection of a widget: a code snippet must be provided and the client can copy and paste it in his website providing it functionalities such as chat, VoIP and real user tracking.
- Chat: allows both clients and operators to start a conversation between them.
- VoIP: allows the client to start a voice conversation with an operator without the need to install any plugins, using the WebRTC capabilities.
- Real User tracking: the widget must do a full track of the clients' path in real time, tracking his viewed pages and referrer, browser details and geolocation.
- Customers' analysis and browser control: allows the operators to see the clients' information and push pages to their browsers.
- Remote Assistant: WIT's demo of Remote Assistant must be integrated with the developed product as part of its functionalities. This feature was added on second semester in order to bring more value to the developed project. This was a calculated risk which led to small backlog, requirements and architecture changes. However competitors' analysis was not revisited due to time issues.

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

- Requirements analysis: identify, discuss and prioritize all the requirements.
- Technologies analysis**:** identify possible technologies to use, discuss the pros and cons and choose the one to use.
- Architecture definition: define how the features will work internally, understand which components must be created and how will they communicate with each other within the application.
- Risk analysis: identify possible risks and problems that may arise and trace a mitigation plan that overcomes them.

3.1.2 Out of the scope

This section will be specify the elements that were analyzed and are interesting to the project, but will not be implemented in this internship.

- Operators' profile management: allows the operator to manage his personal information such as name or password.
- Customize widget: allows the user to customize his widget appearance (colors, logo, *etc*) before the snippet generation.
- VoIP with Flash or plugins: allows the client to perform a VoIP call in browsers where WebRTC capabilities are not supported.

3.2 User Stories

A user story consists in one or more sentences written on small pieces of paper in everyday language that capture the intentions of the user that the program does as part as its job functions. It is used in scrum as the requirements definition because it is an easy and understandable way of handling the requirements without the formal formulation of a document. User stories follow the terminology bellow:

As a <role>, I want <goal/desire> so that <benefit>

Typically, the user stories are created and managed by the product owner, however, as already stated, the scrum team does the task as a learning process.

Usually, user stories are associated with a category, mainly if there are many features to implement. At the internship seven categories were created:

- **Widget**: consists in the development of features that allow the tracking of the client's information.
- **Widget's owner**: corresponds to the requirements of add and remove operators and widget's injection.
- **Client**: This category relates to all the functionalities a client can do while browsing in a page with the product
- **Dashboard**: consists in the development of the features that allow the operator to communicate, and retrieve all users' information from the server
- **Operator**: This category relates to all the functionalities an operator can do while using the dashboard
- **Server**: corresponds to the development of a server-side application that establishes all the communication with the dashboard, widget and database.
- **Documentation**: consists in the construction of all the documents related to the state of the art, requirements, architecture, demos and internship documentation.

3.3 Product Backlog

Each of the created user stories corresponds to an entry in the product backlog. Furthermore, each user story was assigned with a priority and its difficulty was calculated using planning poker. Planning poker is a gamification technique that consists in using a deck of cards where each card is assigned with a number from the Fibonacci sequence from 1 to 13 (100 and "?" are also used). It is assigned the value of "2" to make an easier user story, then, the other stories are valued based on the cards played by each participant relative to the first story ("?" means that a value cannot be assigned or more information is needed).

Based on the priorities and difficulties assigned an ordered list was formed. That list was the first version of the product backlog.

As a **<role>**	, I want **<goal/desire>**	so that **<benefit>** (opt)
As a site visitor	, I want to chat with an operator	in order to ask a question
As a site visitor	, I want to make a voice call with an operator	in order to clear my doubts quickly
As a site visitor	, I want to answer to a chat invitation	so that an operator can talk to me
As an operator	, I want to push a URL to the client	so that it redirects the client's browser to another page
As an operator	, I want to answer a voice call	in order to help my client
As an operator	, I want to answer a chat call	in order to help my client
As an operator	, I want to see my clients info (tracked pages, browser, geolocation, referrer, previous conversations)	so that I can give a customized assistance
As an operator	, I want to see the page's visitors list	

As an operator	, I want to logout from the dashboard	in order to finish my activities
As an operator	, I want to track my clients is real-time	
As an operator	, I want to login at the dashboard	so that I can start my activities
As an operator	, I want to take a note on my customer	So that other operators can see is doubts
As a widget's owner	, I want to copy a snippet in my pages	so that the widget starts to work
As a widget	, I want to open a connection to my server	so that all info can be delivered
As a widget	, I want to track the pages that my associated client visits	
As a widget	, I want to track the referrer from my client	
As a widget	, I want to track the geolocation from my client	
As a widget	, I want to track the browser that my client is using	
As a widget	, I want to recognize a repeat client	so that all the new tracked info is associated to him
As a widget	, I want to load the chat interface	so that a client can start a conversation with an operator
As a widget	, I want to reload a page when URL push command arrives	so that the client can see the page
As a dashboard	, I want to open a connection to my server	so that all info can be delivered
As a dashboard	, I want to start a pro-active chat with a client	
As a dashboard	, I want to ask for the clients' info	so that it can be shown to the operator
As a dashboard	, I want to filter contacts	So that the operator only see a portion of the contact list
As a server	, I want to maintain all the connections as long they are needed	so that widgets and dashboard can send their data
As a server	, I want to redirect all chat messages to their receivers	so that clients and operators can talk
As a server	, I want to change the client's associated operator	so that another operator can help the client
As a server	, I want to open a connection to a DB	so that all info can be persisted
As a server	, I want to persist all tracked info at DB	
As a server	, I want to read info from the DB	so that I can send it to the dashboards
As a server	, I want to log all the important events	so that a registry is maintained

Table 3.1. Initial product backlog

3.4 Sprints

After the project is minimally defined, it is time for the scrum team to work on the product. In an initial stage of the project was done, larger sprints (four weeks), then the sprints started to have a two week-duration.

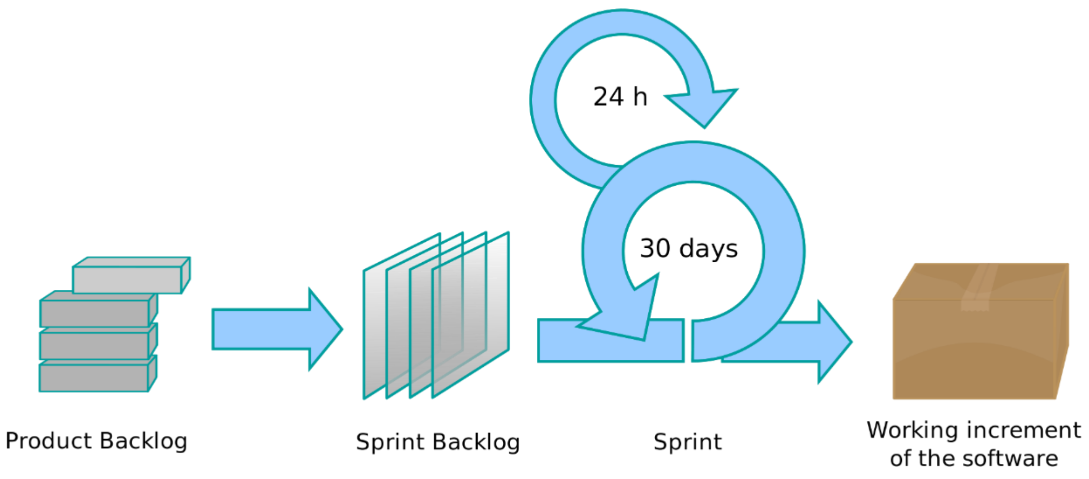


Figure 3.1. Scrum Process

[3]

Presented here is summary of the sprints so far.

- **Sprint #0**
  Sprint Planning: 14-10-2015
  This was the warm up sprint. It was assigned to the scrum team to do the competitors' analysis, technologies' analysis, formulate the user stories and the "Definition of Done".
- **Sprint #1**
  Sprint Planning: 17-11-2015
  In this sprint it was assigned to the scrum team to design the database structure and install and setup the database. It was also required to write the requirements document in order to start the UI design.
  Sprint review and retrospective: The intern felt some difficulties to manage the required tasks during the sprint.
- **Sprint #2**
  Sprint Planning: 01-12-2015
  This sprint marked the development's beginning. It was assigned to set the working environment (install dependencies and IDE) and create the connections between widget/back office and server.
  Sprint Backlog:
  - As a developer, I want to set the working environment.
  - As a server, I want to maintain all the connections as long they are needed so that widgets and dashboard can send their data.
  - As a widget, I want to open a connection to my server so that all info can be delivered.

  Sprint review and retrospective: It was supposed to start the UI implementation, however redlines were not ready.

- **Sprint #3**
  Sprint Planning: 16-12-2015
  In this sprint it was assigned to the team the operators' login and logout features and the UI implementation. The widget should also be loaded with the snippet.
  Sprint Backlog:
    - As a dashboard, I want to open a connection to my server so that all info can be delivered.
    - As widget's owner, I want to paste a snippet in my pages so that the widget starts to work.
    - As a guest, I want to login at the dashboard so that I can start my activities as operator.
    - As an operator, I want to logout from the dashboard in order to finish my activities.

  Sprint review and retrospective: Due to changes in UX, one feature became obsolete and was not implemented.
- **Sprint #4**
  Sprint Planning: 04-01-2016
  It was assigned to start the context collection features.
  Sprint Backlog:
    - As a widget, I want to track the pages that my associated client visits.
    - As a widget, I want to validate all forms in order to know my user.
    - As a widget, I want to track the referrer from my client.
    - As a widget, I want to track the browser that my client is using.
    - As a widget, I want to track my customer's OS information.
    - As an operator, I want to see my customers' tracked URL list.
- **Sprint #5**
  Sprint Planning: 18-01-2016
  This sprint marked the beginning of BO's development. Features regarding to queues loading and display and customers engaging were implemented. Midterm report was also written during this sprint.
  Sprint Backlog:
    - As an operator, I want to see the page's visitors list.
    - As an operator, I want to see my customers' info (tracked pages, browser and geolocation) so that I can give a customized assistance.
    - As an operator, I want to engage a customer, so that he appears in my 'requests accepted' queue.
    - As a widget, I want to track the geolocation from my client.
    - As an intern, I want to prepare the midterm presentation.
    - As an intern, I want to write the midterm report.
    - As a developer, I want to re-visit product backlog, in order to update user stories.
- **Sprint #6**
  Sprint Planning: 02-02-2016
  In this sprint filters on customers' queues were implemented. Operator's engaging feature was finished in this sprint, showing to the customer that he had been selected. Returning customers were also identified on this sprint. This last feature marked the finish of context collection feature set.
  Sprint Backlog:

    - As an operator, I want to select the customer, whose I want to see information.
    - As a widget, I want to get the operator associated photo and name to display to my customer.
    - As a server, I want to persist all tracked info at DB.
    - As a widget, I want to track the geolocation from my client with google API.
    - As a widget, I want to recognize a repeat client so that all the new tracked info is associated to him.
    - As a dashboard, I want filter my clients (recent, old, chat, voice).
- **Sprint #7**
  Sprint Planning: 16-02-2016
  This sprint was dedicated to chat functionalities such as messages exchanging and "is typing" notifications.
  Sprint Backlog:
    - As a server, I want to redirect all chat messages to their receivers so that clients and operators can talk.
    - As a site visitor, I want to chat with an operator in order to ask a question.
    - As a dashboard, I want to transfer clients who start a conversation from visitors list to chat list.
    - As a widget, I want to notify operators that a customer is typing.
    - As an operator, I want to chat with customers.
    - As a BO, I want to notify my customers that an operator is typing.
    - As a server, I want to read info from the DB so that I can send it to the dashboards.
- **Sprint #8**
  Sprint Planning: 01-03-2016
  In this sprint URL push was implemented alongside offline form request for customers.
  Sprint Backlog:
    - As a developer, I want to correct all my bugs.
    - As an operator, I want to push a URL to the client so that it redirects the client's browser to another page.
    - As a widget, I want to reload a page when URL push command arrivesso that the client can see the page.
    - As a BO, I want to display all the history of previous conversations.
    - As a customer, I want to send an offline to message in order to be contacted later.
    - As an operator, I want to close my open conversation.
- **Sprint #9**
  Sprint Planning: 21-03-2016
  This sprint served to implement and integrate Remote Assistant demonstration on OCVC project.
  Sprint Backlog:
    - As a developer, I want to study the remote assistance documentation, in order to know what is needed to do in the project.
    - As an intern I want it fix Sprint #8 bugs.
    - As a developer, I want to break Remote Assistant library into two libraries.
    - As a developer, I want to add Remote Assistant WebRTC libraries to my web app.

    - As an operator I want to answer a Remote Assistant webCall.
- **Sprint #10**
  Sprint Planning: 04-04-2016
  This sprint marked the beginning of VoIP calls implementation. Developed here was all the needed modules to make VoIP call for browsers.
  Sprint Backlog:
    - As intern I want to fix Sprint #9 bugs.
    - As a site visitor, I want to make a browser-to-browser voice call with an operator in order to clear my doubts quickly.
    - As an operator, I want to answer a browser-to-browser voice call in order to help my client.
- **Sprint #11**
  Sprint Planning: 19-04-2016
  In this sprint the product backlog was revisited and some flows were changed. Also in sprint mail sending features was implemented.
  Sprint Backlog:
    - As a widget, I want to send my customer's history via mail
    - As a customer, I want to ask for a browser-to-device voice call.
    - As intern, I want to fix Sprint #10 bugs.
    - As a developer, I want to rethink and redo events flow.
- **Sprint #12**
  Sprint Planning: 03-05-2016
  This sprint was used to implement VoIP calls for devices.
  Sprint Backlog:
    - As an operator, I want to answer to a browser-to-device voice call requested by customers.
    - As a server, I want to log all the important events so that a registry is maintained.
    - As intern, I want to fix Sprint #11 bugs.
    - As an operator, I want to delete my comments.
- **Sprint #13**
  Sprint Planning: 17-05-2016
  This sprint was used to test all the implemented features and bug solving. During this sprint some UI changes were taught and WIT designer Elizabeth Pereira started to work on the project again.
- **Sprint #14**
  Sprint Planning: 31-05-2016
  This sprint was mainly used to the final report writing.

3.5 Definition of Done

In order to have shared understanding of “done” a definition was created, both for user story creation and development. Besides that a definition of done to the increment was created too.

The creation of a new user story is stated as “done” if:

5. The user story follow the notation “As a <role>, I want <goal/desire> so that <benefit>”;
6. Story sized with thirteen story points or less;
7. Story is divided in tasks and each task has a duration in hours;
8. Story necessity is explained and agreed by all.

A user story development is stated as “done” if:

8. All the tasks related to development are coded;
9. Code commented and meeting company’s development standards;
10. Builds without errors;
11. Acceptance tests are written and passing;
12. Code committed on server;
13. Relevant documentation is produced or updated;
14. Remaining hours for story set to zero and story closed.

An increment is stated as “done” if:

6. All modules developed during the sprint are integrated with the previous release;
7. The increment build as no errors;
8. It is ready for demo;
9. Final version is updated to server;
10. A presentation is prepared to present the increment.

4 Risks and mitigation plans

There are always risks in every project. Each risk has a probability to contribute to the failure of the project. Identify the risks, its sources and create mitigation plans are the best actions to take. These strategies should be reviewed periodically because the risk probability can grow and new mitigation plan may be needed.

The risk identification is based in factors that can cause project failure and their analysis is based on the probability of occurrence presented on Table 4.1, the impact on project if the risk occurs presented on Table 4.2. It is also important to take into account when the possibility of the risk itself.

Percentage	**< 30%**	**30% - 50%**	**50% - 75%**	**> 75%**
Probability	Low	Medium	High	Very high

Table 4.1. Risk occurrence probability

Impact	**Description**
Low	Project success is not compromised
Medium	Project success is not compromised, however small adjustments are required so that risks do not evolve
High	Project success can be compromised if no adjustment and additional effort is done
Very High	Project success can be seriously compromised

Table 4.2. Risk associated impact

Occurrence forecast	**Description**
Short-term	Risks can occur in an initial project phase, during the first development weeks
Mid-term	Risks can occur in an intermediate project phase, during the development
Long-term	Risks can occur in a final project phase, after the development

Table 4.3. Risk occurrence forecast time

This is an up-to-date living list with the risks identified, their probabilities, impacts, occurrence forecasts, consequences and mitigation plans.

ID	**RK_01**
Name	Frameworks updates
Probability	Low
Impact	High
Occurrence forecast	Medium-term
Consequence	Deprecated code and/or system failures
Mitigation plan	Regular code reviews in order to look for deprecated code and bugs

Table 4.4. Risk 01 - Frameworks updates

ID	**RK_02**
Name	Unreachable frameworks' servers
Probability	Low
Impact	High
Occurrence forecast	Medium-term
Consequence	System failures
Mitigation plan	Frameworks should be ready to be served from internship server

Table 4.5. Risk 02 - Unreachable frameworks' servers

ID	**RK_03**
Name	Poorly defined requirements
Probability	Low
Impact	Medium
Occurrence forecast	Short-term
Consequence	Delay or failure in the requirements
Mitigation plan	Requirements documents should be reviewed and approved by scrum master

Table 4.6. Risk 03 - Poorly defined requirements

ID	RK_04
Name	Requirements’ changes
Probability	Medium
Impact	Medium
Occurrence forecast	Medium-term
Consequence	Delay or failure in the requirements
Mitigation plan	Stakeholders should be present regularly at sprint review to approve the work done so far

Table 4.7. Risk 04 - Requirements' changes

ID	RK_05
Name	Bad planning
Probability	Medium
Impact	Very High
Occurrence forecast	Medium-term
Consequence	Delay or failure in the requirements
Mitigation plan	Standup meetings (daily scrum) should be performed every day in order to see the current difficulties

Table 4.8. Risk 05 - Bad planning

ID	RK_06
Name	Not meeting stakeholders’ expectations
Probability	Low
Impact	Very High
Occurrence forecast	Long-term
Consequence	Delay or failure in the requirements
Mitigation plan	Stakeholders should be present regularly at sprint review to approve the work done so far

Table 4.9. Risk 06 - Not meeting stakeholders' expectations

ID	**RK_07**
Name	Technologies learning curve
Probability	Medium
Impact	High
Occurrence forecast	Short-term
Consequence	Delay or failure in the requirements
Mitigation plan	At planning execution time should be allocated by taking into account the time needed to learn the technologies to use

Table 4.10. Risk 07 - Technologies learning curve

ID	**RK_08**
Name	Strong market competition
Probability	High
Impact	Medium
Occurrence forecast	Long-term
Consequence	Project failure
Mitigation plan	Constant market analysis in order to know the competitors and how can the internship product differentiates from them

Table 4.11. Risk 08 - Strong market competition

To prioritize the identified risks and know which are the most urgent to fix was used Pareto's Top N strategy [34] based on the occurrence, impact and occurrence forecast of each risk.

The urgency level of each risk was determined with the analysis of Table 4.12.

		Impact			
		Low	**Medium**	**High**	**Very High**
Probability	**Very High**	🟨	🟧	🟧	🟥
	High	🟩	🟨 RK_08	🟧	🟧
	Medium	🟩	🟩 RK_04	🟨 RK_07	🟧 RK_05
	Low	🟩	🟩 RK_03	🟩 RK_01, RK_02	🟨 RK_06

Table 4.12. Risk exposure

Color	🟩	🟨	🟧	🟥
Risk Exposure	Low	Medium	High	Very High

Table 4.13. Table 4.12'scolor code

From the table below it is possible to analyze the risk exposure to each risk. The green risks are the ones that will not jeopardize the project either because they have a low probability or a low impact. On the other hand, red risks are the most dangerous and will cause the project failure.

To prioritize the risk list it was taken into account both risk exposure and occurrence forecast to each risk.

Risk ID	Risk Exposure	Occurrence forecast
RK_05	High	Medium-term
RK_06	Medium	Long-term
RK_07	Medium	Short-term
RK_08	Medium	Long-term
RK_04	Low	Medium-term
RK_01	Low	Medium-term
RK_02	Low	Medium-term
RK_03	Low	Short-term

Table 4.14. Risk prioritization based on exposure and occurrence forecast

Lastly, it is important to know which risks are important to mitigate. Those, are found based with their probability and impact. The risks that are important to mitigate are those who have a high or very high probability and a high or very high impact, which means that only RK_05 needs to mitigate.

5 References

[1] Noop, “Simple vs. Complicated vs. Complex vs. Chaotic,” 20 August 2008. [Online]. Available: http://noop.nl/2008/08/simple-vs-complicated-vs-complex-vs-chaotic.html. [Acedido em 12 October 2015].

[2] “Scrum Guide,” 2014. [Online]. Available: http://www.scrumguides.org/scrum-guide.html. [Acedido em 12 October 2015].

[3] [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/thumb/5/58/Scrum_process.svg/2000px-Scrum_process.svg.png. [Acedido em 12 October 2015].

[4] J. C. Helm, “Risk Management - Methods ans Tools,” 1 January 2006. [Online]. Available: http://sce.uhcl.edu/helm/BB-TestRiskMan/my_files/module6/topn.htm. [Acedido em 14 November 2015].

[5] M. Vieira, *Project Management, Module 3: Planning & Tracking - Risk Management,* October, 2014.

6 Apendix

6.1 Requirements

This annex will describe the formal requirements, functional and non-functional, divided to each actor.

The requirements' priority will be classified as P1 – Must have, P2 – Should have, P3- Nice to Have.

6.1.1 Functional Requirements

6.1.1.1 Customer

ID	**FR_CU_01**
Name	Expand chat window
Requirement	The widget icon will expand to a chat window. The expanded window should display FR_CU_02 form by default or display FR_CU_04 form if there are no operators online.
Priority	P1
Dependencies	None

ID	**FR_CU_02**
Name	Pre-chat form
Requirement	The user must fill a form with some information before start a chat conversation with an operator. The form fields are: • Name (required) • Email (optional)
Priority	P1
Dependencies	FR_CU_01

ID	**FR_CU_03**
Name	Chat
Requirement	The user must be able to have a chat conversation with an online operator, through the chat window, after submitting **FR_CU_02** form.
Priority	P1
Dependencies	FR_CU_02

ID	**FR_CU_04**
Name	Offline contact

Requirement	The user must fill a form with some information to send a message/or receive a phone call on his device when every operators are offline. The form fields for messaging are: • Name (required) • Email (required) • Message (required) The form fields for the call are: • Name (required) • Phone number (required) • Message (required)
Priority	P2
Dependencies	FR_CU_01

ID	**FR_CU_05**
Name	Start a voice call
Requirement	A new browser window must open if the user click to start a new voice call with an operator in the chat window.
Priority	P1
Dependencies	FR_CU_04

ID	**FR_CU_06**
Name	Voice call
Requirement	The user must be able to have a voice call with the operator through WebRTC.
Priority	P1
Dependencies	FR_CU_05

ID	**FR_CU_07**
Name	Finish Voice call
Requirement	The user must be able to finish the started voice. Only customers can end the call.
Priority	P1
Dependencies	FR_CU_06

ID	**FR_CU_08**
Name	Mute microphone

Requirement	The user must be able to mute his microphone during the voice call.
Priority	P2
Dependencies	FR_CU_06

ID	**FR_CU_09**
Name	Change volume level
Requirement	The user must be able to lower and rise the volume level.
Priority	P2
Dependencies	FR_CU_06

6.1.1.2 Operator

ID	**FR_OP_01**
Name	User login
Requirement	Users must be able to authenticate in the system through a username-password form. If the authentication is successful the operator state must be changed to online and he shall be sent to the dashboard page.
Priority	P1
Dependencies	None

ID	**FR_OP_02**
Name	User logout
Requirement	User must be able to leave the dashboard and logout his account. This must change the operator state back to offline and display to him the login's page.
Priority	P1
Dependencies	FR_OP_01

ID	**FR_OP_03**
Name	Online customers list
Requirement	Operators must be able to see all the online customers. This list must be sliced in three parts: 1. Customers with conversations started 2. Customers online 3. History

Priority	P1
Dependencies	FR_OP_01

ID	**FR_OP_04**
Name	Select user
Requirement	The operator must be able to select a user from the customers list. After selecting a user his information must be displayed at the dashboard. The information displayed should be: • Geolocation; • Referrer; • Browser information; • OS information; • List of tracked pages; • Chat history (last 20 messages). If the selected user (list 2.) already started a conversation with other operator the chat option must be disabled.
Priority	P1
Dependencies	FR_OP_01; FR_OP_03

ID	**FR_OP_05**
Name	Chat
Requirement	The operator must be able to have a chat conversation with the selected user (list 1. and 2.) if he has no conversations started with another operator.
Priority	P1
Dependencies	FR_OP_01; FR_OP_04

ID	**FR_OP_06**
Name	Take note
Requirement	The operator must be able to take notes on the selected customer profile.
Priority	P1
Dependencies	FR_OP_01; FR_OP_04

ID	**FR_OP_07**
Name	Accept browser-to-browser voice call

Requirement	A new browser window must open if the operator click to accept a new voice call invitation with a customer. No more than a browser-to-browser or browser-to-device call is allowed at once.
Priority	P1
Dependencies	FR_OP_01

ID	**FR_OP_08**
Name	Accept a browser-to-device voice call
Requirement	A new browser window must open if the operator click to accept a request to a voice call to the customer's device. No more than a browser-to-browser and browser-to-device call is allowed at once.
Priority	P3
Dependencies	FR_OP_01

ID	**FR_OP_09**
Name	Voice call
Requirement	The user must be able to have a voice call with the customer through WebRTC.
Priority	P1
Dependencies	FR_OP_01; FR_OP_07

ID	**FR_OP_10**
Name	Mute microphone
Requirement	The user must be able to mute his microphone during the voice call.
Priority	P2
Dependencies	FR_OP_01; FR_OP_07; **FR_OP_08**

ID	**FR_OP_11**
Name	Change volume level
Requirement	The user must be able to lower and rise the volume level.
Priority	P2
Dependencies	FR_OP_01; FR_OP_07; **FR_OP_08**

ID	**FR_OP_12**
Name	URL Push
Requirement	The operator must be able to send commands to the selected user. This commands should reload customer’s page browser with the URL sent.
Priority	P1
Dependencies	FR_OP_01; FR_OP_05

6.1.1.3 Widget’s Owner

ID	**FR_WO_01**
Name	Widget Injection
Requirement	The widget must be injected in the website through a snippet without the need of software installation.
Priority	P1
Dependencies	None

6.1.1.4 Widget

ID	**FR_WI_01**
Name	Graphical collapsed window
Requirement	The widget must load a collapsed window. Once clicked, the window should do **FR_CU_01’s actions.**
Priority	P1
Dependencies	**FR_WO_01**

ID	**FR_WI_02**
Name	Establish connection
Requirement	Once the graphical window loads a new connection (Comet/Long polling) must be established with the server. This connection will be used to send and receive both tracked data and chat messages.
Priority	P1
Dependencies	**FR_WI_01**

ID	**FR_WI_03**
Name	Recognize customer
Requirement	If a cookie was set previously the widget must recognize the customer as a returning one and send the cookie to the server.
Priority	P2

Dependencies	FR_WI_01

ID	**FR_WI_04**
Name	Set cookie
Requirement	If no cookie is set the widget must set a new cookie with a unique identifier generated at the server.
Priority	P2
Dependencies	FR_WI_01

ID	**FR_WI_05**
Name	Get customer’s personal information
Requirement	If a returning customer is recognized his information from previous conversations should be retrieved. If he expands the widget the form must be already pre-filled with: • Name • Email • Phone’s number If the user change any field should be assumed that the customer is not a returning one, and FR_WI_04’s actions should be performed
Priority	P3
Dependencies	FR_WI_03

ID	**FR_WI_06**
Name	Get customer’s chat history
Requirement	If a returning customer is recognized his chat history from previous conversations should be retrieved and displayed at the chat window once a new conversation starts. Only the last 20 messages should be retrieved.
Priority	P3
Dependencies	FR_WI_03

ID	**FR_WI_07**
Name	Get more customer’s chat history
Requirement	Every time the customer scrolls to the top of the conversation 20 more messages should be retrieved from the server.
Priority	P3

Dependencies	FR_WI_03; FR_WI_06

ID	**FR_WI_08**
Name	Page tracking
Requirement	The widget should track the list of pages visited by the customer. Every time the widget loads it must track the page where it is at the moment and send it to server. Ajax loads should not be considered.
Priority	P1
Dependencies	FR_WI_01

ID	**FR_WI_09**
Name	Referrer tracking
Requirement	The widget should track the referrer to the website used by the customer. In the first time the widget loads it must track the referrer that send the customer to the website.
Priority	P1
Dependencies	FR_WI_01

ID	**FR_WI_10**
Name	Geolocation tracking
Requirement	The widget should track the geolocation of the customer. In the first time the widget loads it must track geolocation of the user.
Priority	P1
Dependencies	FR_WI_01

ID	**FR_WI_11**
Name	Browser information tracking
Requirement	The widget should track the browser that the customer is using. In the first time the widget loads it must track both browser’s name and version.
Priority	P2
Dependencies	FR_WI_01

ID	**FR_WI_12**

Name	OS information tracking
Requirement	The widget should track the OS that the customer is using. In the first time the widget loads it must track OS's name.
Priority	P2
Dependencies	FR_WI_01

ID	**FR_WI_13**
Name	Send tracked information
Requirement	The tracked information from FR_WI_08 to FR_WI_10 should all be tracked in the first time the widget loads only. After all the information is tracked it must be sent to the server.
Priority	P1
Dependencies	FR_WI_01; FR_WI_08; FR_WI_09; FR_WI_10

ID	**FR_WI_14**
Name	Real time monitoring
Requirement	The tracked information from FR_WI_07 should be tracked every time the widget loads. Every time a new page is tracked it must be sent to the server.
Priority	P1
Dependencies	FR_WI_01; FR_WI_07

ID	**FR_WI_15**
Name	Close chat connection
Requirement	If the user closes the graphical chat window the customer must be set as offline to the operators (remove him from the online customers list), however the server connection should stand and the tracking should continue.
Priority	P1
Dependencies	FR_WI_01

6.1.1.5 Dashboard

ID	**FR_DB_01**
Name	Prioritize customers list
Requirement	The customers list should be prioritized by last activity from the user, i.e., users with the oldest unread messages should be on the top off the customers list 1.

	Others lists (list 1. and 2.) can be left as how they were received.
Priority	P2
Dependencies	**FR_OP_03**

ID	**FR_DB_02**
Name	Notify operator on message
Requirement	The dashboard should notify the operator on a new message arrival. The number of unread messages should be displayed side by side with the customer's name on the customers list as badge, until a maximum of 9 unread messages. After that maximum, the badge should be 9+.
Priority	P3
Dependencies	None

ID	**FR_DB_03**
Name	Notify operator on voice call
Requirement	The dashboard should notify the operator on a new voice call arrival. A phone call badge should appear side by side with the customer's name on the customers list and sound notification should play.
Priority	P3
Dependencies	None

ID	**FR_DB_04**
Name	Get more customer's chat history
Requirement	Every time the operator scrolls to the top of the selected customer's conversation 20 more messages should be retrieved from the server.
Priority	P3
Dependencies	**FR_OP_04**

6.1.1.6 Server

ID	**FR_SE_01**
Name	Accept connections
Requirement	The server must accept new connections from authenticated widgets and dashboards.
Priority	P1

Dependencies	None

ID	**FR_SE_02**
Name	Calculate customer’s ID
Requirement	A new unique identifier should be calculated and sent to every non-returning customer.
Priority	P1
Dependencies	None

ID	**FR_SE_03**
Name	Associate customer with operator
Requirement	Every customer should be associated with an operator based on operator’s customers queue size.
Priority	P2
Dependencies	None

ID	**FR_SE_04**
Name	Change associated operator to a customer
Requirement	When a chat transfer is requested the server should change the operator associated with a customer.
Priority	P2
Dependencies	None

ID	**FR_SE_05**
Name	Transfer information
Requirement	Every time a new message arrives to the server it should be forwarded to the respective owner.
Priority	P1
Dependencies	None

ID	**FR_SE_06**
Name	Persist tracked data at the database

Requirement	Every time new tracked data arrives from the widget it should be persisted at the database.
Priority	P1
Dependencies	None

ID	**FR_SE_07**
Name	Persist messages
Requirement	Every time new messages arrives from both widgets and dashboards they should be persisted at the database.
Priority	P1
Dependencies	None

ID	**FR_SE_08**
Name	Retrieve tracked data from the database
Requirement	Once asked, all information about a customer should be retrieved and sent to the operator.
Priority	P1
Dependencies	None

ID	**FR_SE_09**
Name	Retrieve messages from the database
Requirement	Once asked, the last 20 messages from a customer conversation should be retrieved and sent to the operator or operator.
Priority	P1
Dependencies	None

6.1.2 Non-Functional Requirements

6.1.2.1 Customer

ID	**NFR_OP_01**
Name	Geolocation obfuscation
Requirement	At the geolocation tracking moment, geolocation coordinates should be obfuscated before send them to the server.
Priority	P1
Dependencies	None

6.1.2.2 Operator

ID	**NFR_OP_01**
Name	Credentials encryption
Requirement	At the authentication moment, credentials should be encrypted before send them to the server.
Priority	P1
Dependencies	None

6.1.2.3 Dashboard

ID	**NFR_DB_01**
Name	Responsive web dashboard
Requirement	The operators' dashboard must be responsive in order to adapt the layout to the viewing environment.
Priority	P1
Dependencies	None

6.1.2.4 Server

ID	**NFR_SE_01**
Name	OS independence
Requirement	The server should be OS independent.
Priority	P1
Dependencies	None

ID	**NFR_SE_02**
Name	DB independence
Requirement	The server should be DB independent.
Priority	P1
Dependencies	None

ID	**NFR_SE_03**
Name	Data security
Requirement	Both 4.2.1 and 4.2.2 must be served via HTTPS.

Priority	P1
Dependencies	None

Masters in Informatics Engineering

Internship 2015/2016
Final Report

Online context for voice communications

Annex C – Architecture

José Manuel Marques Grilo

jgrilo@student.dei.uc.pt

Supervisors:

Jorge Sousa
Luís Matos

Carlos Bento

July, 1st 2016

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

Index

Index of Tables

Index of Figures

1 Introduction

The architecture is an essential component of any software project. It defines the structure and behavior of the components, how they interact between themselves, while helps to hold the non-functional requirements and defines a guideline to implementation.

The current annex is divided into three chapters:

- **System architecture**: the architecture is designed and justified;
- **Data model**: with the document references for the documents presented at the data storage;
- **Interfaces**: where the interfaces used for communication between modules is documented.

2 System Architecture

The model used to represent the system architecture was an adaptation of the C4 Model by Simon Brown [1]. The C4 Model [2] divides the architecture in four views:

- System Context: The system plus users and system dependencies
- Containers: The overall shape of the architecture and technology choices
- Components: Logical components and their interactions within a container
- Classes: Component or pattern implementation details.

This architecture representation was chosen mainly because it was concluded to be complete enough to present the system to all involved clearly. Another reason was the limitations given by the technologies, it would be very hard to create an architecture representation using UML diagrams while using Javascript language mainly.

C4 Model allow the creation of four levels of representation, each level could be used to present the architecture depending on the interested. Besides that, C4 uses a simple ubiquitous language that everyone can understand easily.

The first and second levels show who will use the product and how its components are used to construct it. These levels are the most abstract and easily give the overall shape and context of the product. They can be used to present the product to newly arrived stakeholders or potential interested customers.

Third and fourth levels are more technical and could be used by development team as a development guideline to ensure non-functional requirements. Here are presented the components inside each container, their interactions and the implementation patterns used.

2.1 System Context

System context is the highest level of abstraction and represents something that delivers value to somebody. A system is made up of a number of separate containers.

The system context is meant to answer three questions:

- What are we building?
- Who is using it?
- How does it fit into the existing environment?

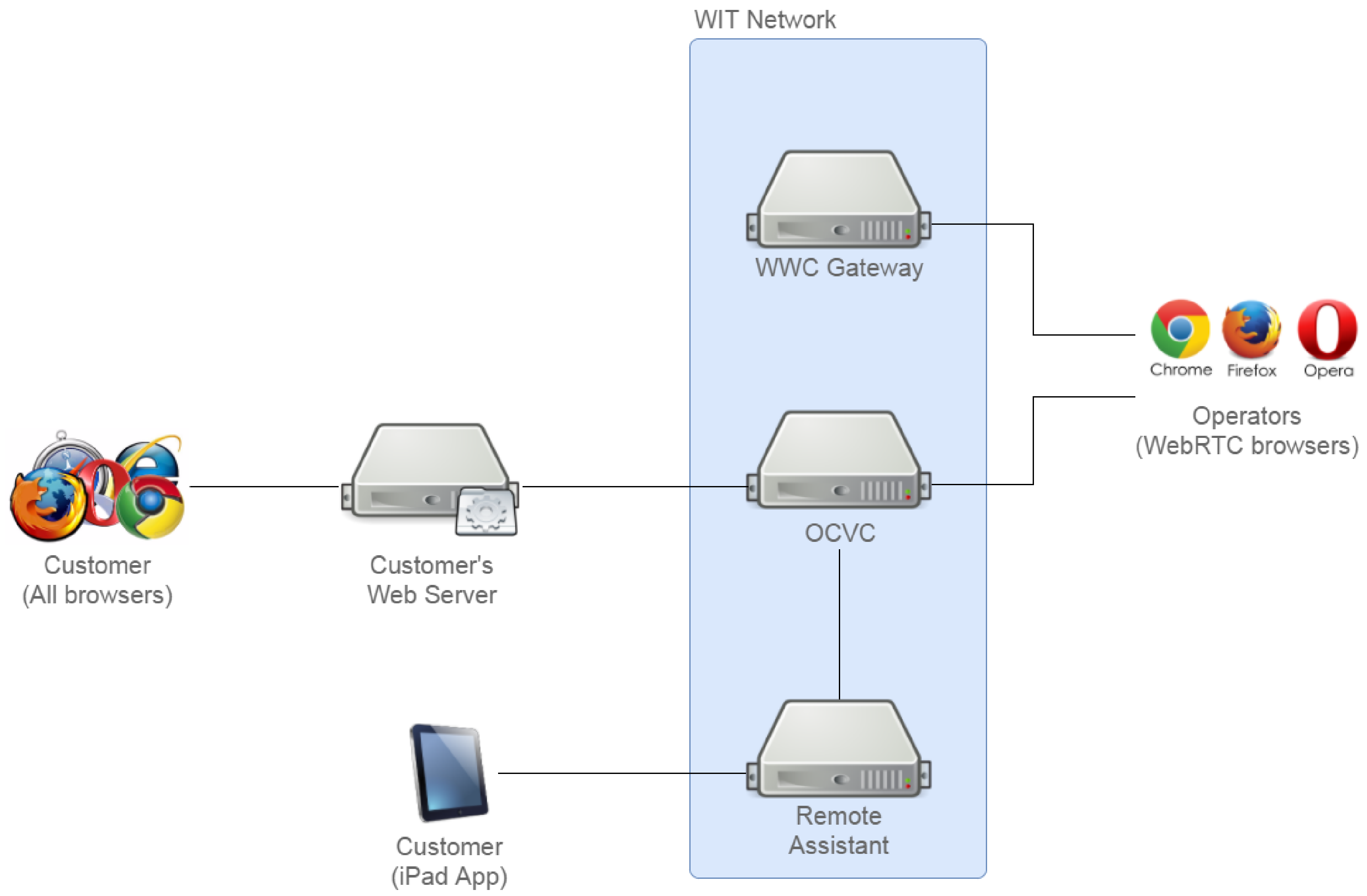


Figure 2. System context

With the previous diagram it is clear the kind of system that will be build and the actors who will use it.

This kind of view is suited for non-technical audience. It is clear to the ones who see this view that the system, will be used, on a first-level by operators. On a second level, customers will interact with a website that connect with the system or with an iPad App connected to OCVC server by Remote Assistant server.

The OCVC system, will serve a widget to customer’s website server. This widget will make the context collection and connection with the web server. On the other hand, it will serve a web application as well in order to operators to manage customers.

Regarding to actors there are two groups. One group of actors are the operators, they will interact with the system via a web application, pushing and pulling information from it that will enable them to see the customers that are using the websites, their tracked info and communicate with them. The other group of actors is the customers, which can be broken in two categories: web customers and iPad Customers. Web customers will not interact directly with the system. Instead, they will interact with the websites that get the widget from the system. The widget will track customers’ information and allow them to communicate with operators. iPad customers interact with WIT’s Remote Assistant application which allow them to call to operators at back office.

OCVC server is deployed in WIT’s Network, alongside WWC Gateway and Remote Assistant server, which act as project’s dependencies to implement breakout and iPad call respectively. Widget will be injected in customer’s website.

2.2 Containers

If we lower one level in the C4 Model we are looking for:

- What are the high level technology decisions?
- How do containers communicate with one another?
- As developer, where do I need to write code?

A container represents something in which components are executed or where data resides. This could be anything from a web or application server through to a rich client application or database. Containers are typically executables that are started as a part of the overall system, but they don’t have to be separate processes in their own right.

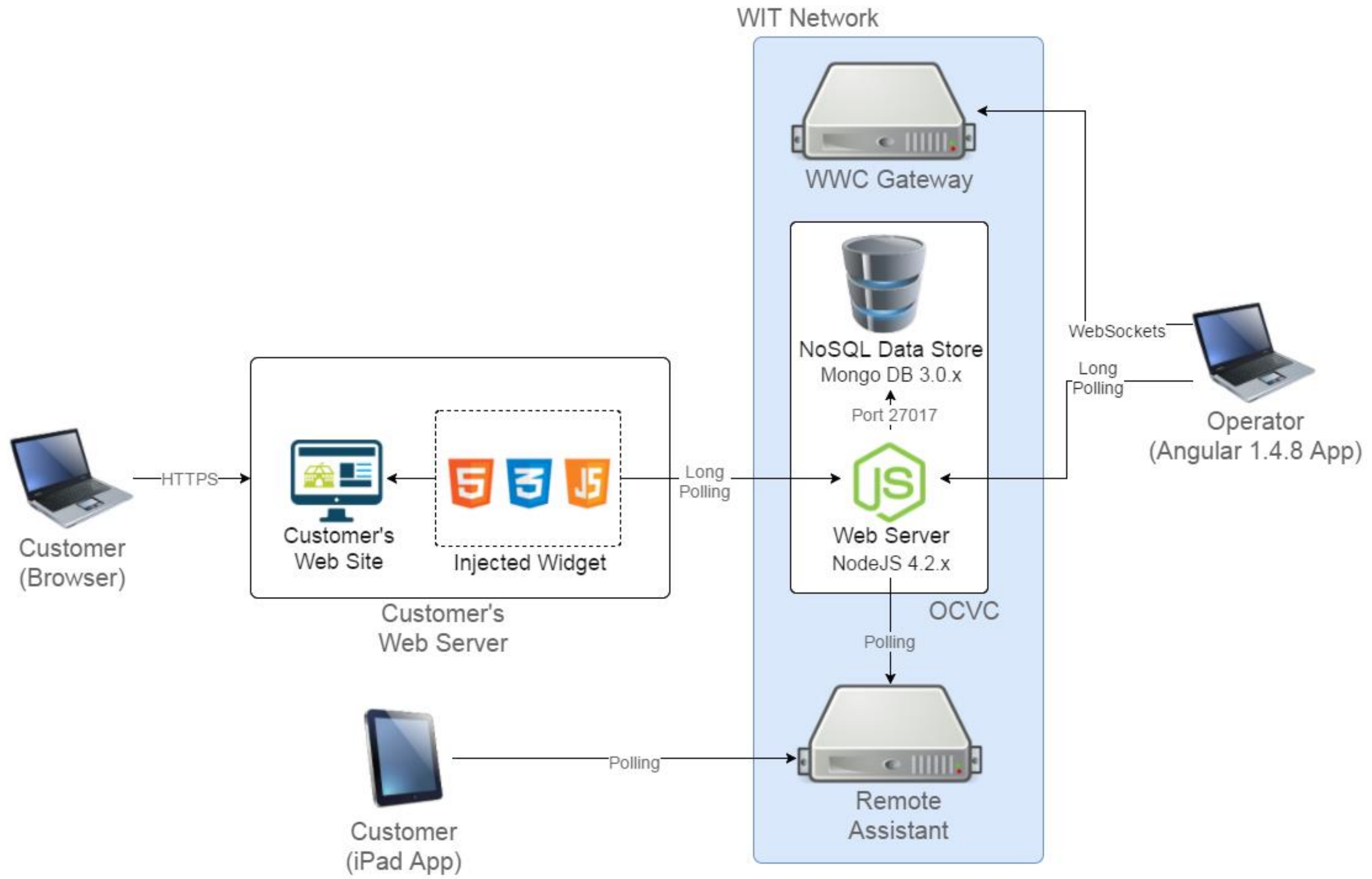


Figure 3. System containers

This view is suited for people that are semi-technical. Here they can see technologies choices, what containers will compose the system and how will they communicate between themselves.

At the technologies choices, the system will hold a Node.js server and a MongoDB.

The communication with the system will be done via HTTPS using self-signed certificates. Inside the system the web server will communicate with the data store that will run at the port 27017.

After UI is served communications proceed by Long Polling between Widget/Web App and OCVC server. To communicate between OCVC server and Remote Assistant is used Polling. This communication method was not changed due to the complexity to change an existent server while there were another priority features to develop. Between Web App and WWC gateway communication is made by WebSockets.

The web server will hold all the logic. The web server will take the request to widgets and response with the desired one, and will take requests from operators to access and manage the back office application. Besides that, the web application will also write and read from the MongoDB, that will store operators and customers' information, tracked data and sent messages.

User Interfaces, customers and operators, will both use HTML, Cascading Style Sheets (CSS) and JS. However selected frameworks will be different. Will use jQuery on Widget, Socket.io and Bootstrap. These frameworks were chosen because it was to inject them at Customer's web pages without breaking them and would fast development process. On operator's back office, Bootstrap, AngularJS and Socket.io as working frameworks were mainly used.

2.3 Components

A component can be thought of as a logical grouping of one or more classes. Components are typically made up of a number of collaborating classes, all sitting behind a higher-level contract.

At components level we need to answer to:

- What components/services is the container made up off?
- Are the technology choices and responsibilities clear?

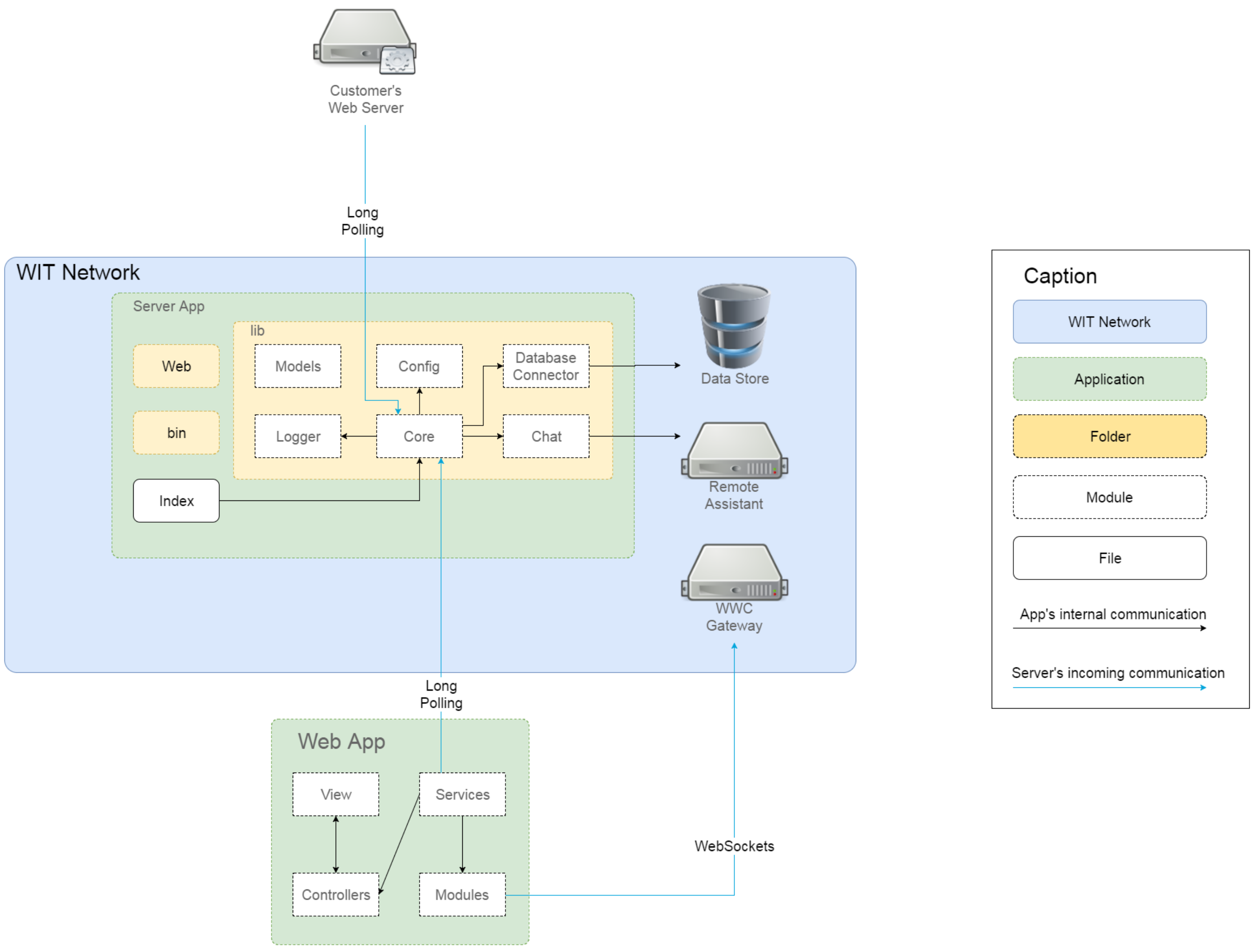


Figure 4. Components

As presented in Figure 3. there are 3 major containers being developed.

First there is a widget that will be injected on customers' web pages. Widget has 4 big components: Starter component, chat component, Context Collector UI constructor. Starter component is responsible for loading dependencies such as HTML, CSS and JS modules from server, Socket.io module and jQuery, Chat component is responsible for maintain communication with server, Context Collector as it states collects navigation history from the customer and UI constructor holds functions to construct UI as it is needed.

The biggest developed container is the OCVC server. OCVC server has 3 main folders: web, bin and lib. Web folder holds web components both for widget and BO applications, while bin folder holds installation scripts. Lib is the main component of the server. It has 6 modules, each with its own responsibility. Core module is started by index.js at root and it is responsible for starting the remaining modules. It also accepts HTTPS request requesting HTML, CSS and JS files. Chat module is responsible for serving widget and BO information request. Database connector is an abstraction layer between OCVC server and database with the purpose to manage communication between them. Logger is responsible to log all important events. Config holds configuration files needed to run the application. Finally, Models module hold the object models used in hte application.

Last container is the BO web app. This container has four components: view holding the HTML files and CSS to create the UI, controller who are responsible to manage UI aspect, modules used through the application and Services responsible for data and communication management.

2.4 Classes

For most of us in an OO world, classes are the smallest building blocks of our software systems.

This is an optional level with two purposes:

- Detail big components.
- Describe any particular pattern.

Since JS is not an OO language UML will not be used. However there is a need to draw this level in order to describe in more detail OCVC server (big component with a component-based development [3]) and BO web app (particular pattern).

OCVC Server

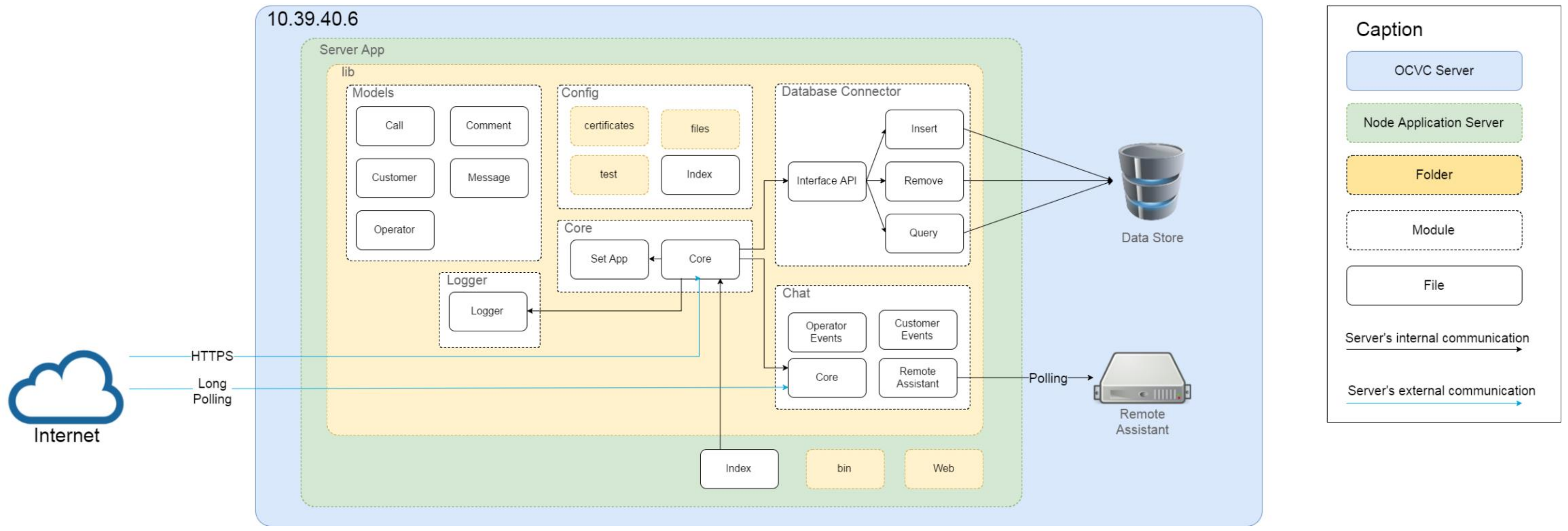


Figure 5. Server classes

OCVC Server’s Index at root is the file that starts the server, however all logic is under lib folder.

As stated before, lib has six modules: Core, Chat, Database connector, Config, Logger and Models. Server’s CBD help to keep responsibilities strict and changes on code easy to maintain, this means that the components are loosely coupled. This is particularly helpful for module maintenance as long as the communication standards do not change. The introducing of new features inside each module or new modules with new responsibilities is also a very easy operation. This architecture pattern was chosen because it kept the system easy to maintain, easy to improve and due to the communication type with customers (Event-Driven [4]).

Core module is the starter. Its main file starts all the remaining modules with the configurations presented at Config folder. It also processes HTTPS requests to serve HTML, CSS and JS files both to widget and BO.

Chat module is one of the most important server modules. It receives the long polling requests and serve the answer to both widget and BO. This module has a main file, Index is where Socket.IO requests arrive and responses leave and where all the functionalities are registered. Other three files have the implementation. So, a Socket.IO request arrive and is redirected to the file holding the implementation needed. After a response is produced it is then sent in Index.

Database connector classes can also be divided into two categories: Interface and queries. Interface class is the Interface API, this class is an abstraction class called by the other components in order to store, read, update or delete elements from the Data Base. The remaining files have the queries implementation calls.

Config file holds configuration files, certificates and tests.

Logger is the module responsible for log the important events that occur on server.

Models module holds the JSON structure for the objects used on server for the remaining modules.

Web App

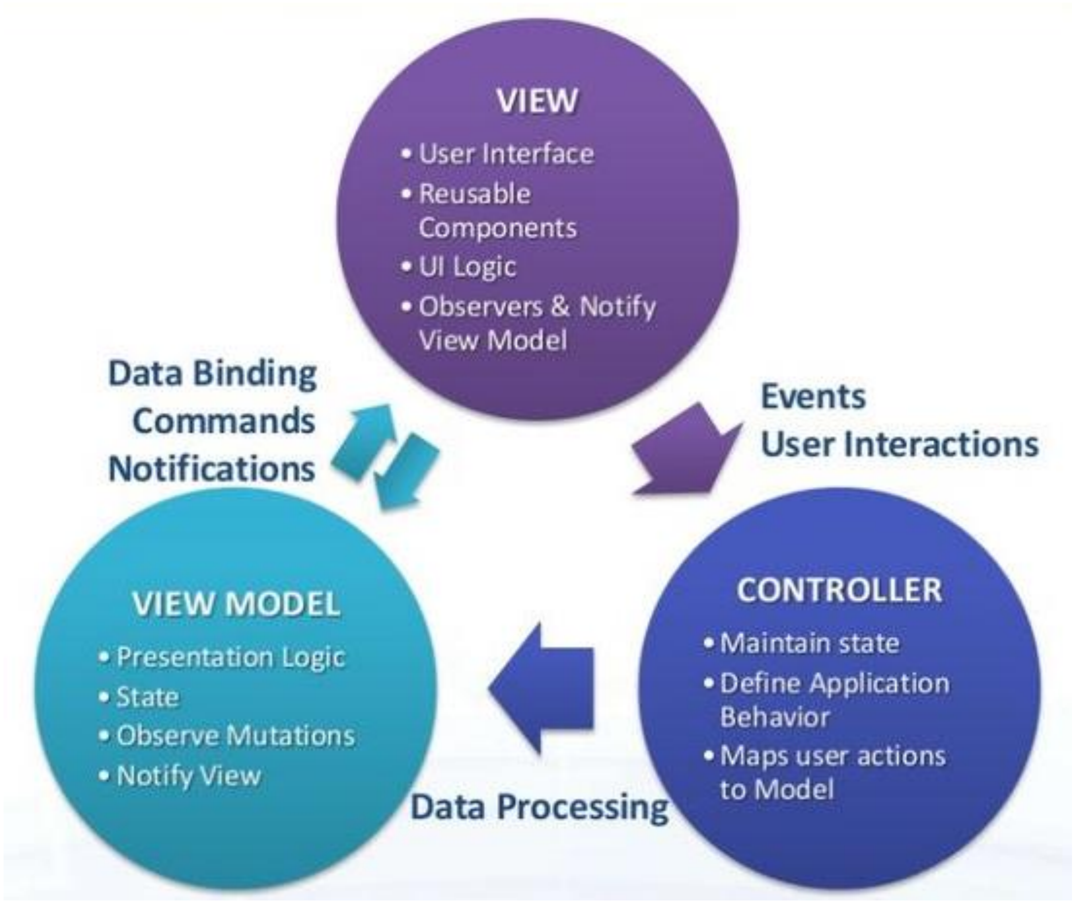


Figure 6. Angular MVVM model

The frontend application uses AngularJS framework MVVM.

Model (M) component acts as data access layer. Services presented in model are singletons always in execution.

The view (V) is the application's UI and includes several partial HTML and CSS files that compose the frontend.

The controller (VM) is an abstraction of the view exposing properties and commands. The controllers will register the callbacks that they expect at the service.

View and ViewModel have a bidirectional data binder, that allow changes in the model to propagate to the matching views, and changes made in the views are also propagated to the model. When the application data changes, so does the UI, and vice-versa.

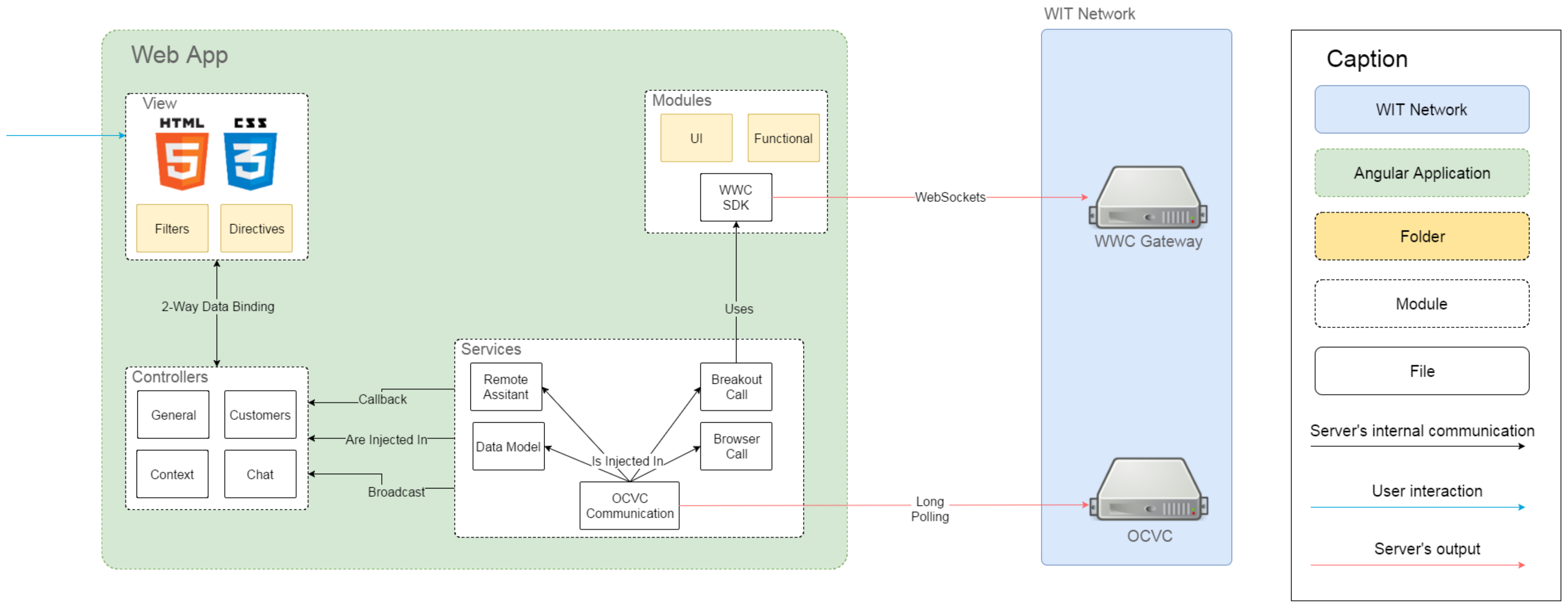


Figure 7. Web App classes

As described above, BO web app follow a MVVM pattern. This pattern is ensured using Angular JS framework. Other frameworks such as scroll glue were used in order to help UI management.

The web app is divided is four components: View, Controllers, Services and Modules.

Modules hold all the frameworks needed in the app.

Users interact with the view, composed by several partial HTML and CSS files using Bootstrap framework.

Controllers manage UI behavior, map users' actions and maintain states. Controller were divided in four groups of controller. General controllers manage global aspects of the BO such as window resizing, login and logout. Customers manage all customers' tabs and are responsible for managing features such as select a customer, answering or closing a conversation. Context manage the collected context and comments tabs and are responsible for editing customers' information, adding new comments or open URLs. Chat manage all chat's UI such as sending new messages or making calls.

Services are singletons injected in other components of the web app. After a service is injected in a component it can use service's exposed functions. Services were divided in OCVC Communication, Data Model, Remote Assistant, Browser Call and Breakout Call. If any component needs to communicate with OCVC server it injects OCVC Communication service and the communication is done by it, to receive information callbacks are registered in the same service that will notify the components. Data model holds the needed data received from server. The remaining services are used for call management. Browser call manage call between BO and widget by WebRCT. Breakout calls manage call between BO and a device. In this case WWC SDK is used, the SDK allow communication with WWC Gateway. In order to do so, SDK gives a Communication Interface to call server methods and events registration methods to receive answers from server. Remote Assistant manage calls between BO and iPad's Remote Assistant app and is an encapsulation of the lib developed implementing an Interface that allow the communication between BO and Remote Assistant Server using OCVC server as proxy.

3 Data model

Data in MongoDB, and in NoSQL databases in general, has a flexible schema. Unlike SQL databases where a declared tables' schema is defined before inserting data, in MongoDB no schema definition is needed. Instead of tables are used collections of documents where each document as the flexibility of mapping different data between themselves even if the data fields have substantial variations. In practice, however, the documents inside a collection share a similar structure.

This section presents the collections used in the internship, and the similar structure definition of their objects. Note that the presented representation of the documents is a general one. Each document inside the collections could have more or less fields than the presented. This is not a schema definition, instead is just a similar structure definition of the documents.

In MongoDB are used two tools to represent relationships between data:

- References: when documents store an ID from one document to another.
- Embedded documents: when a document structure is stored inside a field or array.

References are used when the same document must be stored several times inside a document or a collection. Instead of store the whole document, an ID to it is stored and data redundancy is kept at minimum. Embedded documents are used when the document to save does not justify the creation of a new document (e.g.: date/geographical structures).

Above are the collections created and their reference documents. A description of the collection and document is written, as well as all decisions are justified.

3.1 Customers' Collection

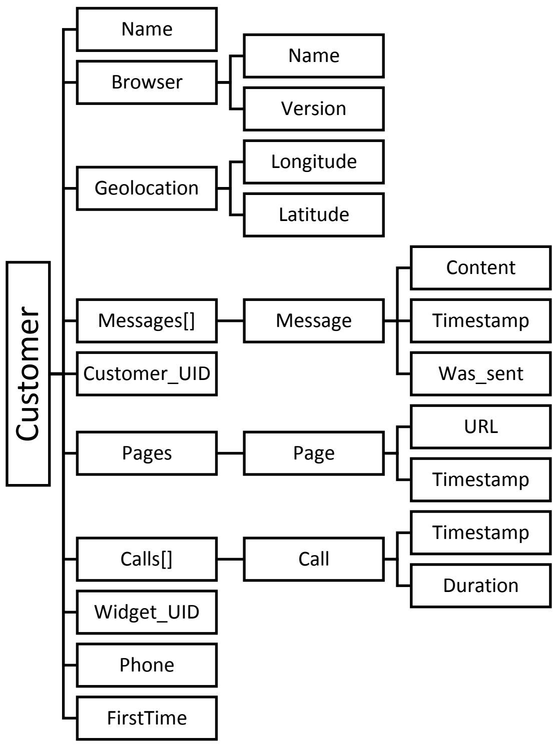


Figure 8. Customer document reference representation

Summary

Customers' collection hold the documents that reference each customer. Each customer will be stored in a document.

Figure 7. represents the reference document for the customers. Some of the fields will always exist every time a new document is created at the collection. This fields are Customer_UID and Pages. The other fields can or cannot be created depending on customer's actions during his visit. For instance, if a customer never chat with an operator the Messages[] is not needed.

For the geolocation field it was opted to use embedded documents instead of references, since every document can have very different geolocation values it was unnecessary to create a collection to hold every geolocation. The same goes to the Messages and Calls fields.

In the cases of the Widget_ID it was used references. It would make no sense to use embedded objects and increment the data redundancy on the database. The use of the references allow the documents to share the same Widget document without the need to store it in every Customer document.

Fields description

Name: The name the user inputs before start to talk with the operator.

Browser: Tuple of name and version to refer the browser used by the customer.

Geolocation: Tuple of latitude and longitude referring the customer location.

Messages[]: An array of message objects with content and timestamp when the message was sent. Was_sent marks if the message was sent from the customer or received.

Customer_UID: Customer unique ID calculated by the server.

Pages[]: Array of page objects tracked by the customer. Each page contains an URL and a timestamp.

Calls[]:Array of call objects did by the customer. Each call contains a duration and a timestamp.

Widget_UID: Reference to the Widget document that the customer is using.

3.2 Operators Collection

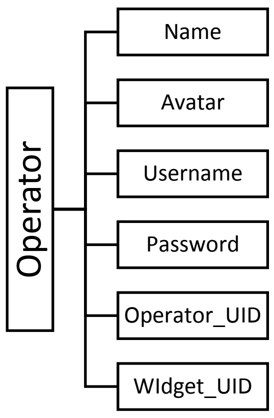


Figure 9. Operator document reference representation

Summary

This collection stores the operators' information. Each operator will be stored in one document. Figure 8. represents the reference document to the collection.

Since no composed fields were needed it was not used embedded objects. However, it was needed a reference to the widget the operator is associated with. The other fields are only used to store operator's profile information.

Fields description

Name: Operator's name.

Avatar: Can be a reference to an avatar collection or a path to the avatar itself.

Username: Operator's name used to authenticate in the system.

Password: Hash of the password used to authenticate in the system.

Operator_UID: Operator unique ID calculated by the server.

Widget_UID: Reference to the Widget document that the customer is using.

3.3 Online List Collection

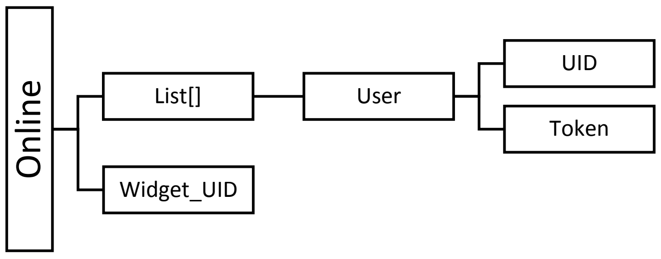


Figure 10. Online list document reference representation

Summary

Collection of documents that store the list of online users per widget. In the internship, this collection will only hold one object since the use case of serve multiple websites is out of scope. However, the document reference is already prepared to store multiple widgets.

In this case it was opted to mix references and embedded documents.

To store the online users it was used embedded objects with a reference to the user. To store the widget it was used a reference.

Fields description

List[]: Array of user objects which contain a reference to the UID. Token is the session token assigned to the corresponding user.

3.4 Widget Collection

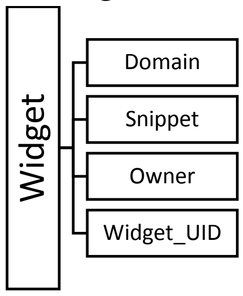


Figure 11. Widget document reference representation

Summary

Like the previous case, in spite of being a collection of documents that store the widget documents, this collection will only hold one document.

This collection stores widget’s information and widget’s owner information.

Fields description

Domain: Domain where the widget will run.

Snippet: Path to JavaScript snippet or the actual snippet itself.

Owner: Owner’s personal information. Yet to define.

Widget_UID: Widget unique ID calculated by the server.

4 Interfaces

In this chapter are presented the interfaces used on the system.

Firstly will be presented the communication Interface used between OCVC server, BO and widget. After that will be presented the WWC Interface used in breakout calls. Lastly will be presented the Interface used by database module.

4.1 Communication Interface

Communication Interface will be divided in 3 subsections: Server Events, BO Events and Widget Events. Those sections will describe the events emitted by each container. The description have the receiver of the event, the object emitted and an explanation.

4.1.1 Server Events

user:StartChat

- Emitted to: All operators.
- Emitted Object: Custom object ({UID: String, timestamp: Date Object}).
- Explanation: Tell the operators that the customer with the UID (user ID) requested a chat. The timestamp indicates the moment of the request.

user:updateType

- Emitted to: Operators.
- Emitted Object: Custom Object ({UID: String, type: String, timestamp: Date}).
- Explanation: Notify the operators that the conversation type of the customer *UID* has changed to *type*.

user:OS

- Emitted to: Operators.
- Emitted Object: Custom Object ({info: String, UID: String}).
- Explanation: Notify the operators to update the customer *UID*'s operative system field with *info*.

user:newReferrer

- Emitted to: operators.
- Emitted Object: Custom Object ({UID: String, url: String}).
- Explanation: Notify the operators to add a new URL to the customer *UID*'s URL list.

user:Mail

- Emitted to: Operators.
- Emitted Object: Custom Object ({mail: String, UID: String}).
- Explanation: Notify the operators to update the customer *UID*'s mail field with *mail*.

user:browser

- Emitted to: Operators.
- Emitted Object: Custom Object ({info: String, UID: String}).
- Explanation: Notify the operators to update the customer *UID*'s browser field with *info*.

user:town

- Emitted to: Operators.
- Emitted Object: Custom Object ({info: String, UID: String}).
- Explanation: Notify the operators to update the customer *UID*'s town field with *info*.

user:country

- Emitted to: Operators.
- Emitted Object: Custom Object ({info: String, UID: String}).
- Explanation: Notify the operators to update the customer *UID*'s country field with *info*.

user:Message

- Emitted to: Operators.
- Emitted Object: Custom Object ({payload: String, UID: String}).

- Explanation: Send message *payload* from customer *UID* to all operators.

user:newCall
- Emitted to: Operators.
- Emitted Object: String.
- Explanation: Notify the operators that the customer requested a browser call.

user:endCall
- Emitted to: Operators.
- Emitted Object: String.
- Explanation: Notify the operators that the customer ended the browser call.

user:closeConversation
- Emitted to: Operators.
- Emitted Object: Custom Object ({UID: String, timestamp: Date}).
- Explanation: Notify operators that the conversation from customer *UID* has been closed.

user:out
- Emitted to: Operators.
- Emitted Object: Custom Object ({UID: String}).
- Explanation: Notify the operators that customer *UID* has left the site.

syncTime
- Emitted to: Operator/Customer.
- Emitted Object: Custom Object ({y: Date, x: Date, a: Date}).
- Explanation: Send to new user the terms to use in NTP calculus.

user:new
- Emitted to: Operators.
- Emitted Object: Custom Object (Customer).
- Explanation: Send to the operators the information of customer that entered the site.

yourInfo
- Emitted to: Customer.
- Emitted Object: Custom Object (operator: Object, customer: Object).
- Explanation: Send to customer his personal information and the information of the operator that was assigned to him.

startTrack
- Emitted to: Customer.
- Emitted Object: None.
- Explanation: Notifies the widget that it can start to track context information.

user:online
- Emitted to: Operators.
- Emitted Object: Custom Object ({UID: String, timestamp: Date}).
- Explanation: Notify the operators that customer *UID* is now online.

user:newURL
- Emitted to: Operators.
- Emitted Object: Custom Object ({UID: String, url: String}).
- Explanation: Tell the operators to add *url* to the customer *UID*'s URL list.

operator:Info
- Emitted to: Operator.
- Emitted Object: Custom Object ({name: String, ID: String, photo: String, WWCname: String, WWCpass: String}).
- Explanation: Send to the operator his personal information.

users:online
- Emitted to: Operator.
- Emitted Object: Custom Object array.
- Explanation: Send to operator the list of online customers.

users:history
- Emitted to: Operator.
- Emitted Object: Custom Object array.
- Explanation: Send to operator the list of customers on history.

operatorLoged
- Emitted to: Operators/Customers.
- Emitted Object: None.
- Explanation: Notify everyone that a operator logged in at the BO.

user:isTyping
- Emitted to: Operators.
- Emitted Object: String.
- Explanation: Notify operators that a customer is typing.

user:stopedTyping
- Emitted to: Operators.
- Emitted Object: String.
- Explanation: Notify operators that a customer stopped typing.

operator:isTyping
- Emitted to: Customer.
- Emitted Object: None.
- Explanation: Notify customer that his operator is typing.

operator:stopedTyping
- Emitted to: Customer.
- Emitted Object: None.
- Explanation: Notify customer that his operator stopped typing.

IceCandidate
- Emitted to: Operators/Customer.
- Emitted Object: String.
- Explanation: Send ICE candidate from operator to customer and vice-versa.

offerSDP
- Emitted to: Operators.
- Emitted Object: String.
- Explanation: Send session description offer from customer to operators.

answerSDP
- Emitted to: Customer.
- Emitted Object: String.
- Explanation: Send session description answer from operator to customer.

operator:callEnded
- Emitted to: Customer
- Emitted Object: None
- Explanation: Notify the customer that the browser call has been closed.

operator:Message
- Emitted to: Customer.
- Emitted Object: Custom Object ({payload: String, UID: String}).
- Explanation: Send message *payload* from operator *UID* to customer.

widget:Message
- Emitted to: Customer.
- Emitted Object: Custom Object ({payload: String, UID: String}).
- Explanation: Send message *payload* from customer *UID* to customer.

noOperatorsOn
- Emitted to: Customers.
- Emitted Object: None.
- Explanation: Notify customers that none operator is online.

user:loggedOut
- Emitted to: Operators.
- Emitted Object: Custom Object ({UID: String}).
- Explanation: Notify operators that customer *UID* is offline.

user:comment
- Emitted to: Operators.
- Emitted Object: Custom Object ({payload: String, UID: String}).
- Explanation: Send comment *payload* to append to customer *UID* to all operators.

user:selected
- Emitted to: Operators.
- Emitted Object: Custom Object ({UID: String, operator: ID, time: Date}).
- Explanation: Notify the operators that the operator *operator* is answering the customer *UID*.

operatorAssigned
- Emitted to: Customer.
- Emitted Object: String.
- Explanation: Notify the customer that he will be answered by the passed operator.

closeConversation
- Emitted to: Customer.
- Emitted Object: None.
- Explanation: Notify the customer that his conversation has been closed.

user:Called
- Emitted to: Operators.
- Emitted Object: Custom Object ({UID: String, operator: String, request: Integer, time: Date}).
- Explanation: Unused.

operator:callAccepted
- Emitted to: Customer.
- Emitted Object: None.
- Explanation: Notify the customer that his browser call has been accepted.

tablet:New
- Emitted to: Operators
- Emitted Object: Custom Object (Customer)
- Explanation: Notify the operators that a new tablet customer arrived to the site.

tablet:newCall
- Emitted to: Operators
- Emitted Object: Custom Object (Call)
- Explanation: Tell to the operators to add a new call to the previous tablet customer.

turnOffStreams
- Emitted to: Operators
- Emitted Object: None
- Explanation: Notify the operators to close all media streams if the operator were in a tablet conversation

tablet:pickupTabletCall
- Emitted to: Operators.
- Emitted Object: Custom Object (config: Object, currentSdpOffer: String, UID: String).
- Explanation:

4.1.2 BO Events

operator:Call
- Emitted to: Server.

- Emitted Object: Custom Object ({UID: String, n: Integer}).
- Explanation: Sent from operator to answer call *n* from user with the ID *UID*.

operator:isTyping
- Emitted to: Server.
- Emitted Object: String.
- Explanation: Sent from operator to notify the customer that the operator is typing.

operator:stopedTyping
- Emitted to: Server.
- Emitted Object: String.
- Explanation: Sent from operator to notify the customer that the operator stopped typing.

operator:userSelected
- Emitted to: Server.
- Emitted Object: String.
- Explanation: Sent from operator to select a customer to talk with.

operator:Message
- Emitted to: Server.
- Emitted Object: Custom Object ({payload: String, from: String, timestamp: Date, UID: String}).
- Explanation: Sent message *payload* to customer with ID *UID*.

operator:Mail
- Emitted to: Server.
- Emitted Object: Custom Object ({payload: String, from: String, timestamp: Date, UID: String}).
- Explanation: Sent message *payload*, via mail, to customer with ID *UID*.

operator:callAccepted
- Emitted to: Server.
- Emitted Object: String.
- Explanation: Accept browser call from a customer.

operator:callEnded
- Emitted to: Server.
- Emitted Object: String.
- Explanation: End an active browser call with a customer.

operator:setUserName
- Emitted to: Server.
- Emitted Object: Custom Object ({name: String, UID: String}).
- Explanation: Set the *UID* customer's name to *name*.

operator:setUserPhone
- Emitted to: Server.
- Emitted Object: Custom Object ({phone: String, UID: String}).
- Explanation: Set the *UID* customer's phone number to *phone*.

operator:setUserMail
- Emitted to: Server.
- Emitted Object: Custom Object ({mail: String, UID: String}).
- Explanation: Set the *UID* customer's mail to *mail*.

operator:comment
- Emitted to: Server.
- Emitted Object: Custom Object ({payload: String, fromID: String, from: String, timestamp: Date, UID: String}).
- Explanation: Add note *payload* to customer with ID *UID*.

operator:hangupTabletCall
- Emitted to: Server.

- Emitted Object: String.
- Explanation: End an active tablet call with a customer.

operator:closeConversation

- Emitted to: Server.
- Emitted Object: String.
- Explanation: End an active conversation with a customer.

operator:pickupTabletCall

- Emitted to: Server.
- Emitted Object: String.
- Explanation: Answer a call from a customer.

operator:breakoutStatus

- Emitted to: Server.
- Emitted Object: Custom Object ({ status: String, UID: String}).
- Explanation: Set the customer's breakout status to *status*.

syncTime

- Emitted to: Server.
- Emitted Object: Date.
- Explanation: Emit time value to perform the NTP calculus.

operator:answerTabletCall

- Emitted to: Server
- Emitted Object: Custom Object ({currentSdpOffer: String, UID: String})
- Explanation: Send the session description offer to the customer *UID* in order to answer his call request from tablet.

answerSDP

- Emitted to: Server
- Emitted Object: Custom Object ({ UID: String, sdp: String })
- Explanation: Send the session description offer to the customer *UID* in order to answer his call request from browser.

operator:candidate

- Emitted to: Server
- Emitted Object: String
- Explanation: Send ICE candidates in order to start a call between browsers.

4.1.3 Widget Events

widget:offlineRequest

- Emitted to: Server.
- Emitted Object: Custom Object ({name: String, mail: String, phone: String, payload: String, type: Integer})
- Explanation: Send a request to be contacted later via the specified (*type*) communication channel.

widget:sendMeMyConveration

- Emitted to: Server.
- Emitted Object: None.
- Explanation: Asks to server to send him the previous conversations available.

widget:requestForCall

- Emitted to: Server.
- Emitted Object: None.
- Explanation: Make a request to be contacted by browser call.

widget:Message

- Emitted to: Server.
- Emitted Object:Custom Object ({payload: String, timestamp: Date}).

- Explanation: Send message *payload* to the operators.

widget:stopedTyping
- Emitted to: Server.
- Emitted Object: None.
- Explanation: Notify the operators that the customer stopped typing.

widget:isTyping
- Emitted to: Server.
- Emitted Object: None.
- Explanation: Notify the operators that the customer started typing.

widget:Mail
- Emitted to: Server.
- Emitted Object: String
- Explanation: Set customer's mail contact.

syncTime
- Emitted to: Server.
- Emitted Object: Date.
- Explanation: Emit time value to perform the NTP calculus.

widget:coords
- Emitted to: Server.
- Emitted Object: String array.
- Explanation: Set customer's coordinates.

widget:browser
- Emitted to: Server.
- Emitted Object: String array.
- Explanation: Set customer's browser information.

widget:referrer
- Emitted to: Server.
- Emitted Object: String
- Explanation: Set customer's page referrer when customer arrives to the page.

widget:OS
- Emitted to: Server.
- Emitted Object: String.
- Explanation: Set customer's operative system information.

widget:country
- Emitted to: Server.
- Emitted Object: String
- Explanation: Set customer's country information.

widget:candidate
- Emitted to: Server
- Emitted Object: String
- Explanation: Send ICE candidates in order to start a browser call.

offerSDP
- Emitted to: Server
- Emitted Object: String
- Explanation: Send session description offer in order to start a browser call.

4.2 WWC Gateway Interface

This subsection describes the methods used and events registered to communicate with WWC Gateway. These descriptions were transcribed from WIT's WIT WebRTC Gateway JavaScript SDK Guide [5].

4.2.1 Event Interface Registration

Method	**sdk.event.bind**		
Description	Binds new functions to be called when an event is fired		
Parameters	**Name**	**Type**	**Description**
	name	String	Name of the Event to listen to
	listener	Object	Function or Array of Functions to be called when the event is fired
Usage	sdk.event.bind(name, listener)		
Returns	Nothing		

Table 15 sdk.event.bind API

Method	**sdk.event.unbind**		
Description	Unbinds functions allocated to a certain event (given that the function as a function name)		
Parameters	**Name**	**Type**	**Description**
	name	String	Name of the Event to remove listeners from
	listener	Object	Function or Array of Functions to be unbound
Usage	sdk.event.unbind(name, listener)		
Returns	Nothing		

Table 16 sdk.event.unbind API

4.2.2 Session Interface Methods

Method	**sdk.session.register**		
Description	Attempts to Register the User through WIT’s WebRTC Gateway		
Parameters	**Name**	**Type**	**Description**
	username	String	User’s URI
	password	String	User’s Password
Usage	sdk.session.register(username, password)		
Returns	Promise		
On Promise resolved	"onSessionStatus:connected”		
	"onSessionConnect:error"	Failed to register	

On Promise rejected	"onSessionStatus:disconnected"	Session was terminated or connection was closed

Table 17 sdk.session.register API

Method	**sdk.session.terminate**
Description	Attempts to terminate the user session in the Gateway
Parameters	None
Usage	sdk.session.terminate()
Returns	Nothing

Table 18 sdk.session.terminate API

4.2.3 Session Interface Events

Event	**onSessionStatus**		
Condition	The state of the user session changed		
Callback Parameters	Object with parameters		
Parameters	**Name**	**Type**	**Description**
	status	String	The session status
Possible Values	"connected"	Successfully connected to the WWG	
	"disconnected"	Connection to the WWG was lost	
	"terminated"	Session was Terminated	

Table 19 onSessionStatus Event API

4.2.4 Call Interface Methods

Method	**sdk.call.invite**		
Description	Triggers a Call Invite Request to a given user		
inviteParams Object parameters	**Name**	**Type**	**Description**
	to	Identity	Mandatory parameter to identify the destination of the call
	mediaType	String	Mandatory parameter (VOICE/VIDEO)
	localStream	Media Stream	Optional; MediaStream object to be used in this particular call. If not present, the default one will be requested.

	audioConstraints	navigator.getUserMedia constraints	Optional; If localStream is not present, the JavaScript SDK will request the local MediaStream. If these constraints are available they will be used when requesting the stream.
	videoConstraints	navigator.getUserMedia constraints	
Usage	sdk.call.invite(inviteParams)		
Returns	Promise		
On Promise resolved	{call: Call}. Refer to 4.2.8		
On Promise rejected	Error Description		

Table 20 sdk.call.invite API

Method	**sdk.call.terminate**		
Description	Terminate a call. This method can be used to cancel an ongoing INVITE or to terminate an ongoing call.		
Parameters	**Name**	**Type**	**Description**
	id	String	Mandatory parameter to identify the call to be terminated (from the Call object). Refer to 4.2.8
Usage	sdk.call.terminate(id)		
Returns	Promise		
On Promise resolved	Nothing		
On Promise rejected	Error Description		

Table 21 sdk.call.terminate API

4.2.5 Call Management Methods

Method	**sdk.call.media.isMicMuted**		
Description	Checks if the local audio stream is enabled		
Parameters	**Name**	**Type**	**Description**
	id	String	Call ID (from the Call object). Refer to 4.2.8
Usage	sdk.call.media.isMicMuted(id)		
Returns	True if the local audio stream associated to the active call is **disabled** (enabled = false).		

Table 22 sdk.call.media.isMicMuted API

Method	**sdk.call.media.changeLocalAudioState**		
Description	Changes the enabled state of the local audio stream		
Parameters	**Name**	**Type**	**Description**
	id	String	Call ID (from the Call object). Refer to 4.2.8
	state (optional)	Boolean	Enabled state. If not present current state is reversed.
Usage	sdk.call.media.changeLocalAudioState(id, state)		
Returns	The new state of the local audio stream (enabled = true / false).		

Table 23 sdk.call.media.changeLocalAudioState API

4.2.6 Call Interface Events

Event	**onCallEvent_<callid>**		
Condition	Event called when the state of a Call with the id *<callid>* changes		
Callback Parameters	Object with parameters		
Object Parameters	**Name**	**Type**	**Description**
	call	Call	Refer to 4.2.8
	status	String	Refer to 4.2.8
	args	ServerResponse	Optional. Refer to point 4.2.7

Table 24 onCallEvent Event API for a specific call

4.2.7 Server Response Code Object

Parameter	**Type**	**Description**
value	Number	SIP response code
description	String	Error Code Description

Table 25 Fields available in the ServerResponseCode object

4.2.8 Call Object

Parameter	**Type**	**Description**
id	String	Call ID shared between the Web Application and the SDK
type	String	Call Type (“VOICE”, “VIDEO”)
direction	String	Call Direction (“INCOMING”, “OUTGOING”)
from	Identity	Identity representing the caller
to	Identity	Identity representing the callee

isVideoShare	Boolean	Identifies if this call is a video share or not
state	String	Call State (See table below for possible values)
localStream	MediaStream	Local Media Stream
remoteStream	MediaStream	Remote Media Stream

Table 26 Fields available in the Call object

The Call object can have various states. The possible Call States are the following:

Parameter	**Description**	**Final State?**
"terminated"	Call was Terminated	YES
"failed"	Call failed	YES
"rejected"	Call was Rejected by the other peer	YES
"busy"	Call was Reject due to the other peer being Busy	YES
"established"	Call was successfully Established	NO
"accepted"	Call was accepted and is being established	NO
"ringing"	Call is in Ringing State and awaiting to be accepted or rejected	NO
"progress"	Call is in Progress State (183 – Session Progress)	NO
"invite"	Call is in Invite State, no ringing response received yet	NO
"local_hold"	Call is in hold state has result to a toggle request from local client	NO
"remote_hold"	Call is in hold state has result to a toggle request from a remote client	NO
"both_points_hold"	Call is on hold state from both call end points	NO

Table 27 Possible call states

4.3 Database Module Interface

Database interface expose three functions: query, insert and remove. Each receive three parameters: option, information and a callback function. Below is presented a description of the function for the option passed to it.

Query function queries elements from the database, remove removes them and insert inserts them if they do not exist, if they do updates their value.

4.3.1 Query

Login

- **Data:** Custom Object (username: String, password: String).
- **Explanation:** Returns 0 if credentials are right, less than 0 if not.

Customer

- **Data:** Custom Object (UID: String).
- **Explanation:** Returns (customer, 1) if customer exist, ([], 0) if not.

Operator

- **Data:** Custom Object (UID).
- **Explanation:** Return operator if exists.

isTokenExpired

- **Data:** Custom Object (token: String).
- **Explanation:** Return true if token *token* is valid, false if not.

pendingCustomers

- **Data:** None.
- **Explanation:** Return the list of customers with pending requests.

History

- **Data:** None.
- **Explanation:** Return the list of customers in history.

4.3.2 Insert

newCustomer

- **Data:** Custom Object (UID: String, urlList: Object Array, time: Date).
- **Explanation:** Insert new customer in Database and returns it.

Login

- **Data:** Custom Object (UID: String, token: String).
- **Explanation:** Set operator *UID* as online.

URL

- **Data:** Custom Object (UID: String, urlList: Object Array).
- **Explanation:** Append *urlList* to *customer* UID, returning customer object.

Browser

- **Data:** Custom Object (UID: String, browser: String).
- **Explanation:** Persist browser information in database. If information already exist update it.

OS

- **Data:** Custom Object (UID: String, OS: String).
- **Explanation:** Persist OS information in database. If information already exist update it.

Country

- **Data:** Custom Object (UID: String, country: String).
- **Explanation:** Persist country information in database. If information already exist update it.

Town

- **Data:** Custom Object (UID: String, town: String).
- **Explanation:** Persist town information in database. If information already exist update it.

Name

- **Data:** Custom Object (UID: String, name: String).
- **Explanation:** Persist name information in database. If information already exist update it.

Mail

- **Data:** Custom Object (UID: String, mail: String).
- **Explanation:** Persist mail information in database. If information already exist update it.

Phone

- **Data:** Custom Object (UID: String, phone: String).
- **Explanation:** Persist phone information in database. If information already exist update it.

Info

- **Data:** Custom Object (UID: String, mail: String, name: String, phone: String).
- **Explanation:** Unused.

Message

- **Data:** Custom Object (UID: String, message: String).
- **Explanation:** Push message into message’s at database.

Comment

- **Data:** Custom Object (UID: String, comment: String).
- **Explanation:** Push comment into comment’s at database.

Pending

- **Data:** Custom Object (UID: String, isPending: boolean).
- **Explanation:** Change isPending value.

Type

- **Data:** Custom Object (UID: String, type: String).
- **Explanation:** Change type value.

Time

- **Data:** Custom Object (UID: String, time: Date).
- **Explanation:** Update time value.

4.3.3 Remove

Logout

- Data: Custom Object (UID: String).
- Explanation: Set operator with *UID* offline.

Comment

- Data: Custom Object (UID: String, payload: String).
- Explanation: Remove comment *payload* from customer *UID*.

5 Referências

[1] S. Brown, “Simon Brown,” [Online]. Available: http://www.codingthearchitecture.com/authors/sbrown/. [Acedido em 12 November 2015].

[2] S. Brown, “C4 model poster,” [Online]. Available: http://www.codingthearchitecture.com/2014/08/24/c4_model_poster.html. [Acedido em 12 November 2015].

[3] Wikipedia, “Component-Based Software,” 7 June 2016. [Online]. Available: https://en.wikipedia.org/wiki/Component-based_software_engineering. [Acedido em 15 June 2016].

[4] Wikipedia, “Event-Driven Architecture,” 7 June 2016. [Online]. Available: https://en.wikipedia.org/wiki/Event-driven_architecture. [Acedido em 15 June 2016].

[5] L. Matos, A. Silva, J. Martins e L. Fernandes, em *WIT WebRTC Gateway Javascript SDK Guide*, 2016, p. June.

Masters in Informatics Engineering

Internship 2015/2016
Final Report

Online context for voice communications

Annex D – Development

José Manuel Marques Grilo

jgrilo@student.dei.uc.pt

Supervisors:

Jorge Sousa
Luís Matos

Carlos Bento

July, 1st 2016

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

Index

Index of Tables

Index of Figures

1 Introduction

This annex was written in order to give the reader a more detailed view of the development process, the challenges that were overcome, tests implementation and project's future work.

This annex is divided into 3 major sections:

- **Developed work**: this section provides an overview of the work developed over the past year, and challenges that were overcome in this section are presented the features implemented and their implementation challenges.
- **Tests**: here is provided the test methodology and sets of tests used to validate the product.
- **Future work**: this section provides a list of features that can be implemented in order to enrich the product.

2 Developed Work

A good quality planning and architecture design help to predict and overcome most of the challenges that will appear during the development. However, sometimes if planned process is not the clear unexpected problems can appear.

During the internship planning most of the problems were predicted and contingency plans were created in order to minimize its risks, still, some were impossible to predict and there was a need to create strategies to resolve them.

This section are presented the major features implemented. Here the functionalities are described and the sequence diagrams are presented along with some screenshots.

2.1 UI specification and implementation

For the UI definition the intern worked with WIT designer Elizabeth Pereira.

The first step was the definition of the requirements and user stories. With that document ready, the designer drew the mockups, which were evaluated by all the team members (scrum master, product owner and scrum team) as well as the stakeholder.

The process of defining an appealing UI that would give to the final user a great experience was progressive and done in several iterations. Elizabeth was in charge of drawing the mockups taking into account the documents of the internship.

Once a satisfactory result was achieved from all involved in the project, which is presented at Figure 2.1 and Figure 2.2, the UI implementation began.

The UI was implemented using Bootstrap 3.3.5 framework that helped to make the back office responsive. For the back office AngularJS framework was used, in order to implement an MVVM model.

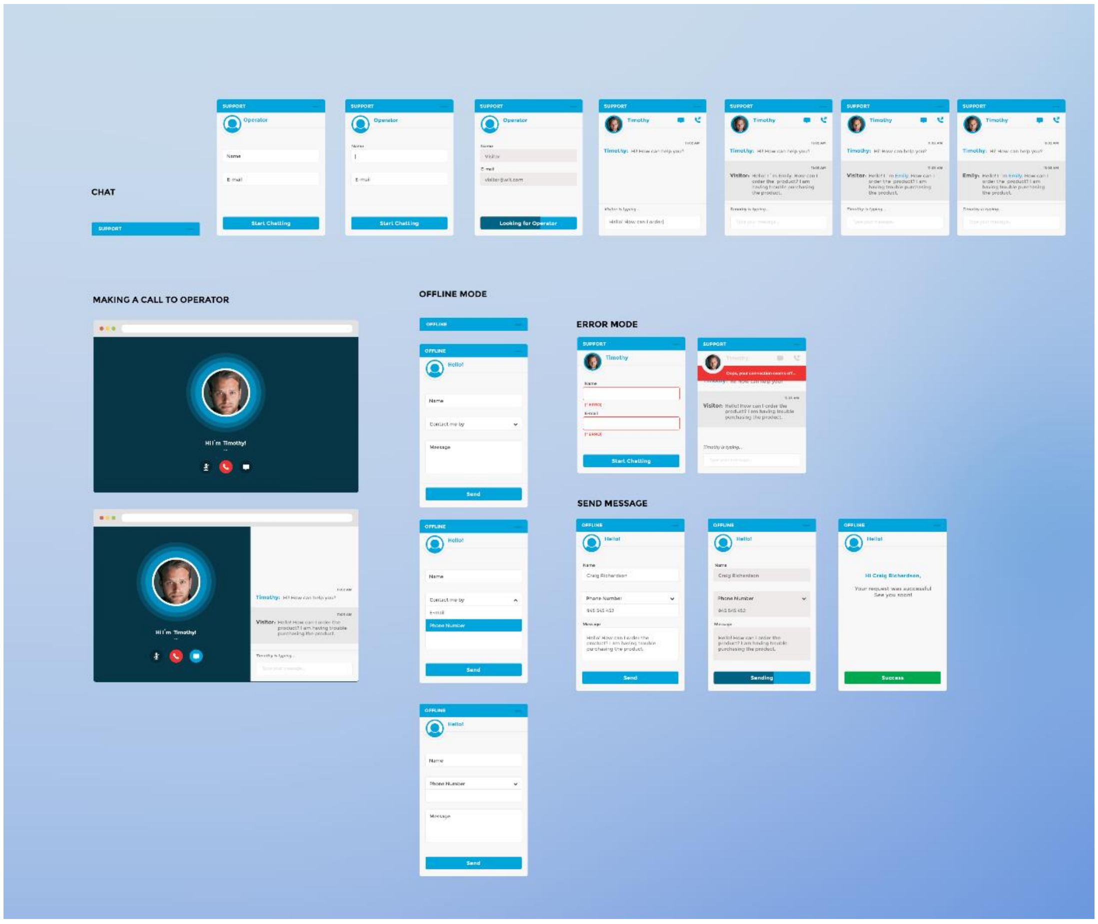


Figure 2.1. First design for widget's UI

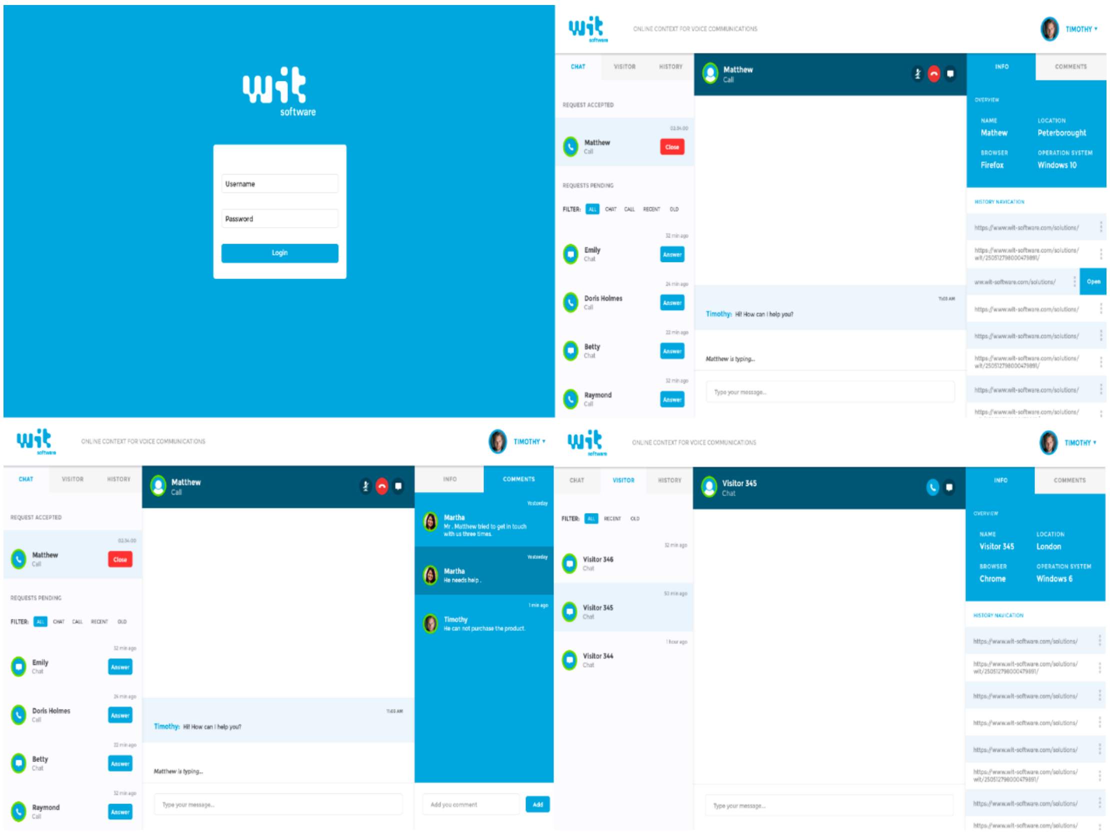


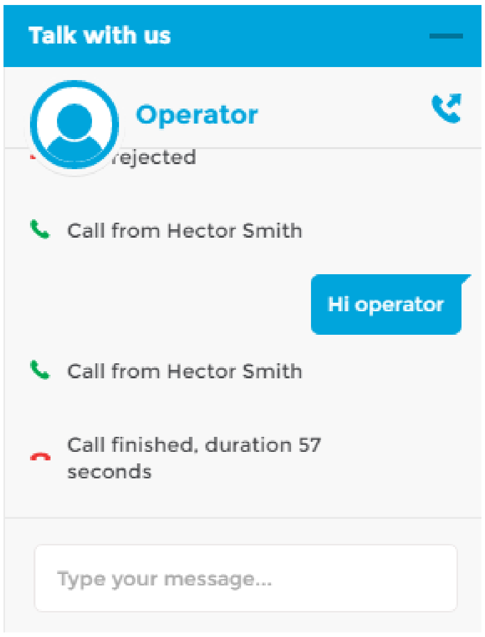
Figure 2.2. First design for back office's UI

The UI implementation was one of the biggest challenges in the project.

It was very hard to find a UI that was appealing to everyone involved in the project, and that suited both operators and customers' purposes in terms of user experience. Another major factor that influenced the UI implementation was the find of new features to implement. These requirement changes were taken into account during project's planning; therefore development was not affected with these changes. However, this led to a major change of the UI.

From the first UI mockups to the last, there were several changes. One of the main changes was the widget's communication flow – in order to attract more customers the identification form was removed and costumers can start a conversation without identify themselves. Messages displayed suffered several changes on development both on BO and the widget – messages are now grouped if they were received at the exact same minute. In the BO some paddings were adjusted between elements to refine the UI. Beside that the contextual information, the column changed a bit along with the communication buttons used to start and answer calls.

The aspect of the UI is presented on Figure 2.3 to Figure 2.7.


Talk with us

Operator

rejected

Call from Hector Smith

Hi operator

Call from Hector Smith

Call finished, duration 57 seconds

Type your message...

Figure 2.3. Final Widget UI

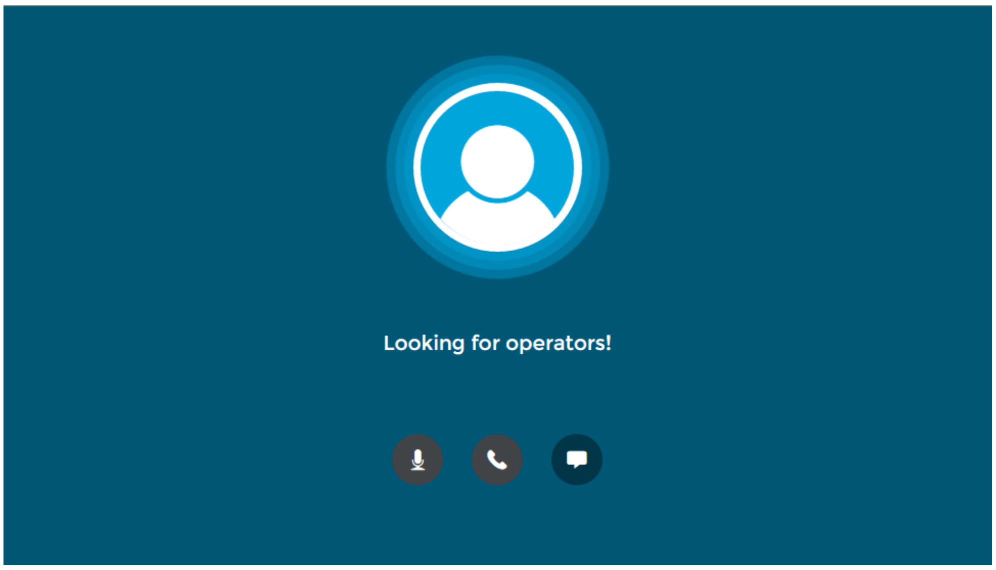
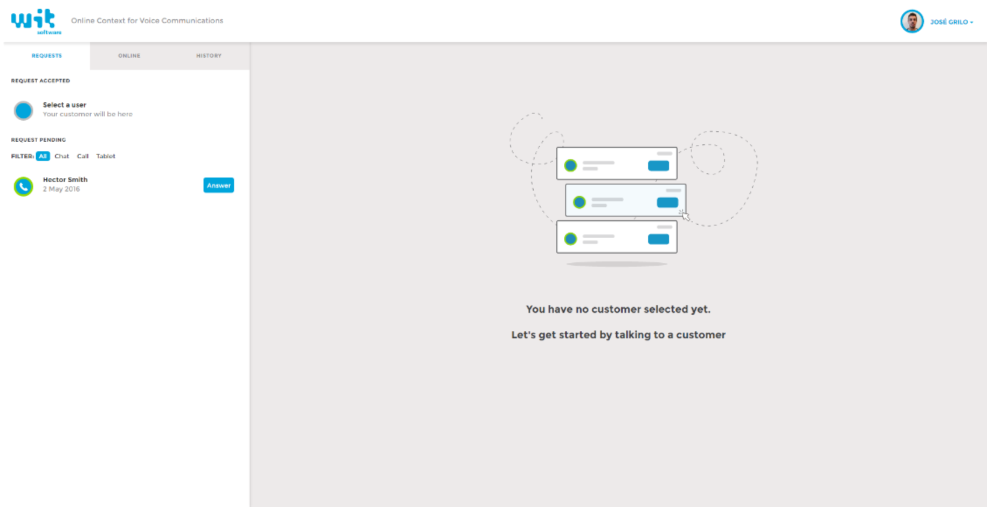

Looking for operators!

Figure 2.4. Customer's call window UI



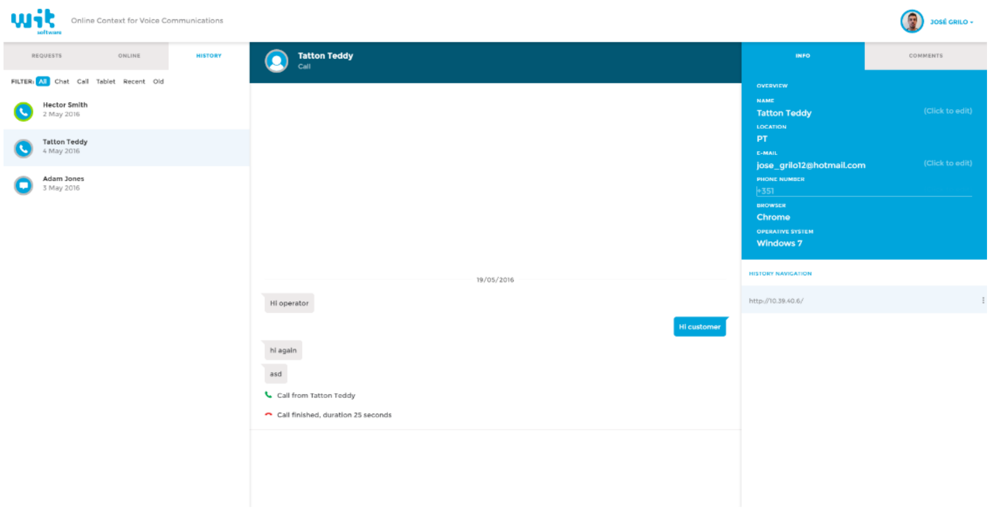
Select a user
Your customer will be here

Hector Smith
2 May 2016

You have no customer selected yet.

Let's get started by talking to a customer

Figure 2.5. Final back office UI - Entrance screen


Online Context for Voice Communications
JOSÉ GRILO
REQUESTS
ONLINE
HISTORY
FILTER: All Chat Call Tablet Recent Old
INFO
COMMENTS
HISTORY NAVIGATION

Hector Smith
2 May 2016

Tatton Teddy
4 May 2016

Adam Jones
3 May 2016

Tatton Teddy
Call

19/05/2016

Hi operator

Hi customer

hi again

asd

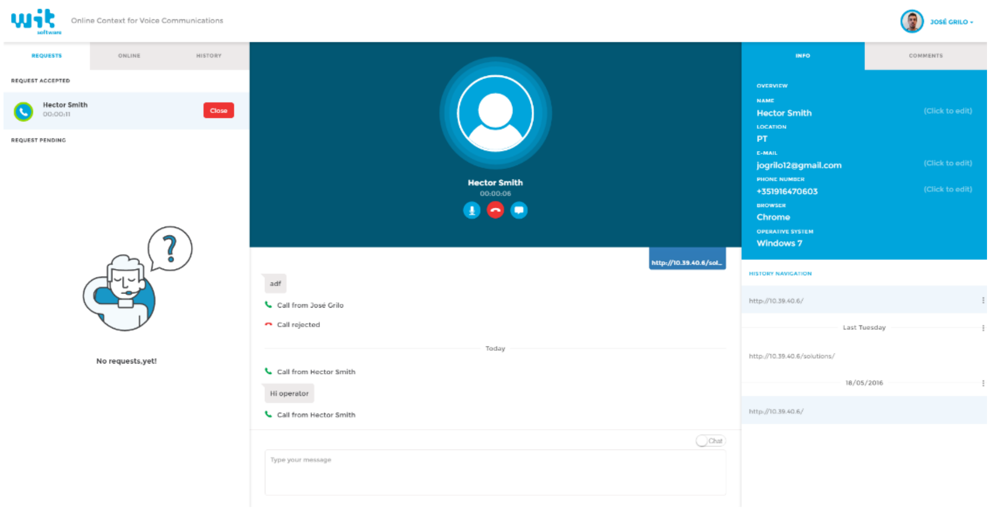
Call from Tatton Teddy

Call finished, duration 25 seconds

OVERVIEW
NAME: Tatton Teddy (Click to edit)
LOCATION: PT
E-MAIL: jose_grilo12@hotmail.com (Click to edit)
PHONE NUMBER: +351
BROWSER: Chrome
OPERATIVE SYSTEM: Windows 7

http://10.39.40.6/

Figure 2.6. Final back office UI - Customer selected



Hector Smith
00:00:11

No requests yet!

Hector Smith
00:00:06

http://10.39.40.6/sol...

asf

Call from Jose Grilo

Call rejected

Today

Call from Hector Smith

Hi operator

Call from Hector Smith

OVERVIEW
NAME: Hector Smith (Click to edit)
LOCATION: PT
E-MAIL: jogrilo12@gmail.com (Click to edit)
PHONE NUMBER: +351916470603 (Click to edit)
BROWSER: Chrome
OPERATIVE SYSTEM: Windows 7

http://10.39.40.6/

Last Tuesday

http://10.39.40.6/solutions/

18/05/2016

http://10.39.40.6/

Figure 2.7. Final back office UI - On call

2.2 Operator's Login and Socket Disconnection

Not everyone can use the back office to answer customers' requests. Since that was a major requirement an authentication window was created. This window pops-up before the user enters the back office administration window. Here in Figure 2.8, the user fills in his login information and it is sent to the backend using HTTPS. Once the credentials have been authenticated, the back office will be displayed, as shown in Figure 2.9, and a connection to the backend is created.

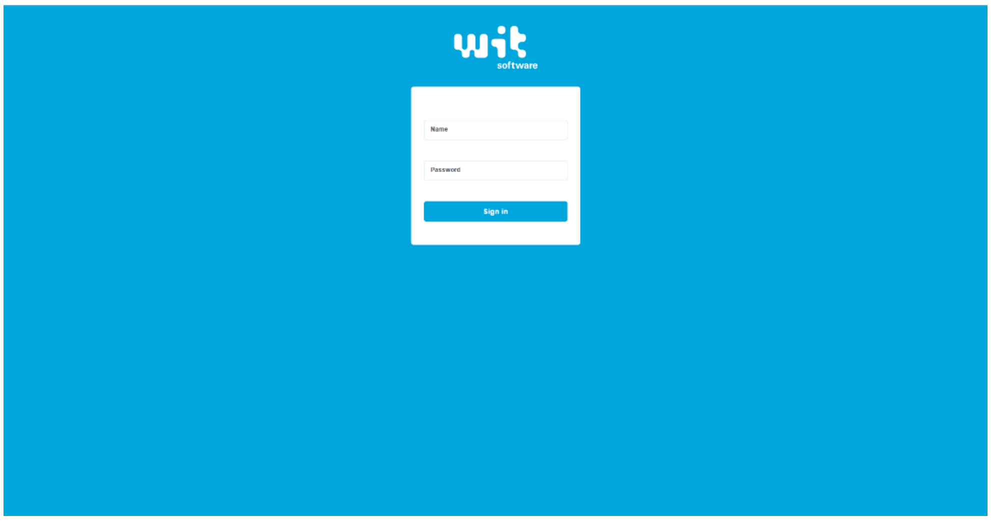


Figure 2.8. Back office's login page

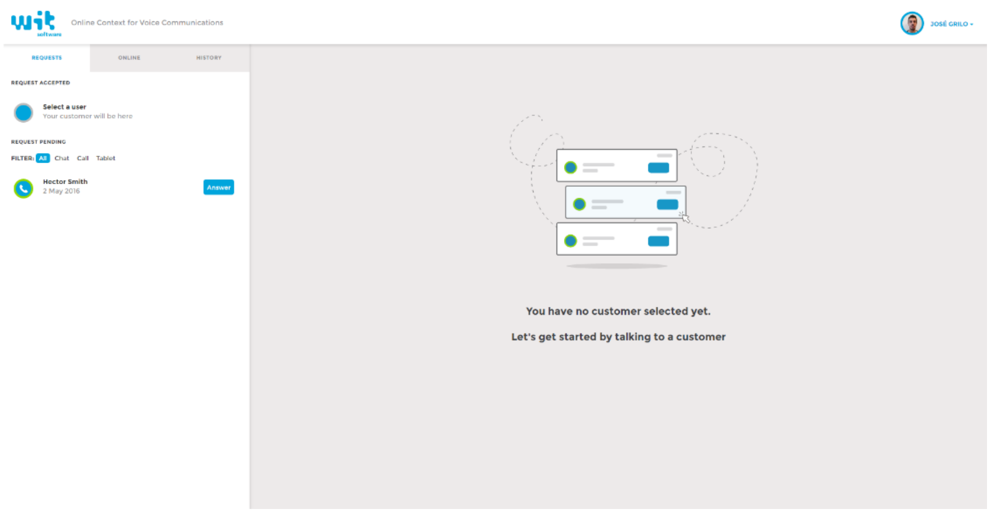

software
Online Context for Voice Communications
JOSÉ GRILO
REQUESTS
ONLINE
HISTORY
REQUEST ACCEPTED
Select a user
Your customer will be here
REQUEST PENDING
FILTER: All Chat Call Tablet
Hector Smith
2 May 2016
Answer
You have no customer selected yet.
Let's get started by talking to a customer

Figure 2.9. Back office's entrance screen

It’s as equally important to have a logout action, as it is to have a login action in place, which allows the operation to close the working session. In this case the connection is broken and the operator automatically returns to the login page (Figure 2.10).

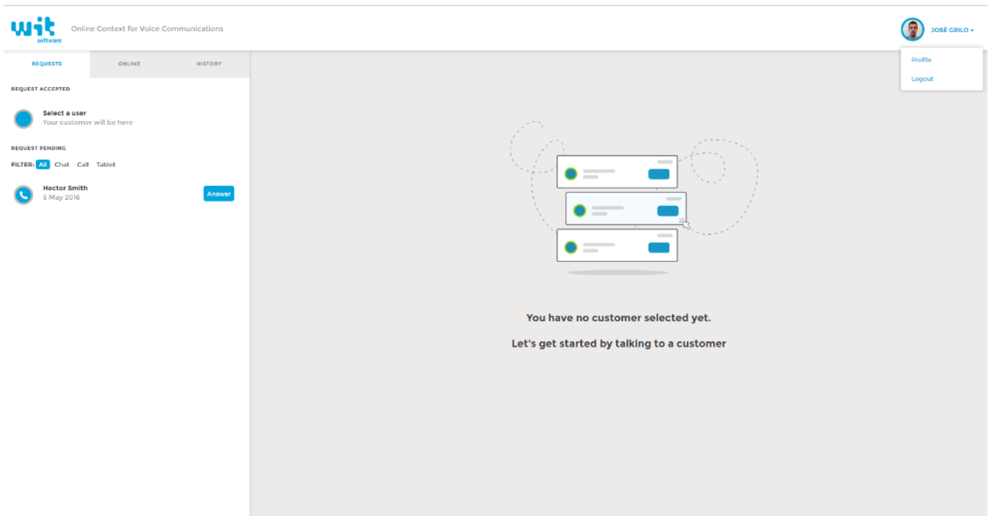

wit software
Online Context for Voice Communications
JOSÉ GRILO
Profile
Logout
REQUESTS
ONLINE
HISTORY
REQUEST ACCEPTED
Select a user
Your customer will be here
REQUEST PENDING
FILTER: All Chat Call Tablet
Hector Smith
5 May 2016
Answer
You have no customer selected yet.
Let's get started by talking to a customer

Figure 2.10. Back office's logout use case

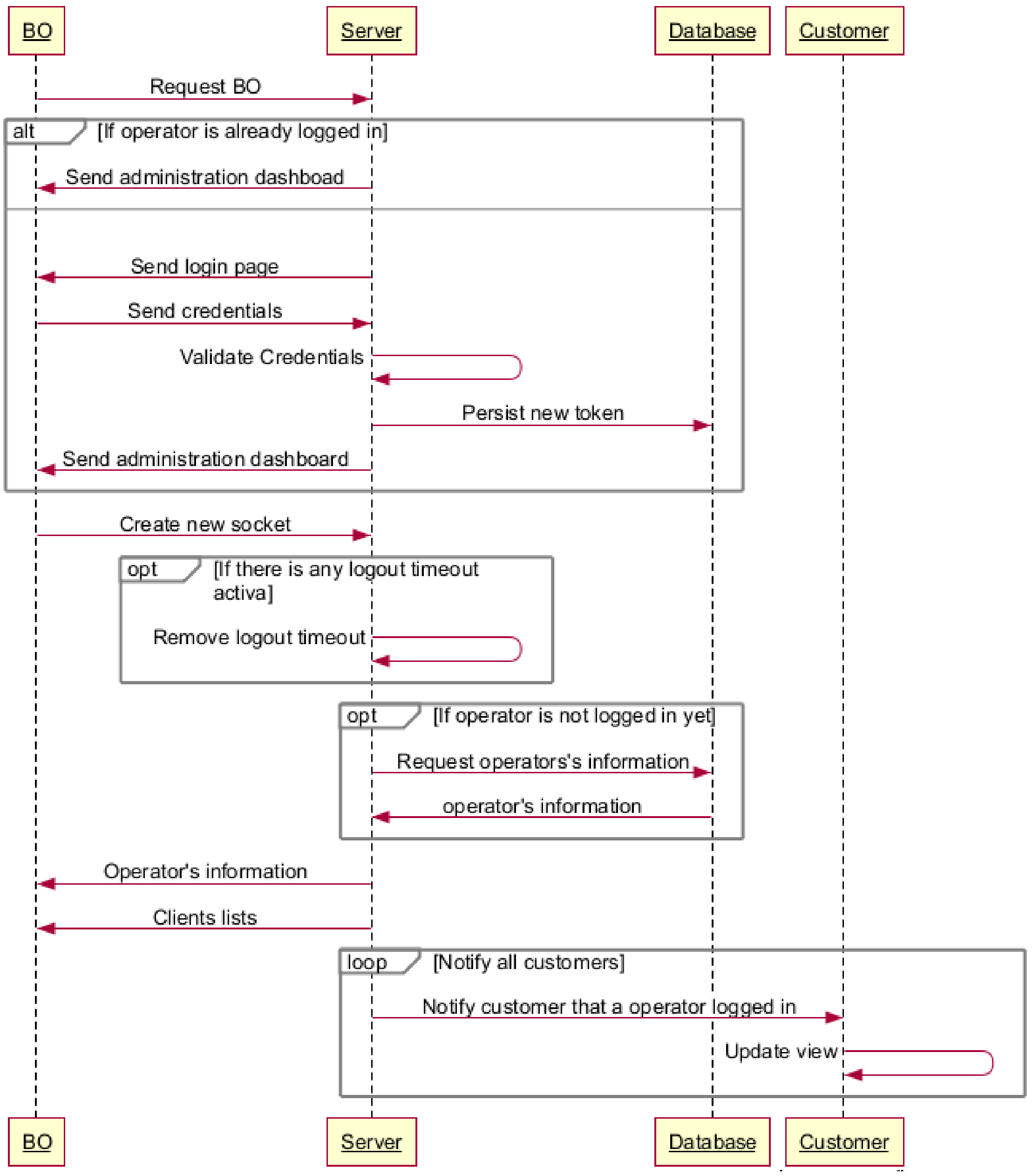


Figure 2.11. Login sequence diagram

The previous diagram illustrates the login flow.

The operator enters his credential at the login menu (Figure 2.8). When he clicks to send the request to the server, the application validates the fields and sends the information to the server, which will be validated to see if user exists and is valid. If the user is not valid an error is displayed, or the server will verify if the user is already logged in and generate a session token. This token will be used in future requests to authenticate the operator in the server and validate if the session is still valid.

After the user enters the administration page a new socket with the server is established. The operator's personal information and customers list is sent over. The socket will be used to exchange messages between operator and server bi-directionally in real time.

The last point is to notify the customers that an operator has entered the back office.

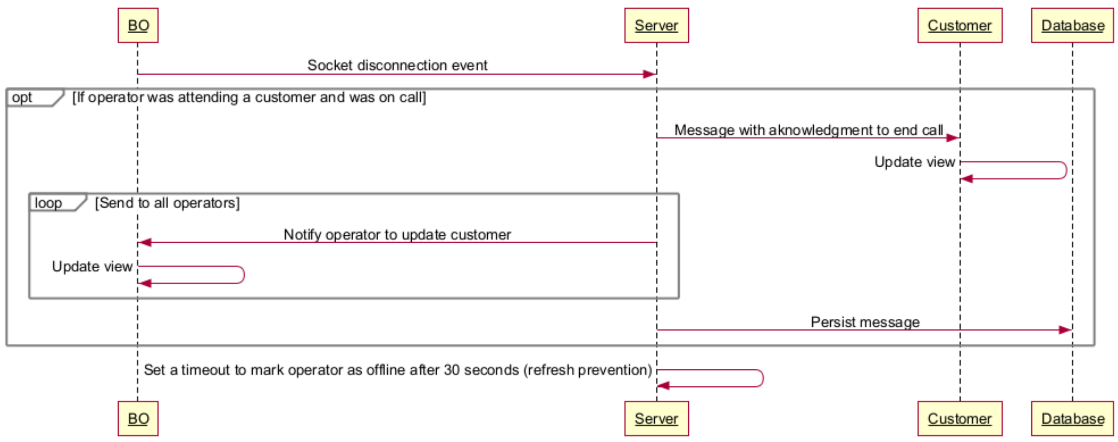

BO
Server
Customer
Database
Socket disconnection event
opt
[If operator was attending a customer and was on call]
Message with aknowledgment to end call
Update view
loop
[Send to all operators]
Notify operator to update customer
Update view
Persist message
Set a timeout to mark operator as offline after 30 seconds (refresh prevention)
BO
Server
Customer
Database

Figure 2.12. Operator's socket disconnection sequence diagram

The socket disconnection event has two meanings: the operator is refreshing the back office or the operator has left the back office. When this event occurs, ongoing calls are verified with the operator and they are finished and the customer is notified. Since the events are the same for both the refresh and leaving of the page, a timeout session is set. If the operator creates a new socket before the timeout fires, they equally are dismissed. If the timeout is triggered then the operator is seen as being offline and customers are notified that the operator has left.

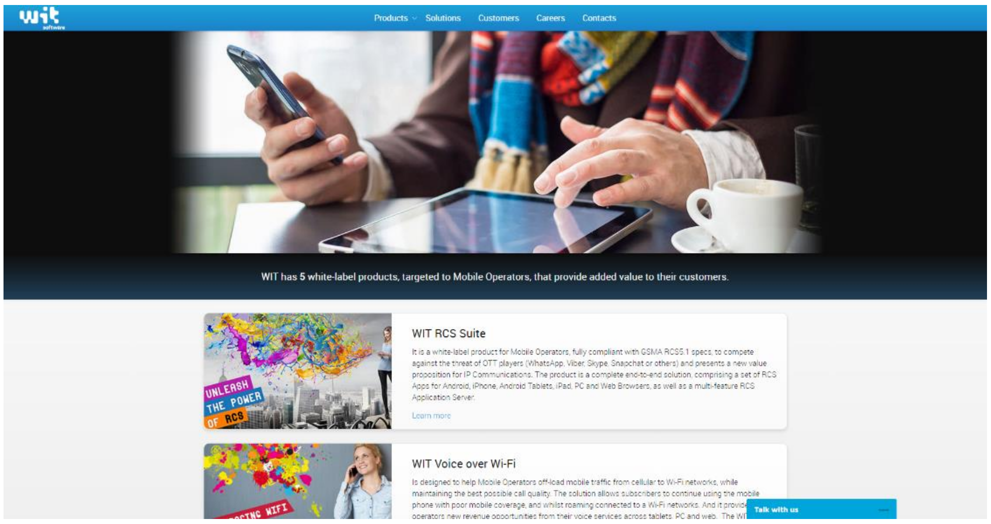
If the operator uses the logout feature he sends a request, and the server removes the operator from the online data store and destroy his session token. Additionally, the socket disconnection event is triggered and the previous actions are performed.

2.3 Customer's socket connection and disconnection

These were one of the most challenging features to implement due to its complexity.

To encourage customers to start a conversation it was required that a conversation could start without a previous login, Figure 2.14. It was also required to identify the customer that was sending the requests, in order to keep that track a Universal Unique Identifier (UUID) was given so that it appended to every request and was used to unequivocally identify a customer, since that UUID was passed as an HTTPOnly cookie that can only be altered on server.

The implementation challenge came from the requirement that required that a customer would be able to navigate the site on multiple tabs on the same browser. This brought problems, especially to differentiate when a customer left the site from when he refreshed or navigated through the browser. To overcome this challenge, the solution found was to create a timeout every time the last socket from a client broke, and clean the timeout if the customer reconnects to the site.



WIT has 5 white-label products, targeted to Mobile Operators, that provide added value to their customers.

WIT RCS Suite

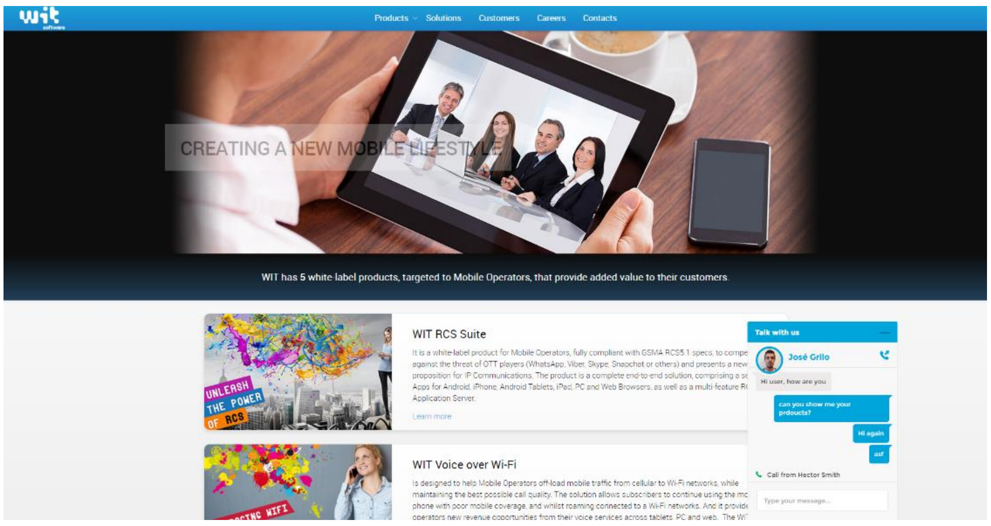
It is a white-label product for Mobile Operators, fully compliant with GSMA RCS5.1 specs, to compete against the threat of OTT players (WhatsApp, Viber, Skype, Snapchat or others) and presents a new value proposition for IP Communications. The product is a complete end-to-end solution, comprising a set of RCS Apps for Android, iPhone, Android Tablets, iPad, PC and Web Browsers, as well as a multi-feature RCS Application Server.

Learn more

WIT Voice over Wi-Fi

Is designed to help Mobile Operators off-load mobile traffic from cellular to Wi-Fi networks, while maintaining the best possible call quality. The solution allows subscribers to continue using the mobile phone with poor mobile coverage, and whilst roaming connected to a Wi-Fi networks. And it provide operators new revenue opportunities from their voice services across tablets, PC and web. The WI

Figure 2.13. Collapsed widget UI



WIT has 5 white-label products, targeted to Mobile Operators, that provide added value to their customers.

WIT RCS Suite

It is a white-label product for Mobile Operators, fully compliant with GSMA RCS5.1 specs, to compe against the threat of OTT players (WhatsApp, Viber, Skype, Snapchat or others) and presents a new proposition for IP Communications. The product is a complete end-to-end solution, comprising a se Apps for Android, iPhone, Android Tablets, iPad, PC and Web Browsers, as well as a multi-feature R Application Server.

Learn more

WIT Voice over Wi-Fi

Is designed to help Mobile Operators off-load mobile traffic from cellular to Wi-Fi networks, while maintaining the best possible call quality. The solution allows subscribers to continue using the mc phone with poor mobile coverage, and whilst roaming connected to a Wi-Fi networks. And it provide operators new revenue opportunities from their voice services across tablets, PC and web. The WI

Hi user, how are you

can you show me your prdoucts?

Hi again

asf

Call from Hector Smith

Figure 2.14. Expanded widget UI

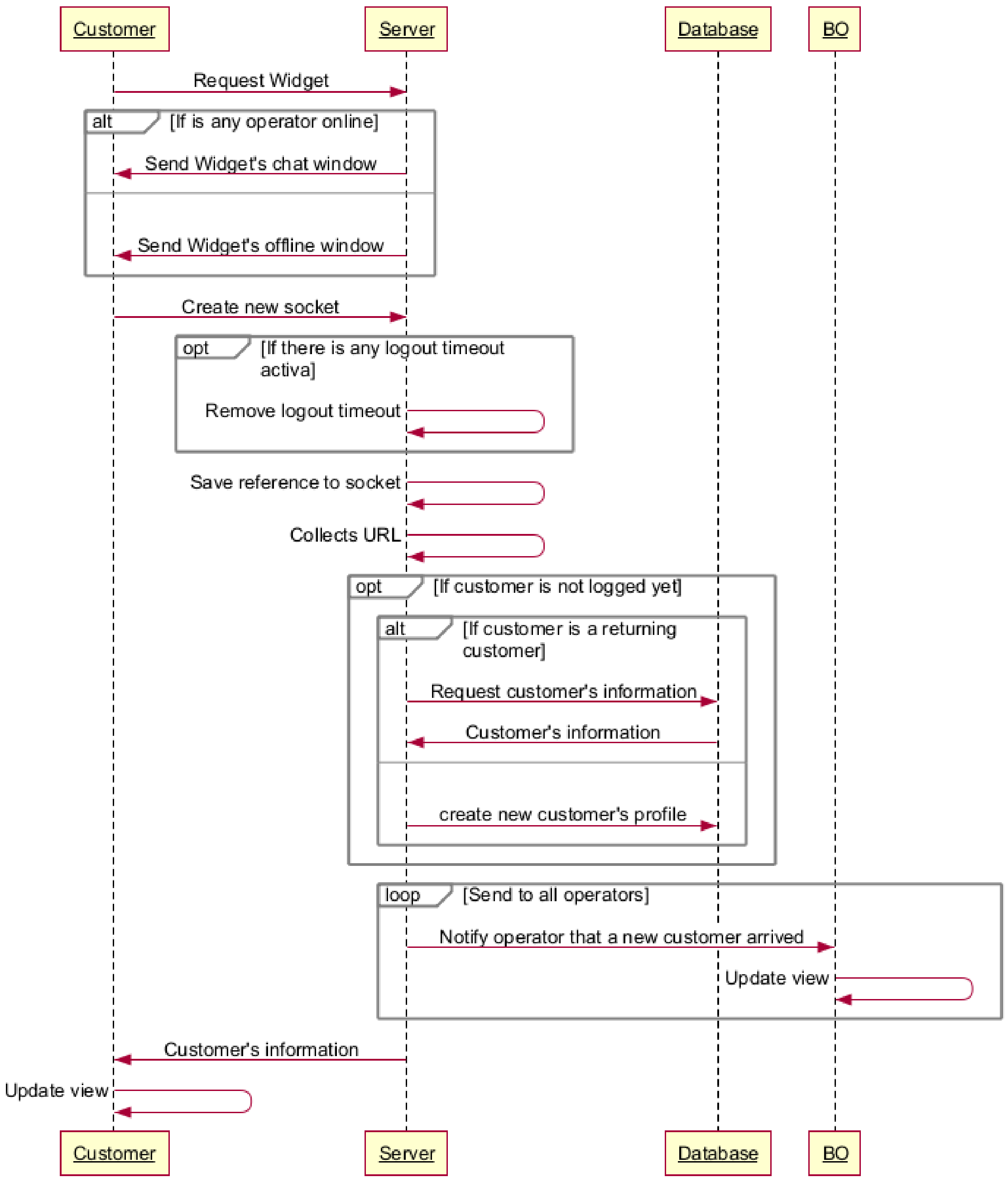

Customer
Server
Database
BO
Request Widget
alt
[If is any operator online]
Send Widget's chat window
Send Widget's offline window
Create new socket
opt
[If there is any logout timeout activa]
Remove logout timeout
Save reference to socket
Collects URL
opt
[If customer is not logged yet]
alt
[If customer is a returning customer]
Request customer's information
Customer's information
create new customer's profile
loop
[Send to all operators]
Notify operator that a new customer arrived
Update view
Customer's information
Update view
Customer
Server
Database
BO

Figure 2.15. Customer's socket connection sequence diagram

Unlike the typical chat system, in the product developed the customers don't need to input any personal information to start a chat. This decision was made to reduce the probability of a customer leaving the site because he was forced to share his name or email.

To use the product the customer only needs to access the site and the widget assists automatically. If operators are not available an offline window is displayed, otherwise an online window is shown. After, the socket is created and personal information is sent over. The socket is then used to send and receive chat messages, call events, and contextual information is collected.

Finally, the operators are notified that a costumer entered the site.

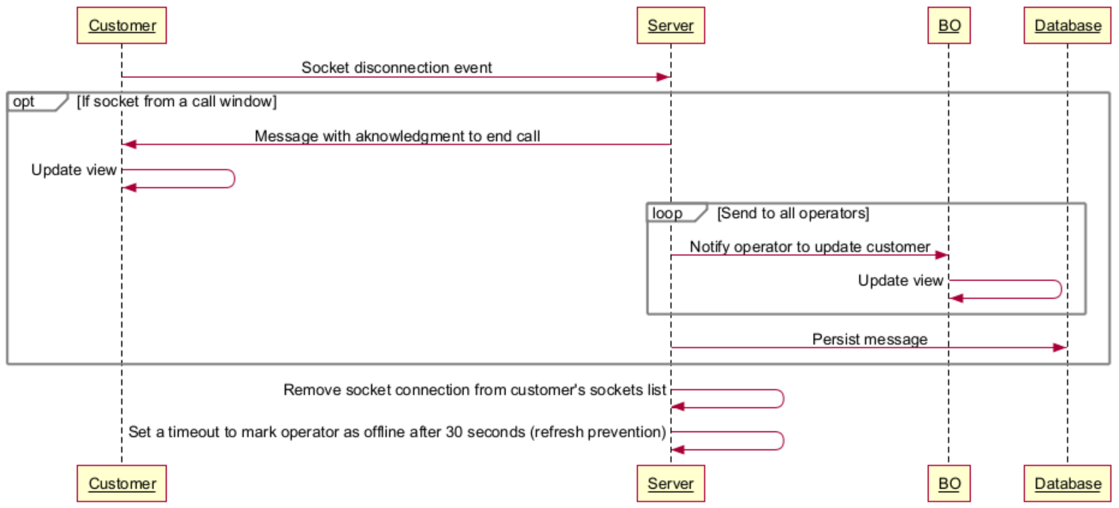

Customer
Server
BO
Database
Socket disconnection event
opt
[If socket from a call window]
Message with aknowledgment to end call
Update view
loop
[Send to all operators]
Notify operator to update customer
Update view
Persist message
Remove socket connection from customer's sockets list
Set a timeout to mark operator as offline after 30 seconds (refresh prevention)
Customer
Server
BO
Database

Figure 2.16. Customer's socket disconnection sequence diagram

Unlike the typical chat system, in the product developed the customers don't need to input any personal information to start a chat. This decision was made to reduce the probability of a customer leaving the site because he was forced to share his name or email.

To use the product the customer only needs to access the site and the widget assists automatically. If operators are not available an offline window is displayed, otherwise an online window is shown. After, the socket is created and personal information is sent over. The socket is then used to send and receive chat messages, call events, and contextual information is collected.

Finally, the operators are notified that a costumer entered the site.

2.4 Context Collection

The context collection feature refers to all the information that can be collected that can help operators understand the reasons behind the navigation and questions from the customer and provide better service.

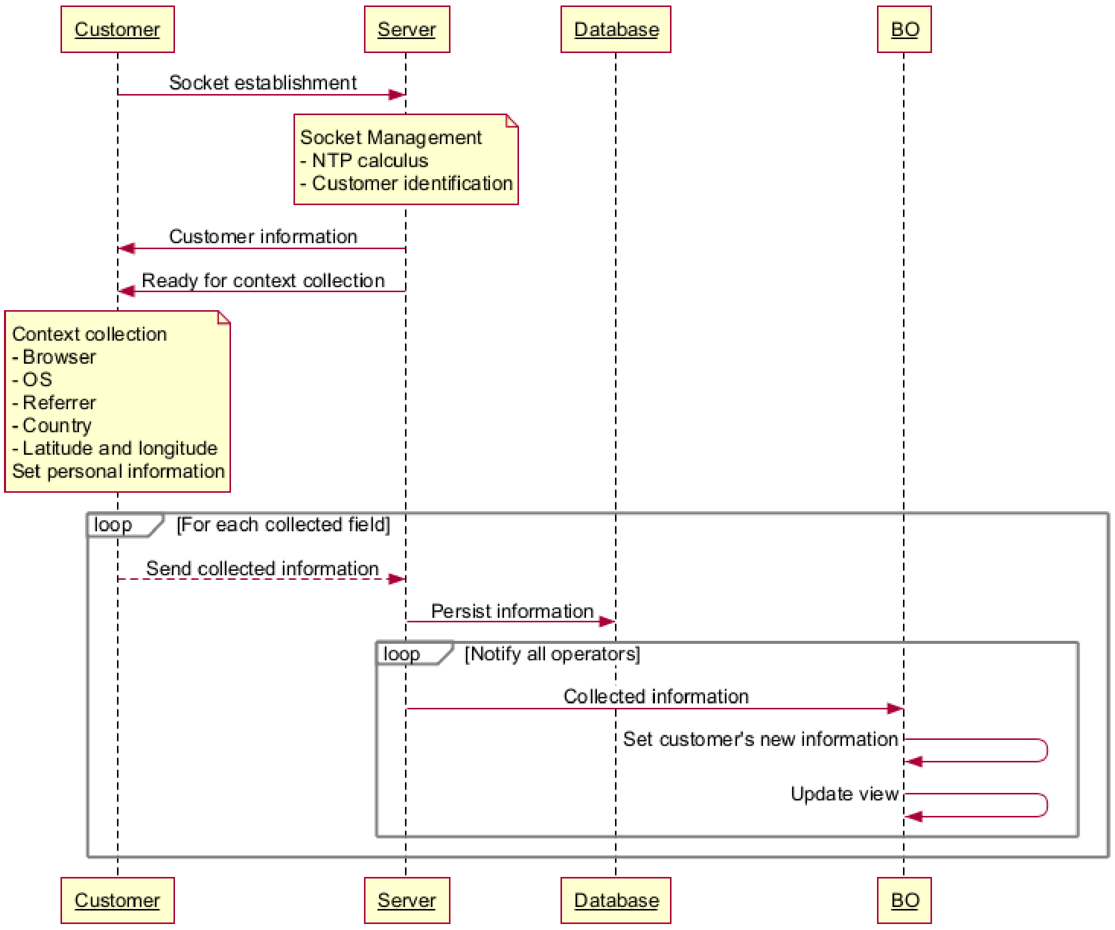


Figure 2.17. Context collection sequence diagram

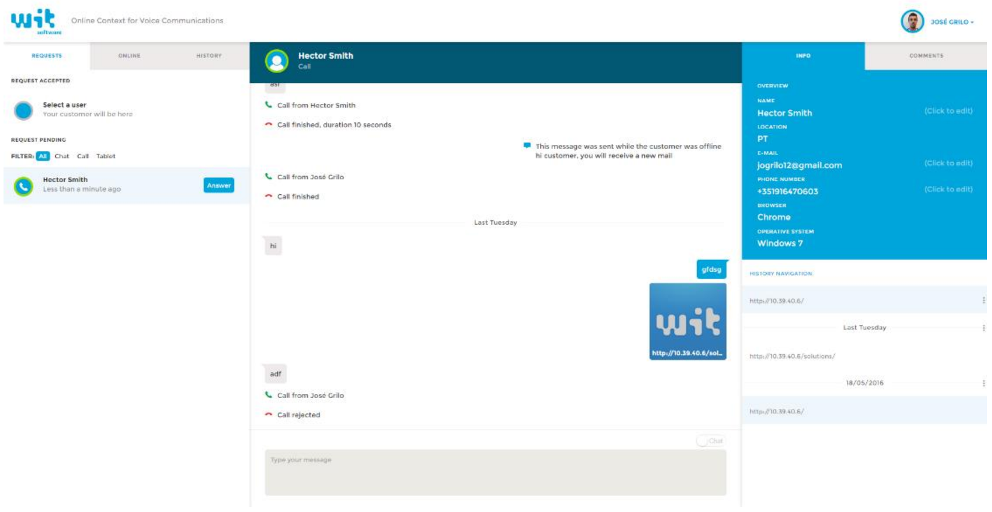
To collect both OS and browser's information, the *window.navigator.userAgent* property from the browser was used. Every time that the widget starts, it analyses and parses the string returned by the property, and returns the OS and browser's information, which are sent to the server via Socket.io.

To collect the country code it is called http://ipinfo.io [1] API that returns a JSON with several fields, among them is the country code.

The latitude and longitude values are obtained with the Geolocation API from the HTML5. This information is then translated on server side, using Google's Geocode API location.

For the visited URLs, every time a new Socket.io connection is created, its header is parsed. If the origin matches the server URL, the URL is ignored because it means that the connection is from an operator. Otherwise, the referrer field is parsed to get the URL where the customer sent his request.

After it has been parsed and validated on server side, the collected information is then sent to the operators, Figure 2.18, in real time in order to allow for easy assistance of the customers.


Online Context for Voice Communications

JOSÉ GRILO

REQUESTS | ONLINE | HISTORY

REQUEST ACCEPTED

Select a user
Your customer will be here

REQUEST PENDING

FILTER: All Chat Call Tablet

Hector Smith
Less than a minute ago
Answer

Hector Smith
Call

asf

Call from Hector Smith

Call finished, duration 10 seconds

This message was sent while the customer was offline
hi customer, you will receive a new mail

Call from José Grilo

Call finished

Last Tuesday

hi

gfdsg

http://10.39.40.6/sol...

adf

Call from José Grilo

Call rejected

Chat

Type your message

INFO | COMMENTS

OVERVIEW

NAME
Hector Smith (Click to edit)

LOCATION
PT

E-MAIL
jogrilo12@gmail.com (Click to edit)

PHONE NUMBER
+351916470603 (Click to edit)

BROWSER
Chrome

OPERATIVE SYSTEM
Windows 7

HISTORY NAVIGATION

http://10.39.40.6/

Last Tuesday

http://10.39.40.6/solutions/

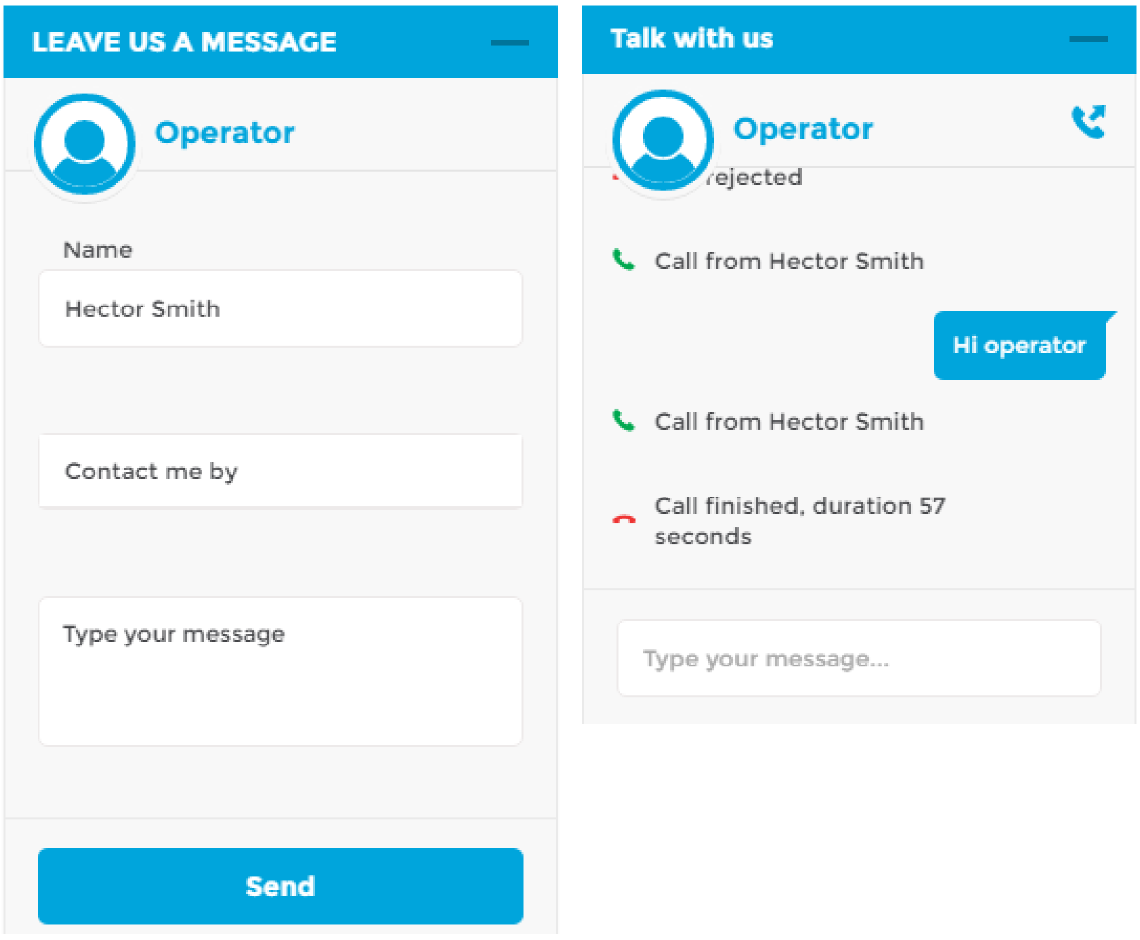
18/05/2016

http://10.39.40.6/

Figure 2.18. Back office with context collected from a user. On the right column, on top, is presented customer's personal information – name, email, location – and some information about the customer's device. At bottom is shows the navigation history by day from the customer's.

2.5 Start and End a conversation

There are two ways to start a conversation. The most common is when a customer makes a request to be contacted, either by chat, email or phone (Figure 2.19) and an operator opts to answer and contact him (Figure 2.20).


LEAVE US A MESSAGE

Operator

Name

Hector Smith

Contact me by

Type your message

Send

Talk with us

Operator

rejected

Call from Hector Smith

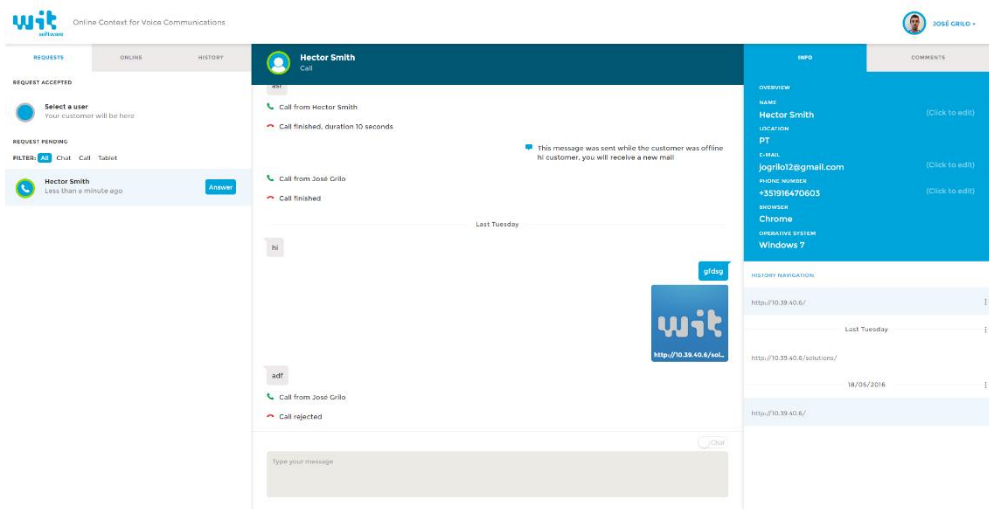
Hi operator

Call from Hector Smith

Call finished, duration 57 seconds

Type your message...

Figure 2.19. Customer's request. On the left, when there's no operators available the user can fill the form and select the way he prefers to be contacted. On the right side, to request a conversation the customer only needs to send a new message.



REQUEST ACCEPTED

Select a user
Your customer will be here

REQUEST PENDING

FILTER: All Chat Call Tablet

Hector Smith
Less than a minute ago

Hector Smith
Call

asi

Call from Hector Smith

Call finished, duration 10 seconds

This message was sent while the customer was offline
hi customer, you will receive a new mail

Call from José Grilo

Call finished

Last Tuesday

hi

gfdsg

http://10.39.40.6/sol...

adf

Call from José Grilo

Call rejected

OVERVIEW

NAME
Hector Smith (Click to edit)

LOCATION
PT

E-MAIL
jogrilo12@gmail.com (Click to edit)

PHONE NUMBER
+351916470603 (Click to edit)

BROWSER
Chrome

OPERATIVE SYSTEM
Windows 7

HISTORY NAVIGATION

http://10.39.40.6/

Last Tuesday

http://10.39.40.6/solutions/

18/05/2016

http://10.39.40.6/

Figure 2.20. Back office customer answering. To answer a customer that made a request the operator selects him on the left column and click on the "Answer" button.

The second way is when an operator chooses a customer from the online list and starts to talk to him without any previous request. This proactive contact can be used to get more insight on the customer's visit, to show the customer any special promotion or to force sales for instance.

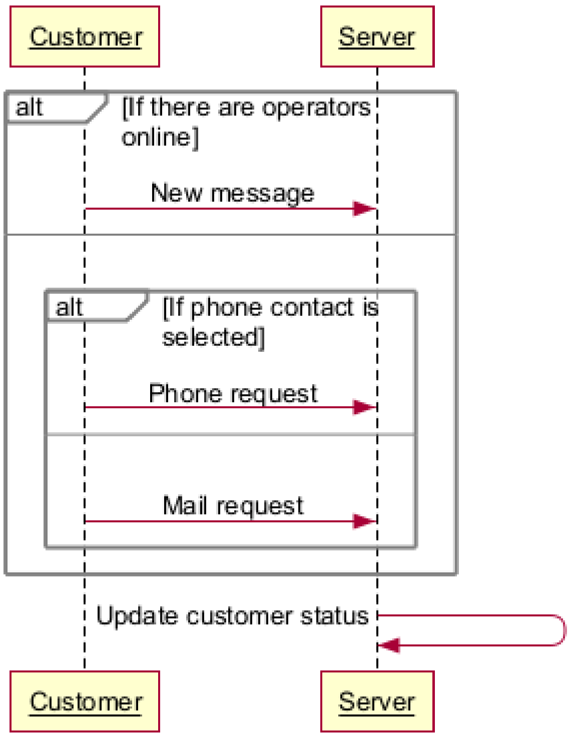


Figure 2.21. Customer request sequence diagram

There are two use cases to make a request to be contacted. Either there are operators ready and available to answer the customer or there are no available operators. In the first case, the customers can request to be contacted via chat just by sending a new message without the need to fill any forms. In the scenario where no operators are available to answer, then the customers need to fill out a small form where they can choose how to be contacted, either by email or phone.

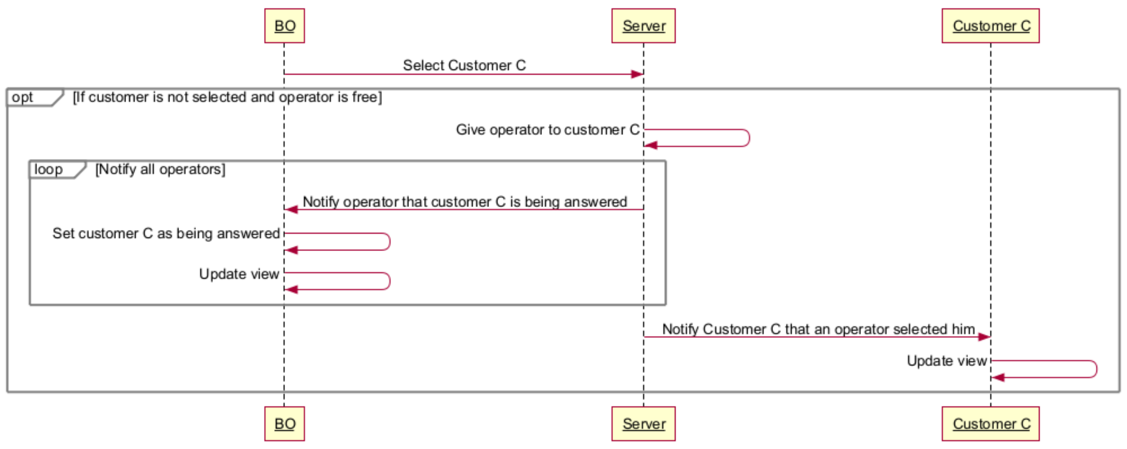

BO
Server
Customer C
Select Customer C
opt [If customer is not selected and operator is free]
Give operator to customer C
loop [Notify all operators]
Notify operator that customer C is being answered
Set customer C as being answered
Update view
Notify Customer C that an operator selected him
Update view

Figure 2.22. Customer answering sequence diagram

After receiving the request from a customer, the operator can choose whom to answer first. The selected customer will be moved from pending to answer currently and the customer will be notified about the operator that is answering him. After selecting a costumer to answer a new conversation can start.

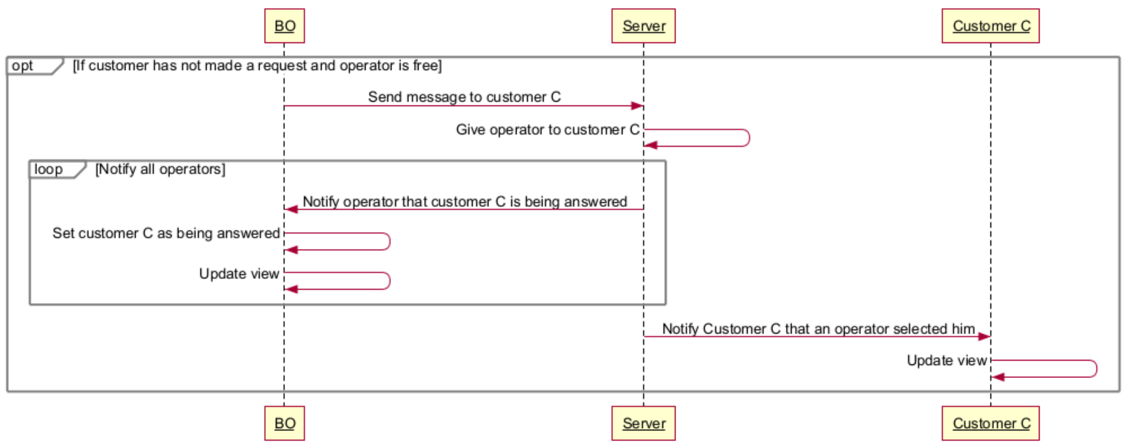

BO
Server
Customer C
opt [If customer has not made a request and operator is free]
Send message to customer C
Give operator to customer C
loop [Notify all operators]
Notify operator that customer C is being answered
Set customer C as being answered
Update view
Notify Customer C that an operator selected him
Update view

Figure 2.23. Proactive engagement sequence diagram

As previously stated, operators can start a conversation as well. In that case, they can send a message to an online customer that has not done a request yet and the customer is selected to be answered as if the operator have selected him to be answered (previous use case).

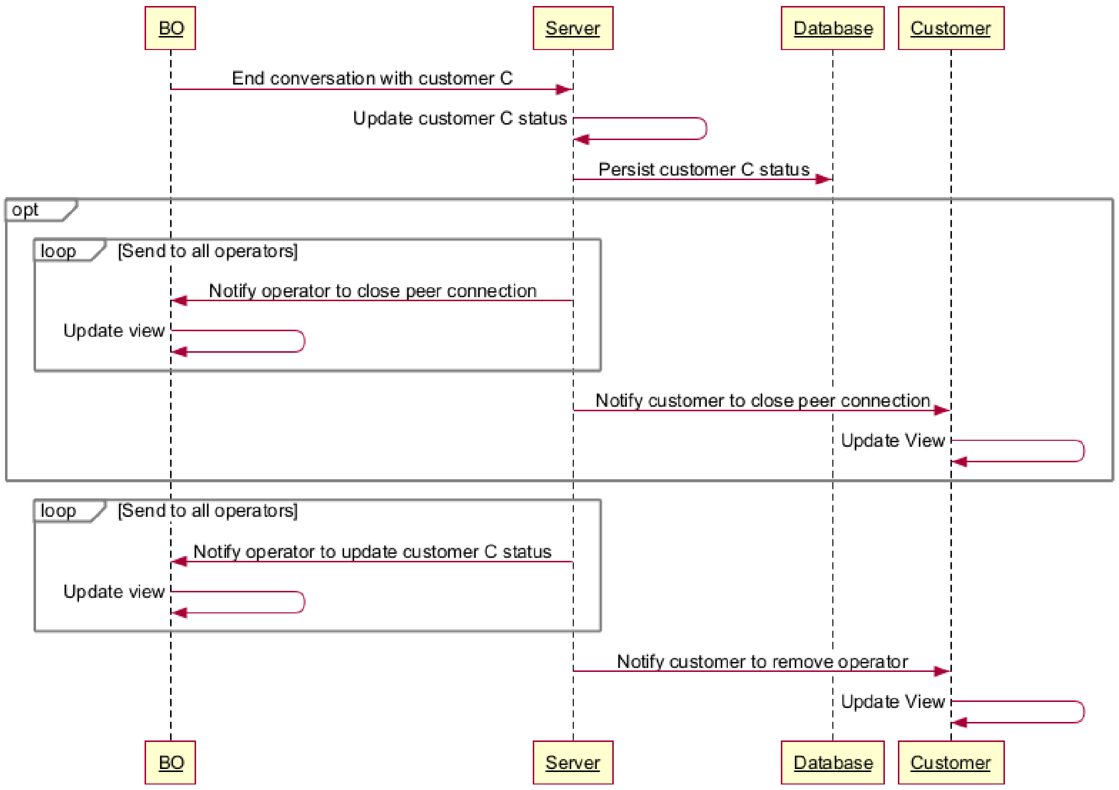
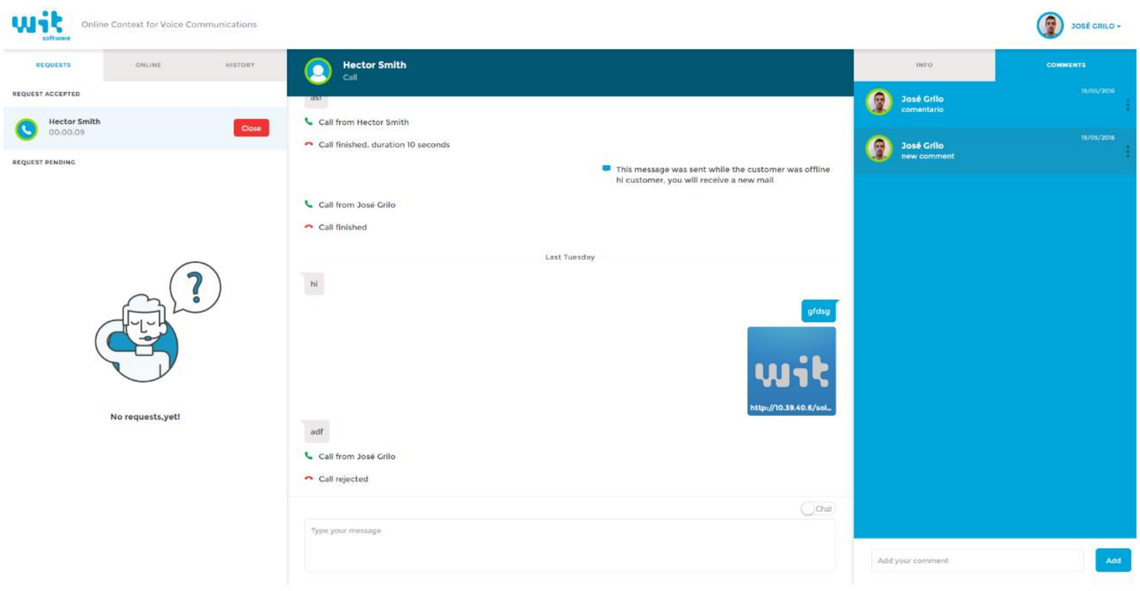

BO
Server
Database
Customer
End conversation with customer C
Update customer C status
Persist customer C status
opt
loop [Send to all operators]
Notify operator to close peer connection
Update view
Notify customer to close peer connection
Update View
loop [Send to all operators]
Notify operator to update customer C status
Update view
Notify customer to remove operator
Update View
BO
Server
Database
Customer

Figure 2.24. End conversation sequence diagram

Only operators can state the conversation as "closed" (Figure 2.25). When a conversation is closed, it is passed onto the history (Figure 2.26). Besides that, the customer is removed from the pending queue and is notified. When a conversation is closed the customer has the option to get the conversation over email if he wants to.



REQUEST ACCEPTED

Hector Smith
00:00:09

REQUEST PENDING

No requests,yet!

Hector Smith
Call

Call from Hector Smith

Call finished, duration 10 seconds

This message was sent while the customer was offline
hi customer, you will receive a new mail

Call from José Grilo

Call finished

Last Tuesday

hi

gfdsg

http://10.39.40.6/sol...

adf

Call from José Grilo

Call rejected

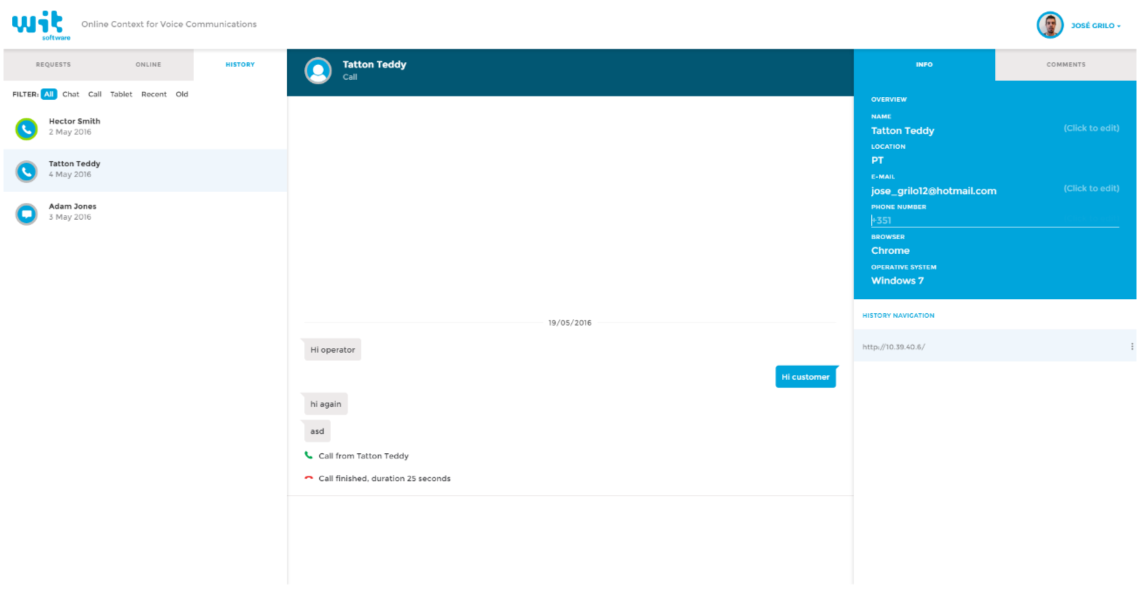
Type your message

José Grilo
comentario
19/05/2016

José Grilo
new comment
19/05/2016

Add your comment

Figure 2.25. Back office conversation ending. To close a conversation with a customer operators select him on the left column and clicks on "Close" button.


Online Context for Voice Communications
JOSÉ GRILO
REQUESTS
ONLINE
HISTORY
FILTER: All Chat Call Tablet Recent Old
INFO
COMMENTS

Hector Smith
2 May 2016

Tatton Teddy
4 May 2016

Adam Jones
3 May 2016

Tatton Teddy
Call

19/05/2016

Hi operator

Hi customer

hi again

asd

Call from Tatton Teddy

Call finished, duration 25 seconds

OVERVIEW

NAME
Tatton Teddy (Click to edit)

LOCATION
PT

E-MAIL
jose_grilo12@hotmail.com (Click to edit)

PHONE NUMBER
+351

BROWSER
Chrome

OPERATIVE SYSTEM
Windows 7

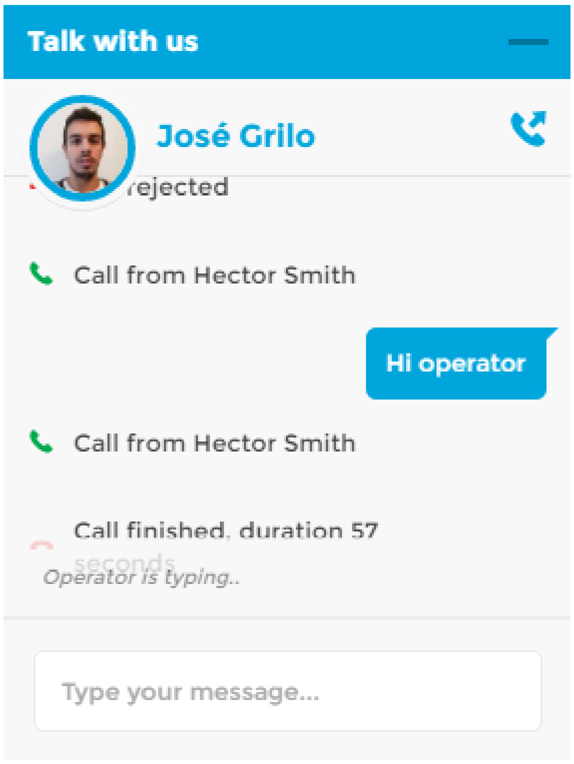
HISTORY NAVIGATION

http://10.39.40.6/

Figure 2.26. Back office history list

2.6 Chat

After a conversation starts any of the actors can contact the other with instant messaging (figures 27 and 28).



José Grilo

rejected

Call from Hector Smith

Hi operator

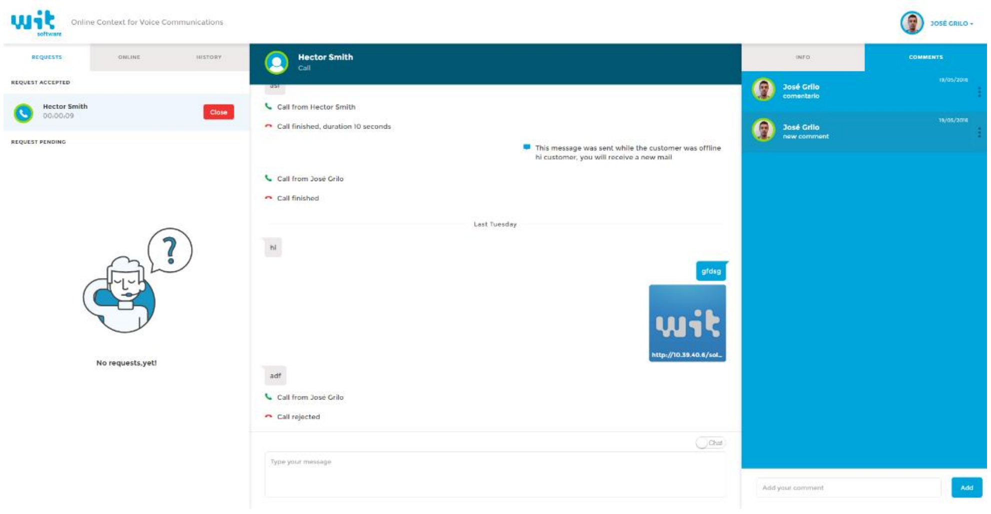
Call from Hector Smith

Call finished, duration 57 seconds

Operator is typing..

Type your message...

Figure 2.27. Customer on conversation with an operator



REQUEST ACCEPTED

Hector Smith
00:00:09

REQUEST PENDING

No requests,yet!

Call from Hector Smith

Call finished, duration 10 seconds

This message was sent while the customer was offline
hi customer, you will receive a new mail

Call from José Grilo

Call finished

Last Tuesday

hi

gfdsg

http://10.39.40.6/sol...

adf

Call from José Grilo

Call rejected

Type your message

José Grilo 19/05/2018
comentario

José Grilo 19/05/2018
new comment

Add your comment

Figure 2.28. Operator on conversation with a customer

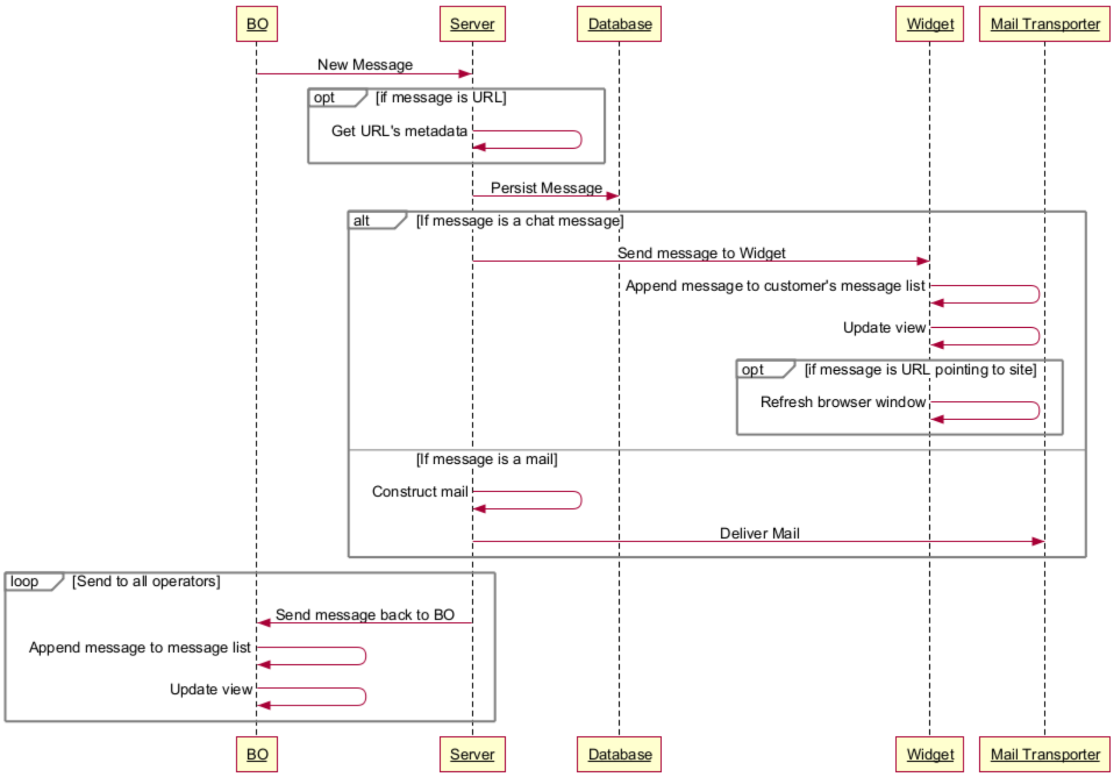


Figure 2.29. Operator's message sequence diagram

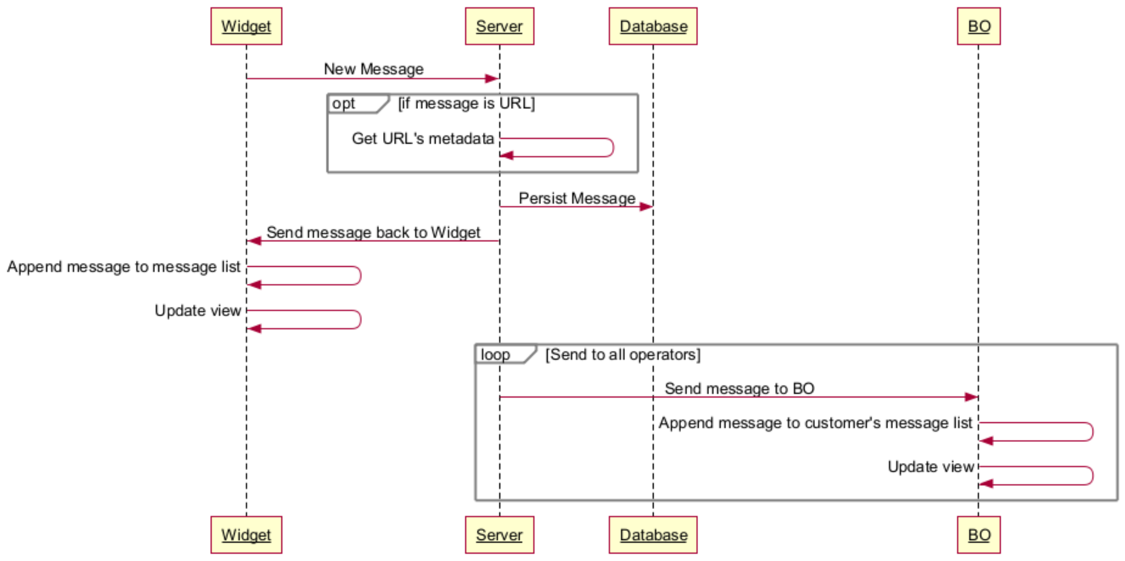


Figure 2.30. Customer's message sequence diagram

The chat flow is very similar for both actors. The message is redirected by the server to the other actor. All messages are sent to all operators. This allows all operators to follow the conversations in real time even if they are not answering the customer.

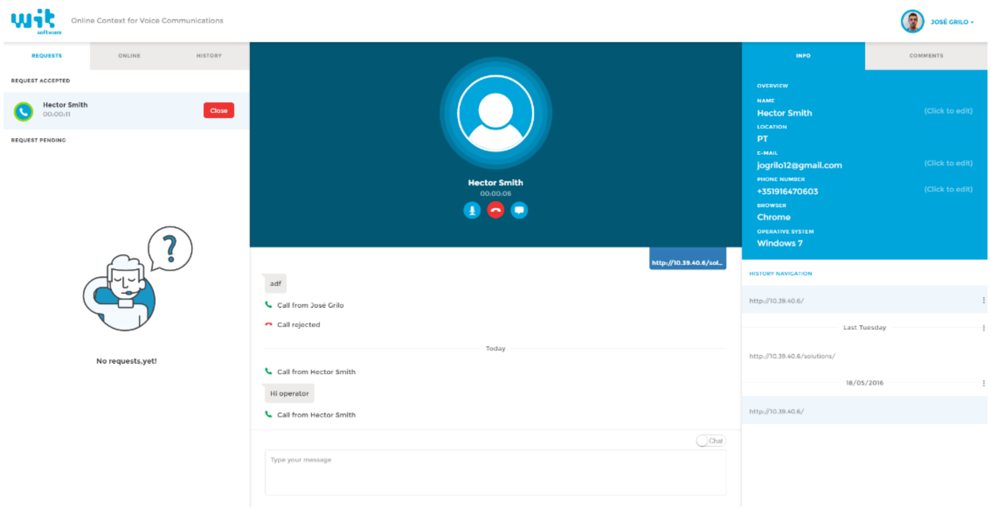
The major differences between customer and operator messaging is the way that messages are processed when received by the widget. A message from an operator received by the widget it is processed looking for a URL. If it contains an URL and points to the site domain, the customer is redirected to that location.

2.7 Calls

Another way to contact the customer is through a call, either by browser or device.

Browser calls were fully developed by the intern and breakout calls were developed using WWC gateway.

In the first case, the customer creates a call request that is sent to the operator through the server. Once the operator accepts, the customer makes the call request, this is sent to the operator through the server. When the server accepts the request, the client creates an offer which is then sent to the operator which in turn responds with an answer and the connection is created (Figure 2.31 and Figure 2.32). On the other hand, the operator starts a breakout call. The flow is the same as the browser call, however the WWC gateway makes the data transformation in order to pass the requests to their receivers (Figure 2.33 and Figure 2.34).



REQUEST ACCEPTED

Hector Smith 00:00:11

REQUEST PENDING

No requests,yet!

Hector Smith
00:00:06

http://10.39.40.6/sol...

adf

Call from José Grilo

Call rejected

Today

Call from Hector Smith

Hi operator

Call from Hector Smith

Type your message

OVERVIEW

NAME
Hector Smith (Click to edit)

LOCATION
PT

E-MAIL
jogrilo12@gmail.com (Click to edit)

PHONE NUMBER
+351916470603 (Click to edit)

BROWSER
Chrome

OPERATIVE SYSTEM
Windows 7

HISTORY NAVIGATION

http://10.39.40.6/

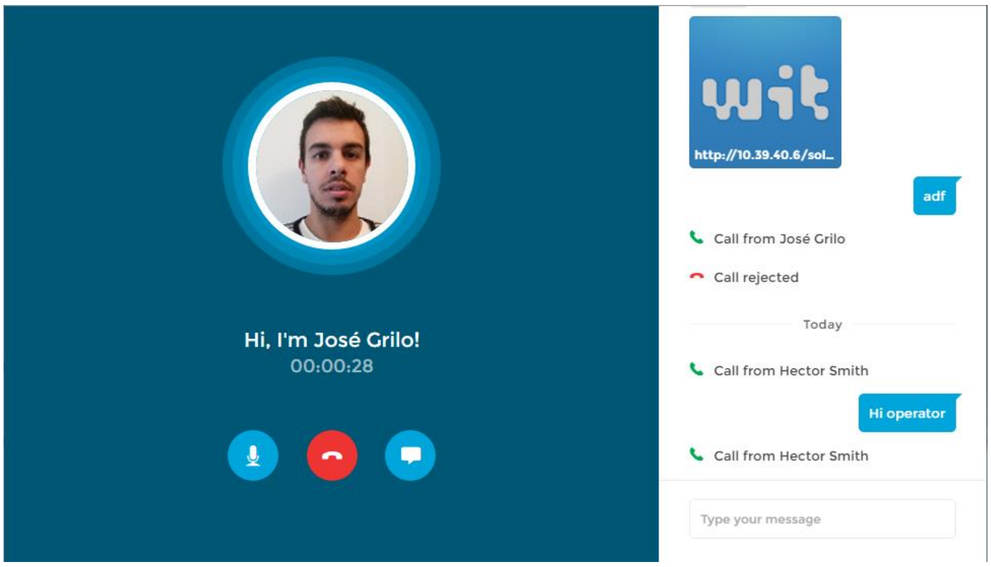
Last Tuesday

http://10.39.40.6/solutions/

18/05/2016

http://10.39.40.6/

Figure 2.31. Operator on call with a customer



Hi, I'm José Grilo!
00:00:28

http://10.39.40.6/sol...

adf

Call from José Grilo

Call rejected

Today

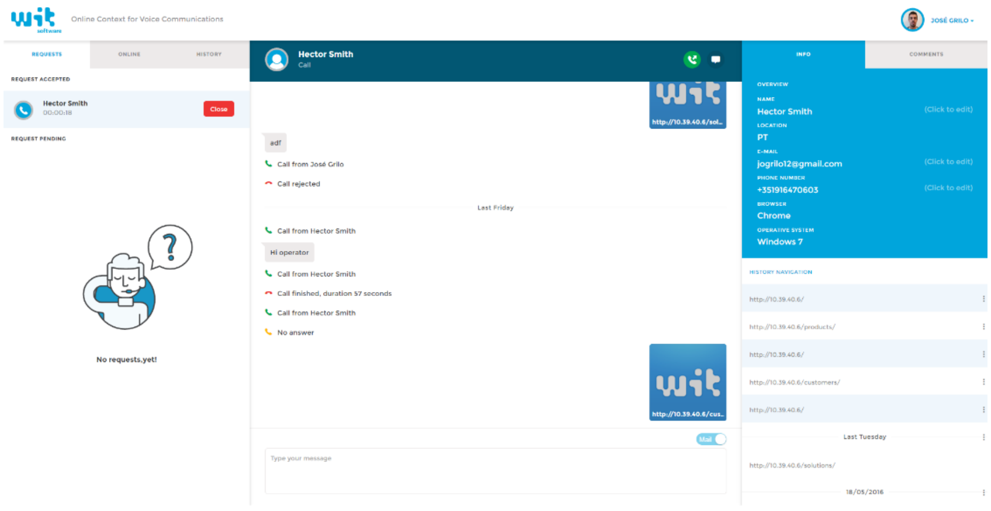
Call from Hector Smith

Hi operator

Call from Hector Smith

Type your message

Figure 2.32. Customer on call with an operator



REQUEST ACCEPTED

Hector Smith 00:00:18

REQUEST PENDING

No requests,yet!

Hector Smith
Call

http://10.39.40.6/sol...

adf

Call from José Grilo

Call rejected

Last Friday

Call from Hector Smith

Hi operator

Call from Hector Smith

Call finished, duration 57 seconds

Call from Hector Smith

No answer

http://10.39.40.6/cus...

Type your message

OVERVIEW

NAME
Hector Smith (Click to edit)

LOCATION
PT

E-MAIL
jogrilo12@gmail.com (Click to edit)

PHONE NUMBER
+351916470603 (Click to edit)

BROWSER
Chrome

OPERATIVE SYSTEM
Windows 7

HISTORY NAVIGATION

http://10.39.40.6/

http://10.39.40.6/products/

http://10.39.40.6/

http://10.39.40.6/customers/

http://10.39.40.6/

Last Tuesday

http://10.39.40.6/solutions/

18/05/2016

Figure 2.33. Operator ready to start a call to customer's device.

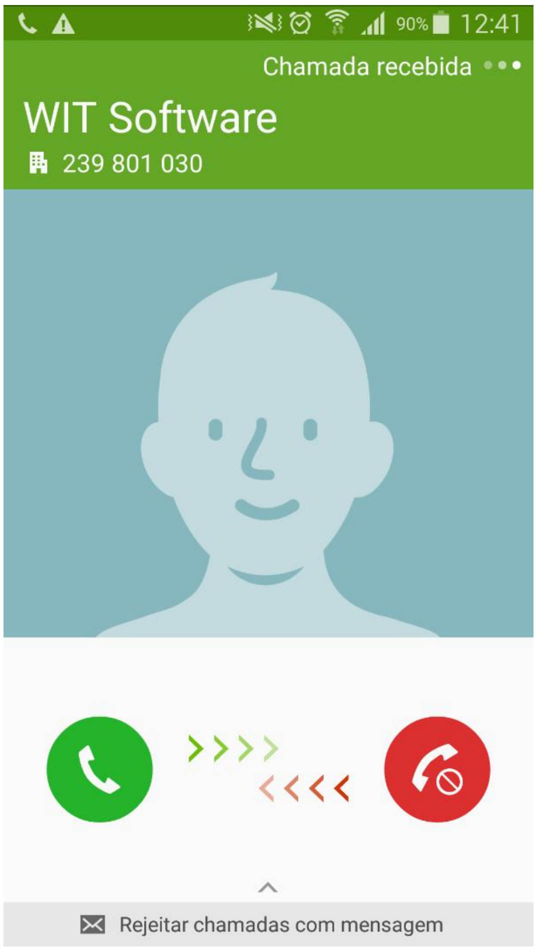


Figure 2.34. Customer receiving call from operator

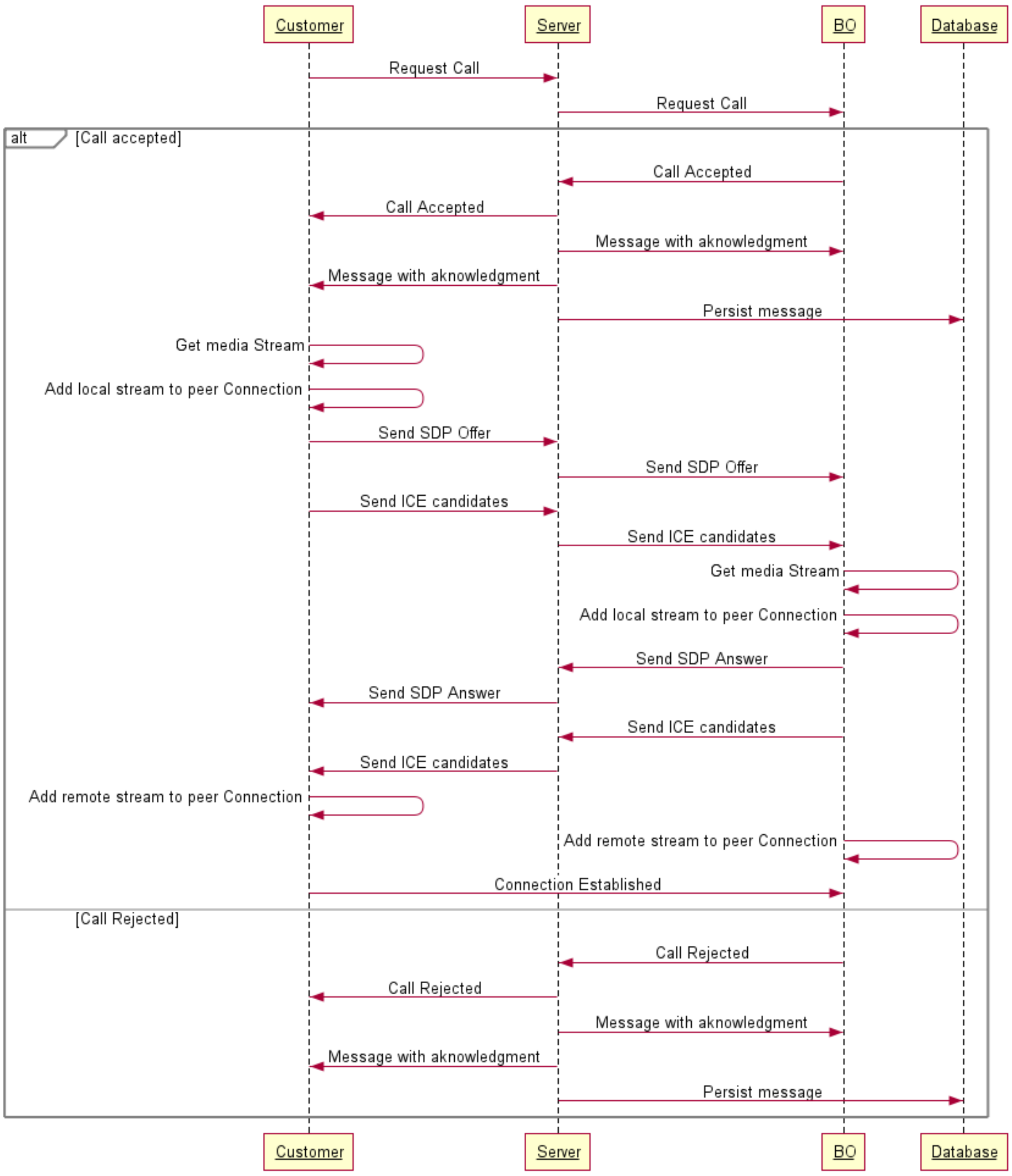


Figure 2.35. Browser call sequence diagram

The sequence presented above describes the browser call sequence. The call is always asked by the customer and the operators needs to accept it, in order to start the call. After all the signaling the actors create a peer connection, gets access to media and an exchange session descriptions – both the customer and operator generate a session description – and Interactive Connectivity Establishment (ICE) candidates. This exchange is asynchronous.

Once both actors have each other's stream a new peer to peer connection is established and the call starts.

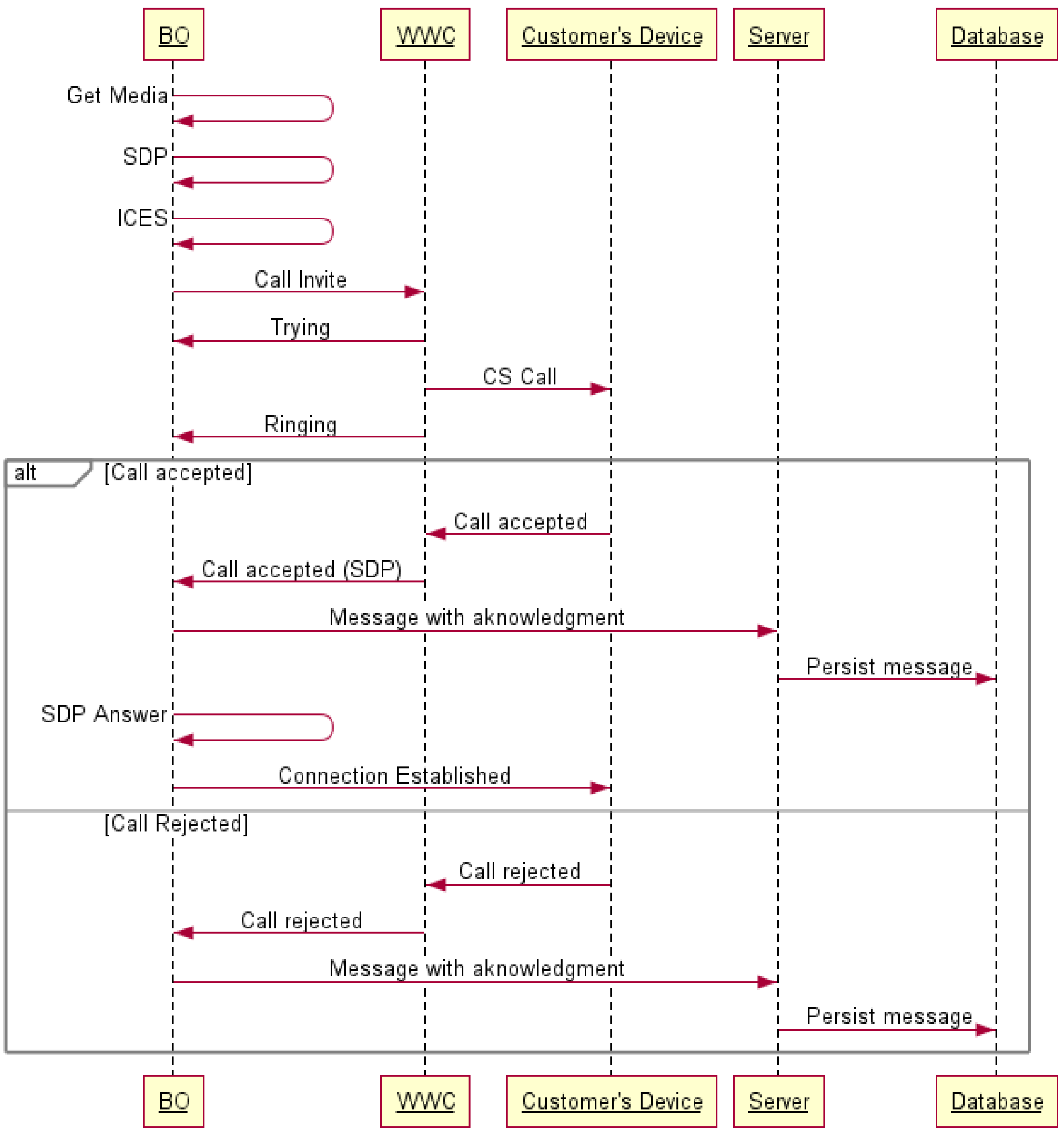


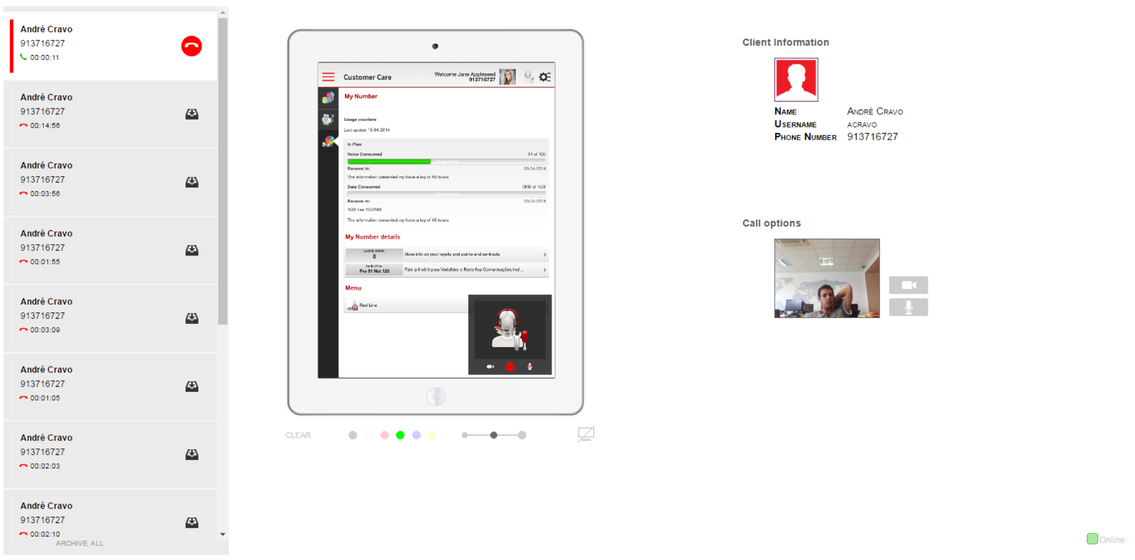
Figure 2.36. Breakout call sequence diagram

Breakout calls and browser calls have a very similar flow, however WWC gateway serves as the middle man to manage and translate packages from the browser to the device.

WWC SDK was used to communicate with WWC gateway. This SDK offers a communication API to make requests and handle responses. After login, WWC SDK registers at WWC using a previously configured account and an API key. After that, the SDK is ready to make requests for the calls. These requests act as a browser call request – SDP and ICE candidates are exchanged, and WWC gateway translates the requests to be interpreted by both the device and browser. Answers are served as events and WWC SDK offer callback functions to handle those events.

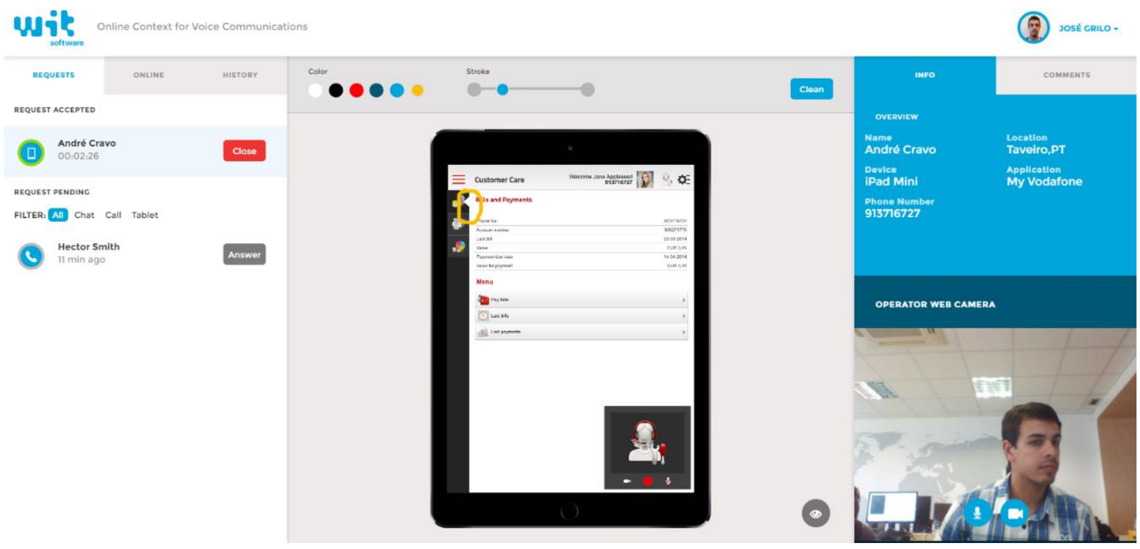
2.8 Remote Assistant

Calls received from the remote assistant app were a different issue. Since this demo was already part of WIT's portfolio it was not an implementation challenge, but rather an integration challenge. The architecture was already flexible enough to integrate with remote assistant server, however some changes needed to be made. Since the communication between customer and server was done by HTTP, the OCVC server was used as proxy so a new service was implemented inside the Chat module. The remote assistant back office (Figure 2.37) was also redone to meet Angular and UI standards (Figure 2.38).



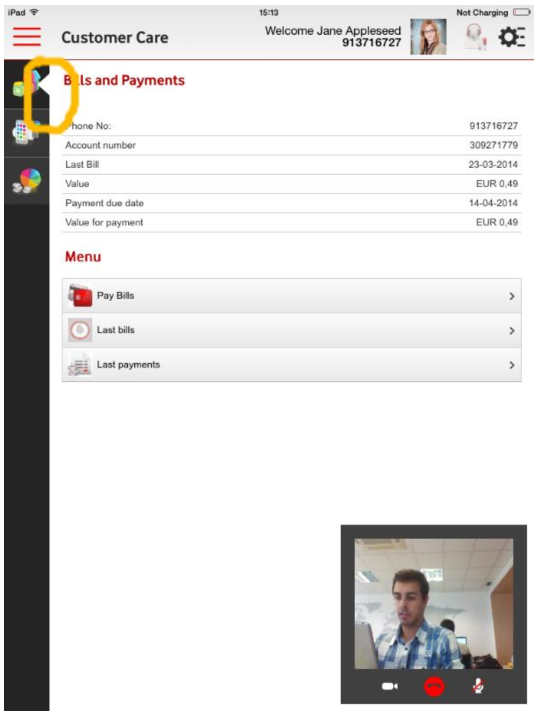
Name André Cravo
Username acravo
Phone Number 913716727

Figure 2.37. Remote assistant's back office old UI



Name André Cravo
Location Taveiro,PT
Device iPad Mini
Application My Vodafone
Phone Number 913716727

Figure 2.38. Remote assistant's back office new UI



iPad 15:13 Not Charging

Customer Care

Welcome Jane Appleseed
913716727

B ls and Payments

hone No:	913716727
Account number	309271779
Last Bill	23-03-2014
Value	EUR 0,49
Payment due date	14-04-2014
Value for payment	EUR 0,49

Menu

- Pay Bills
- Last bills
- Last payments

Figure 2.39. Remote assistant app on call with operator

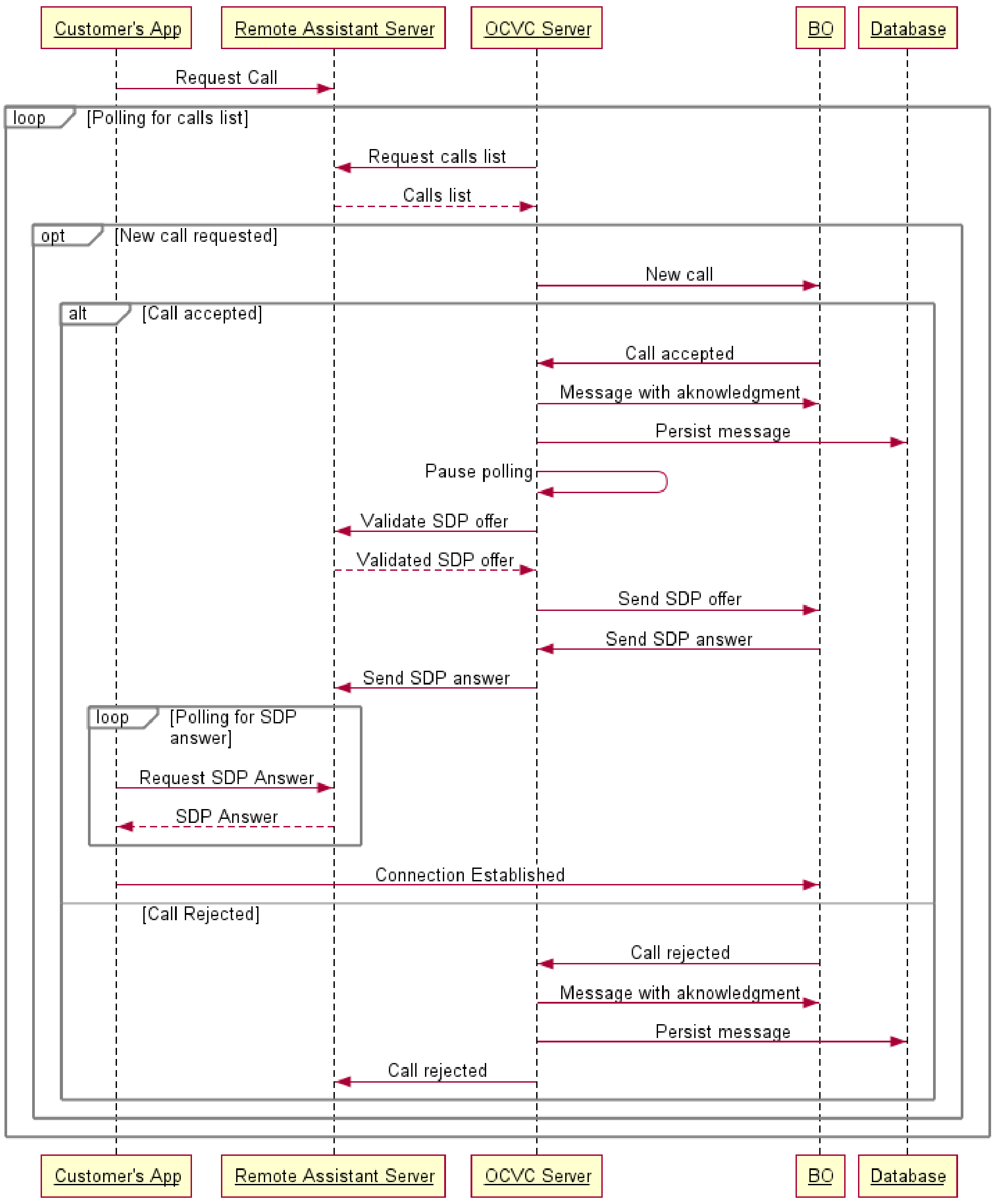


Figure 2.40. App call sequence diagram

As in breakout calls, there is a server in the middle between browser and device, but in this case server implements a REST API which is called every second (polling) to check for call updates.

All communication between actors was processed by OCVC server, which served as proxy due to Remote Assistant HTTP implementation. Requests are posted by the device on Remote Assistant server. OCVC is polling information every second until a change occurs. At this moment the offer is read and an answer is produced and posted. As previously stated, the device is also polling for changes and when it find an answer to its offer a connection is created.

The challenge here was in terms of integration. Besides knowing that there are better communication forms instead of polling (ex.: websockets or long polling) it was opted to not change the Remote Assistant server due to the complexity and time constraints.

The biggest challenge was to handle the communication between BO and the device. On the existing remote assistant demo the communication was by HTTP. On OCVC implementation, BO is served by HTTPS, so HTTP communications are not allowed due to Same Origin Policy. The solution found was to put OCVC server as a proxy, so the Remote Assistant communication lib presented on BO was merged to OCVC server.

2.9 Other features

Besides the previous features, some minor features were added to the project in order to complement it and help the operators keep track of the customers.

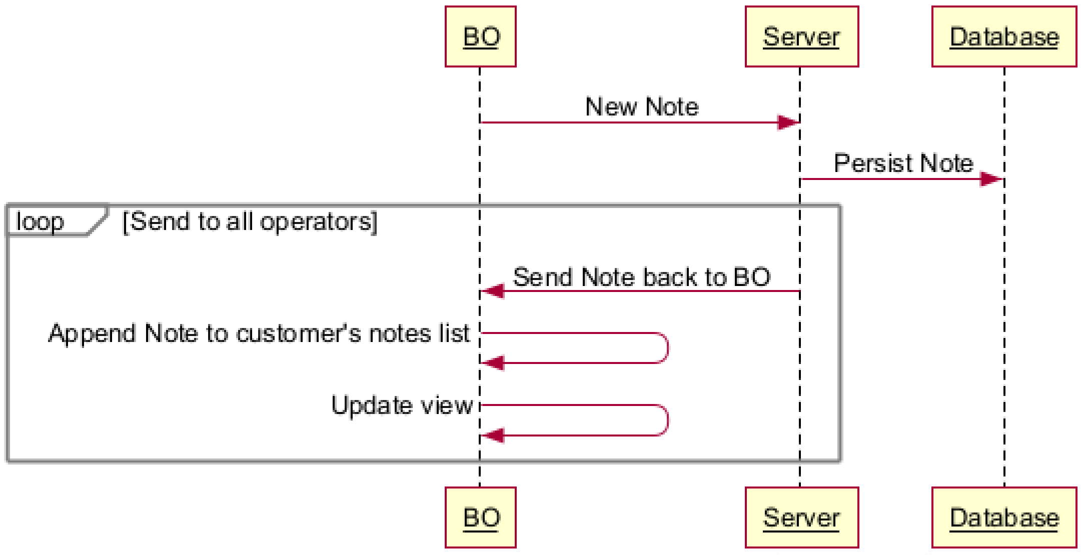


Figure 2.41. Add comment sequence diagram

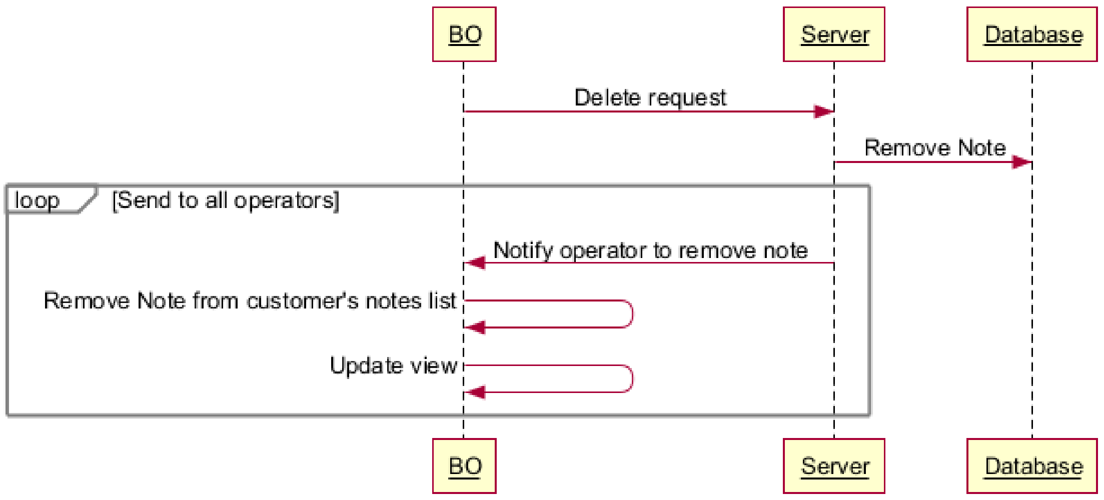
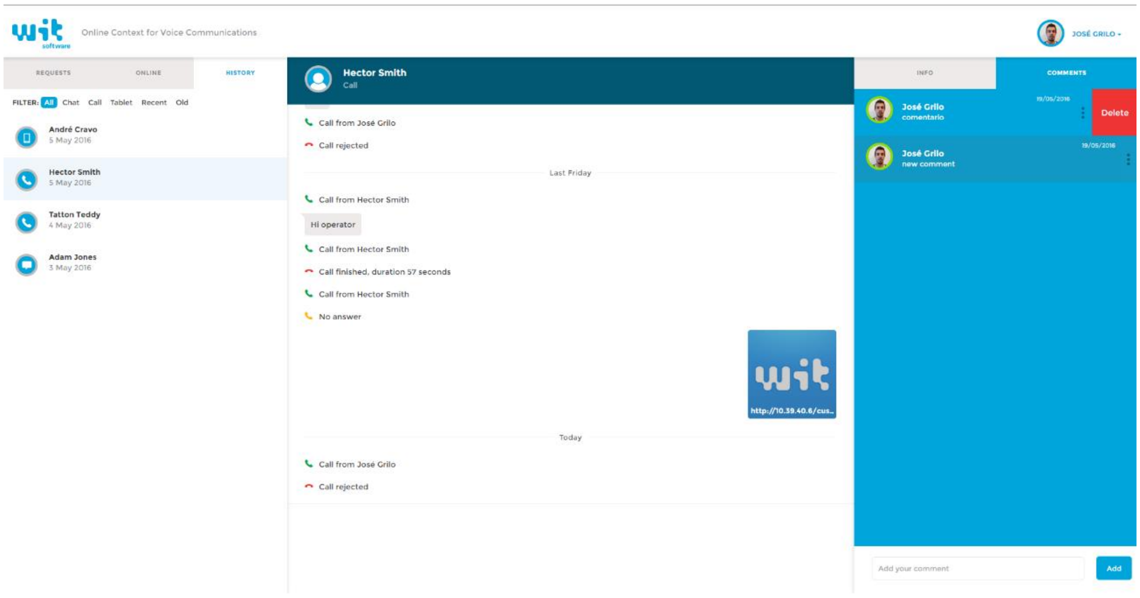


Figure 2.42. Remove comment sequence diagram

To add a new comment, its content is sent to the server, then it continues and notifies the operators that a comment was added to a customer. The same applies to remove a comment, where the server removes in from the database and notifies the operators about the action.


Online Context for Voice Communications
JOSÉ GRILO
REQUESTS
ONLINE
HISTORY
FILTER: All Chat Call Tablet Recent Old
INFO
COMMENTS
Delete
Add your comment
Add

André Cravo
5 May 2016

Hector Smith
5 May 2016

Tatton Teddy
4 May 2016

Adam Jones
3 May 2016

Hector Smith
Call

Call from José Grilo
Call rejected

Last Friday

Call from Hector Smith
Hi operator
Call from Hector Smith
Call finished, duration 57 seconds
Call from Hector Smith
No answer
http://10.39.40.6/cus...

Today

Call from José Grilo
Call rejected

José Grilo 19/05/2016
comentario

José Grilo 19/05/2016
new comment

Figure 2.43. Operator removing a comment. On the right column, if operator hovers on comment a "Delete" button appears which allow him to delete the comment. On the same column, at the bottom new comments can be added.

3 Tests

Testing the product was an important part of the project. Since an agile methodology was used, testing was a continuous process which means that at the end of each sprint the implemented features and integration were fully tested by the ones presented at the sprint meeting.

This testing methodology allowed for a faster way of tracking and fixing bugs during the development, while at the same time that a new increment was created.

Presented in this section, is a set of tests used to test the features and the results obtained.

3.1 Set of Tests

Sets of tests were created to test all features to ensure the project's product quality, reliability and good experience.

Functional tests served to prove the quality and reliability of the application. These allowed to find unexpected behaviors or potential crashes on unexpected situations. The requirements were used to produce the set of tests created. Features were grouped in five sets: authentication, context collection, back office features, widget features and call window features. Since most of the features produce a visual response it is important to guarantee that the developed features produce the expected results.

Functional tests follow WIT's Test Guidelines. For every test produced a unique name was given with a description, conditions to test, requisites and steps needed. Each test was classified in terms of importance in a scale from Low to High – a low important test is a test that verifies a feature that is not very relevant to the product functionality, a high important test is test that verifies a feature that is critical to the product functionality.

The sets of tests produced are as an appendix due to its extensiveness. To see the sets produced please refer to subsection **6.1 – Test Sets**.

In spite of not having a set of tests to test the usability, there was a constant topic discussion over the project. As stated on subsection 2.1 the UI suffered great changes over the project result of a continuous improvement to the user experience. Everyone involved in this discussion actively participated.

The main reason why there were no usability tests was due to the fact that this product is still under development. Since it will be used as POC and a demonstration product, if any potential interested appears the UI must be rethought to fulfill the interested needs.

3.2 Results

At the end of each sprint the corresponding tests were run in order to validate the feature and the results were used to fix bugs.

After the features were implemented from the backlog, sets of tests were run.

The results were:

Test	Result
Authentication Set	
Test Case ASTC-OCVC-0100: Login – Registered operator	✓
Test Case ASTC-OCVC-0200: Login – Unregistered operator	✓

Test Case ASTC-OCVC-0300: Login – Auto Login	✓
Test Case ASTC-OCVC-0400: Logout	✓
Context collection set	
Test Case CSTC-OCVC-0100: Context collection – URL	✓
Test Case CSTC-OCVC-0200: Context collection – Browser	✓
Test Case CSTC-OCVC-0300: Context collection – Operative System	✓
Test Case CSTC-OCVC-0400: Context collection – Geolocation	✓
Test Case CSTC-OCVC-0500: Context collection – Page referrer	✗
Test Case CSTC-OCVC-0600: Context collection – First visit timestamp	✗
Test Case CSTC-OCVC-0700: Returning customer	✓
Back office set	
Test Case BSTC-OCVC-0100: Update customer timer	✓
Test Case BSTC-OCVC-0200: Selecting customer – Accepted request	✓
Test Case BSTC-OCVC-0300: Selecting customer – Waiting customer	✓
Test Case BSTC-OCVC-0400: Selecting customer – Online customer	✓
Test Case BSTC-OCVC-0500: Selecting customer – History customer	✓
Test Case BSTC-OCVC-0600: Filter customers list	✓
Test Case BSTC-OCVC-0700: Edit customer’s information – Correct value	✓
Test Case BSTC-OCVC-0800: Edit customer’s name field – Wrong value	✓
Test Case BSTC-OCVC-0900: Edit customer’s e-mail field – Wrong value	✓
Test Case BSTC-OCVC-1000: Edit customer’s phone number field – Wrong value	✓
Test Case BSTC-OCVC-1100: Accept request	✓
Test Case BSTC-OCVC-1200: Close request – Online customer	✓
Test Case BSTC-OCVC-1300: Close request – Offline customer	✓
Test Case BSTC-OCVC-1400: Proactive engagement	✓
Test Case BSTC-OCVC-1500: Switch message delivery mode	✓
Test Case BSTC-OCVC-1600: Send new Message – Instant Message	✓
Test Case BSTC-OCVC-1700: Send new Message – E-mail	✓
Test Case BSTC-OCVC-1800: Send Messages to offline customers	✓
Test Case BSTC-OCVC-1900: Send Messages – Send URLs	✓
Test Case BSTC-OCVC-2000: Escape characters	✓
Test Case BSTC-OCVC-2100: Group messages	✓
Test Case BSTC-OCVC-2200: Open URL from customer’s navigation history	✓
Test Case BSTC-OCVC-2300: Add comments	✓
Test Case BSTC-OCVC-2400: Remove comments	✓

Test Case BSTC-OCVC-2500: Answer call request – Browser request	✓
Test Case BSTC-OCVC-2600: Answer call request – Customer’s app request	✓
Test Case BSTC-OCVC-2700: Call customer	✓
Test Case BSTC-OCVC-2800: Reject call	✓
Test Case BSTC-OCVC-2900: End call	✓
Test Case BSTC-OCVC-3000: Collapse call area	✓
Test Case BSTC-OCVC-3100: Expand call area	✓
Test Case BSTC-OCVC-3200: Mute stream – Audio	✓
Test Case BSTC-OCVC-3300: Mute stream – Video	✓
Test Case BSTC-OCVC-3400: Unmute stream – Audio	✓
Test Case BSTC-OCVC-3500: Unmute stream – Video	✓
Test Case BSTC-OCVC-3600: Select stroke size	✓
Test Case BSTC-OCVC-3700: Select stroke color	✓
Test Case BSTC-OCVC-3800: Draw on mobile device	✓
Test Case BSTC-OCVC-3900: Clear drawings	✓
Test Case BSTC-OCVC-4000: Blur screen	✓
Test Case BSTC-OCVC-4100: Clear screen	✓
Widget set	
Test Case WSTC-OCVC-0100: Access widget – Operator is online	✓
Test Case WSTC-OCVC-0200: Access widget – Operator is not online	✓
Test Case WSTC-OCVC-0300: Change widget state – Operator is not online and logs in after	✓
Test Case WSTC-OCVC-0400: Change widget state – Operator online and logs out after	✓
Test Case WSTC-OCVC-0500: Request help – Operator is online – Message	✓
Test Case WSTC-OCVC-0600: Call button – Show it	✓
Test Case WSTC-OCVC-0700: Call button – Do not show it	✓
Test Case WSTC-OCVC-0800: Request help – Operator is online – Call	✓
Test Case WSTC-OCVC-0900: Request help – Operator is not online – Select contact type	✓
Test Case WSTC-OCVC-1000: Request help – Operator is not online – Form filled correctly	✓
Test Case WSTC-OCVC-1100: Request help – Operator is not online – Form filled correctly	✓
Test Case WSTC-OCVC-1200: Request help – Operator is not online – Phone number filling	✓
Test Case WSTC-OCVC-1300: Request help – Operator is not online – Phone number incorrect filling	✓
Test Case WSTC-OCVC-1400: Send Message	✓
Test Case WSTC-OCVC-1500: Escape characters	✓
Test Case WSTC-OCVC-1600: Open URLs – URL pointing out-side the site	✓

Test Case WSTC-OCVC-1700: Open URLs – URL pointing inside the site	✓
Test Case WSTC-OCVC-1800: URL push	✓
Test Case WSTC-OCVC-1900: Group messages	✓
Call window set	
Test Case CSTC-OCVC-0100: Open call window	✓
Test Case CSTC-OCVC-0200: Open chat column	✓
Test Case CSTC-OCVC-0300: Close chat column	✓
Test Case CSTC-OCVC-0400: Send Message	✓
Test Case WSTC-OCVC-0500: Escape characters	✓
Test Case WSTC-OCVC-0600: Open URLs – URL pointing out-side the site	✓
Test Case WSTC-OCVC-0700: Group messages	✓
Test Case WSTC-OCVC-0800: Call operator	✓
Test Case WSTC-OCVC-0900: End call	✓
Test Case WSTC-OCVC-1000: Mute audio stream	✓
Test Case WSTC-OCVC-1100: Unmute audio stream	✓

Table 3.1. Tests results

From a total of 82 tests only 2 not passed which led to a 98% of acceptance.

Test Set	# of tests	Passed	Failed	% Acceptance	% Failure
Authentication	4	4	0	100%	0%
Context Collection	7	5	2	71%	29%
BO's Features	41	41	0	100%	0%
Widget's Features	19	19	0	100%	0%
Call Window Features	11	11	0	100%	0%
Total	82	80	2	98%	2%

Table 3.2. Tests interpretation

From the table analysis it is easy to see that the tests that failed both belong to the context collection group. In Table 3.1 we see that the tests were related with the referrer and first time collection. In the beginning those features weren’t thought to be implemented, and the UI was not prepared to display those values, however they were tracked and saved because they seemed relevant for future work on the project.

To analyze WebRTC calls, WebRTC Internals [2] from Google Chrome was used.

In addition to Chrome’s WebRTC Internals, WWC SDK also offers an event to analyze connectivity status with window.getStats API if available. The returned results are:

- Available bandwidth;
- Input level;
- Packets lost;
- RTT;
- Packets sent;
- Bytes sent.

4 Future Work

All the proposed scope was implemented and the project was successful.

The developed product will be used for demos and as a proof of concept either for its technologies, experience and product concept, which means that this POC will be used in the future and can be modified or evolve to a product if any potential interest appears.

Some features were left out of the scope, but those features were not forgotten and can be used to improve the project's concept.

This section presents the reader some features that were analyzed, but were not implemented due to time or complexity reasons.

Multiple operators
Change server in order to work with multiple registered operators. Furthermore, a new page can be created in order to add and remove operators.

Priority queues for customers and operators teams
Different request types can be added to different queues with different priorities. For instance, clients that requested a call could be all on the same queue, and the same operators team would answer them all.

Operators team could be divided by request type (chat, call, tablet), page type (products, sales, ...) and others.

Automatic customers' engagement
Currently, operators without any request can engage customers. This approach could be done automatically using rules to approach the customers.

For instance, if the same user visits the same page a certain number of times an automatic message could be sent in order to approach him. The set of rules could be customizable by operator.

Develop promotion feature
The reserved keyword :promo: is used to open a promotion div on the customer's browser. This feature could be improved by adding a list of promotions on the BO so that the operator could choose one and send it to the customer.

Collect/show more context
The context collection can be expanded. For instance, the first time the customer visits the hosting site or the page referrer are collected but not displayed. Besides that, the last conversation time, last visit, time between visits, visit duration or page visit duration can also be tracked.

CRM integration
If the hosting site holds a CRM, integration could be possible in order to enrich the customers' information provided to the operator.

Multiple active chats
The number of active conversations at the same time could be customizable.

Automatically infer customers' relationships
The BO could help the operator to answer the customers' needs. For instance, if the hosting site is a sell retailer website if the customer search for an amount of the same type of products the operator could suggest some related.

Chat transfer
Chat transfer could be possible in order to transfer the conversation to a more suited operator.

VoIP to all browsers
A huge improvement would be the development of VoIP calls to all browsers instead of the current solution that only support browsers with WebRTC technology.

Security Issues

Security issues such as data privacy and encryption and the use of cookies to track customers should be studied before this POC is used as a product.

UI Redefinition

A new UI is being prepared in order to meet the new features proposed and companies UI standards. This UI is currently being evaluated to decide if it will be rethought or implemented.

Message and comment edition
After sending a message, the same could be passive of edition or deletion.

Operators' metrics
From time to time (each week or month) a report with the operators' metrics could be generated. The report could contain information such as conversation mean duration, number of clients answered among others specific to some hosting sites such number of customers converted into clients or number of sales. This reports would help to improve the operators' approach and to collect the most active and efficient operators/teams.

Conversation qualification
In order to improve the customers' attendance and operators' quality, at the end of a conversation the customer could qualify (star rating for instance) the quality of his conversation.

Canned responses to FAQs
To help the operators to respond faster to their clients, a list of canned responses to frequently asked questions could be available on the BO. The operator would select and send the response immediately.

Widget Personalization
A page to customize the widget could be added in order to change the widget colors, depending on the widget owner preferences.

Operators internal chat
Operators could have an internal chat network in order to help themselves to answer their customers.

Profiles
Both customers and operators could have a profile. On the customer profile all of his information could be displayed and passive of edition. On the operator profile the operator could be able to edit his information (password, name, ...) and edit his engagement rules.

Find less intrusive ways to get users' information
The information on the BO could be autocompleted during the conversation. For instance, if the customer writes his name on the conversation the BO would identify it and set it to the database.

5 References

[1] "ipinfo.io," [Online]. Available: https://ipinfo.io/.

[2] Google Inc., "WebRTC internals," [Online]. Available: chrome://webrtc-internals/. [Accessed 03 June 2016].

[3] "Web Sequence Diagrams," [Online]. Available: https://www.websequencediagrams.com/. [Accessed 24 May 2016].

[4] "WebRTC internals," [Online]. Available: chrome://webrtc-internals/. [Accessed 1 June 2016].

[5] Wikipedia, "Jitter," 2016 April 2016. [Online]. Available: https://en.wikipedia.org/wiki/Jitter. [Accessed 1 June 2016].

6 Appendix

6.1 Test sets

6.1.1 Test Suite: Authentication

Test Case ASTC-OCVC-0100: Login – Registered operator		
Author:	jgrilo	
Summary: The purpose of this test case is to verify that the login of a registered operator is successful.		
Preconditions: 1. Internet connection.		

#:	Step actions:	Expected Results:
1	Go to the back office Login/Dashboard Page.	Login page is displayed correctly.
2	Type the username and password of a registered account.	Fields are filled successfully.
3	Press “Enter” or click on “Sign In”.	User is logged successfully.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	Registered operator at Database	
Keywords:	None	

Test Case ASTC-OCVC-0200: Login – Unregistered operator		
Author:	jgrilo	
Summary: The purpose of this test case is to verify that the login of a registered operator is unsuccessful.		
Preconditions: 1. Internet connection.		
#:	Step actions:	Expected Results:
1	Go to the back office Login/Dashboard Page.	Login page is displayed correctly.
2	Type the username and password of an unregistered account.	Fields are filled successfully.
3	Press “Enter” or click on “Sign In”.	A message error is displayed.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	None	
Keywords:	None	

Test Case ASTC-OCVC-0300: Login – Auto Login	
Author:	jgrilo

Summary: The purpose of this test case is to verify that the login of a registered operator is automatic if the operator have logged on the browser previously.		
Preconditions: 1. Execute a previous login without logout. 2. Internet connection.		
#:	Step actions:	Expected Results:
1	Go to the back office Login/Dashboard Page.	User is logged successfully.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	Registered operator at Database	
Keywords:	None	

Test Case ASTC-OCVC-0400: Logout		
Author:	jgrilo	
Summary: The purpose of this test case is to verify that the logout of a logged operator is successful.		
Preconditions: 1. Internet connection.		
#:	Step actions:	Expected Results:
1	Open operator management dropdown.	Operator management dropdown opens successfully.
2	Click on “Logout” button.	User successfully logout.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	ASTC-OCVC-100 or ASTC-OCVC-300	
Keywords:	None	

6.1.2 Test Suite: Context Collection

Test Case CSTC-OCVC-0100: Context collection – URL		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that the navigation history is collected on the widget and displayed at the BO		
Preconditions: None.		
#:	Step actions:	Expected Results:
1	Access widget.	Widget displays correctly.

		Collected URL is displayed correctly at BO.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	None	
Keywords:	None	

Test Case CSTC-OCVC-0200: Context collection – Browser		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that browser information is collected on the widget and displayed at the BO		
Preconditions: None.		
#:	Step actions:	Expected Results:
1	Access widget.	Widget displays correctly. Collected browser information is displayed correctly at BO.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	None	
Keywords:	None	

Test Case CSTC-OCVC-0300: Context collection – Operative System		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that OS information is collected on the widget and displayed at the BO		
Preconditions: None.		
#:	Step actions:	Expected Results:
1	Access widget.	Widget displays correctly. Collected OS information is displayed correctly at BO.
Execution type:	Manual	
Estimated exec.		

duration (min):	
Importance:	Medium
Requirements:	None
Keywords:	None

Test Case CSTC-OCVC-0400: Context collection – Geolocation

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that geolocation information is collected on the widget and displayed at the BO		
Preconditions: 1. Customer must allow access to geolocation.		
#:	Step actions:	Expected Results:
1	Access widget.	Widget displays correctly. Collected geolocation information is displayed correctly at BO.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	None	
Keywords:	None	

Test Case CSTC-OCVC-0500: Context collection – Page referrer

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that page referrer URL is collected on the widget and displayed at the BO		
Preconditions: 1. First page visited during customer's navigation.		
#:	Step actions:	Expected Results:
1	Access widget.	Widget displays correctly. Collected referrer information is displayed correctly at BO.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Low	
Requirements:	None	

Keywords:	None

Test Case CSTC-OCVC-0600: Context collection – First visit timestamp

Author:	jgrilo

Summary:

The purpose of this test case is to ensure that hte first visit timestamp information is collected on the widget and displayed at the BO

Preconditions:

1. Customer never used widget before.

#:	Step actions:	Expected Results:
1	Access widget.	Widget displays correctly. Collected information is displayed correctly at BO.

Execution type:	Manual
Estimated exec. duration (min):	
Importance:	Low
Requirements:	None
Keywords:	None

Test Case CSTC-OCVC-0700: Returning customer

Author:	jgrilo

Summary:

The purpose of this test case is to ensure that a returning customer is identified successfully.

Preconditions:

1. Customer already used site before.

#:	Step actions:	Expected Results:
1	Access widget.	Widget displays correctly with previous conversations. BO displays customer with previous information collected.

Execution type:	Manual
Estimated exec. duration (min):	
Importance:	High
Requirements:	None
Keywords:	None

6.1.3 Test Suite: Back office's features

Test Case BSTC-OCVC-0100: Update customer timer		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that customers' timers successfully update in real time.		
Preconditions: 1. At least one customer is on BO.		
#:	Step actions:	Expected Results:
1	Select a tab with customers.	"Requests" tab is displayed correctly.
2	Wait for at least one minute.	Customers' timers are updated correctly.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	ASTC-OCVC-100 or ASTC-OCVC-300	
Keywords:	None	

Test Case BSTC-OCVC-0200: Selecting customer – Accepted request		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that a customer is selected successfully.		
Preconditions: 1. Operator already accepted a customer.		
#:	Step actions:	Expected Results:
1	Select "Requests" tab.	"Requests" tab is displayed correctly.
2	Select customer on "Request Accepted"	BO is populated with customer's information.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	ASTC-OCVC-100 or ASTC-OCVC-300	
Keywords:	None	

Test Case BSTC-OCVC-0300: Selecting customer – Waiting customer		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that a customer is selected successfully.		
Preconditions: 1. At least one customer requested help.		
#:	Step actions:	Expected Results:

1	Select “Requests” tab.	“Requests” tab is displayed correctly.
2	Select customer on “Requests Pending”	BO is populated with customer’s information.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	ASTC-OCVC-100 or ASTC-OCVC-300	
Keywords:	None	

Test Case BSTC-OCVC-0400: Selecting customer – Online customer

Author:	jgrilo	

Summary:
The purpose of this test case is to ensure that a customer is selected successfully.

Preconditions:
1. At least one customer is online and have not requested help.

#:	Step actions:	Expected Results:
1	Select “Online” tab.	“Online” tab is displayed correctly.
2	Select customer on the list	BO is populated with customer’s information.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	ASTC-OCVC-100 or ASTC-OCVC-300	
Keywords:	None	

Test Case BSTC-OCVC-0500: Selecting customer – History customer

Author:	jgrilo	

Summary:
The purpose of this test case is to ensure that a customer is selected successfully.

Preconditions:
1. At least one customer is on history.

#:	Step actions:	Expected Results:
1	Select “History” tab.	“History” tab is displayed correctly.
2	Select customer on the list	BO is populated with customer’s information.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	

Requirements:	ASTC-OCVC-100 or ASTC-OCVC-300
Keywords:	None

Test Case BSTC-OCVC-0600: Filter customers list

Author:	jgrilo	

Summary:

The purpose of this test case is to ensure that only customers that met the filter specification are displayed.

Preconditions:

1. At least one customer is on the list.

#:	Step actions:	Expected Results:
1	Select a tab with customers.	Tab is displayed correctly.
2	Select filter, or set of filters.	Filters UI is displayed correctly. Customer that don't met filters specification are hidden.

Execution type:	Manual
Estimated exec. duration (min):	
Importance:	Low
Requirements:	ASTC-OCVC-100 or ASTC-OCVC-300
Keywords:	None

Test Case BSTC-OCVC-0700: Edit customer's information – Correct value

Author:	jgrilo

Summary:

The purpose of this test case is to ensure that is possible to edit some of the customer's information fields. The possible fields are:

- Name;
- E-mail;
- Phone number.

Preconditions:

1. At least one customer is in one list.

#:	Step actions:	Expected Results:
1	Select a tab with customers.	Tab is displayed correctly.
2	Select a customer.	BO is populated with customer's information.
3	Click field to edit to select it.	Field UI reacts to selection.
4	Entry new value.	Input field displays new characters.

5	Leave field to save value.	New value is saved and correctly displayed.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	BSTC-OCVC-200, BSTC-OCVC-300, BSTC-OCVC-400 or BSTC-OCVC-500	
Keywords:	None	

Test Case BSTC-OCVC-0800: Edit customer's name field – Wrong value

Author:	jgrilo	

Summary:
The purpose of this test case is to ensure that is not possible to edit customer's name field with invalid values.

Preconditions:
1. At least one customer is in one list.

#:	Step actions:	Expected Results:
1	Select a tab with customers.	Tab is displayed correctly.
2	Select a customer.	BO is populated with customer's information.
3	Click name field to edit to select it.	Field UI reacts to selection.
4	Entry new (invalid) value.	Input field displays new characters.
5	Leave field to save value.	New value is discarded and old value is displayed. Invalid names include: • Empty name; • Blank name; • Visitor #.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Low	
Requirements:	BSTC-OCVC-200, BSTC-OCVC-300, BSTC-OCVC-400 or BSTC-OCVC-500	
Keywords:	None	

Test Case BSTC-OCVC-0900: Edit customer's e-mail field – Wrong value

Author:	jgrilo	

Summary:
The purpose of this test case is to ensure that is not possible to edit customer's e-mail field with invalid values.

Preconditions:

1. At least one customer is in one list.

#:	Step actions:	Expected Results:
1	Select a tab with customers.	Tab is displayed correctly.
2	Select a customer.	BO is populated with customer's information.
3	Click e-mail field to edit to select it.	Field UI reacts to selection.
4	Entry new (invalid) value.	Input field displays new characters.
5	Leave field to save value.	New value is discarded and old value is displayed. Invalid names include: • Empty e-mail; • Blank e-mail; • Wrong format e-mail.

Execution type:	Manual
Estimated exec. duration (min):	
Importance:	Low
Requirements:	BSTC-OCVC-200, BSTC-OCVC-300, BSTC-OCVC-400 or BSTC-OCVC-500
Keywords:	None

Test Case BSTC-OCVC-1000: Edit customer's phone number field – Wrong value

Author:	jgrilo

Summary:

The purpose of this test case is to ensure that is not possible to edit customer's phone number field with invalid values.

Preconditions:

1. At least one customer is in one list.

#:	Step actions:	Expected Results:
1	Select a tab with customers.	Tab is displayed correctly.
2	Select a customer.	BO is populated with customer's information.
3	Click phone number field to edit to select it.	Field UI reacts to selection.
4	Entry new (invalid) value.	Input field displays new characters if valid. Valid character are: • Numbers;

		• '+' at the beginning of the string.
5	Leave field to save value.	New value is discarded and old value is displayed. Invalid names include: • Empty phone number; • Blank phone number; • Wrong format phone number.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Low	
Requirements:	BSTC-OCVC-200, BSTC-OCVC-300, BSTC-OCVC-400 or BSTC-OCVC-500	
Keywords:	None	

Test Case BSTC-OCVC-1100: Accept request		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that is possible to accept a customer's request in order to help him.		
Preconditions: 1. At least one customer requested help. 2. Operator is not answering any customer yet.		
#:	Step actions:	Expected Results:
1	Select a tab "Requests"	Tab is displayed correctly.
2	Select a customer from "Requests Pending".	BO is populated with customer's information.
3	Click on "Answer" button.	Customer is moved from "Requests Pending" to "Request Accepted". Customer's widget displays operator's information
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	

Requirements:	BSTC-OCVC-300
Keywords:	None

Test Case BSTC-OCVC-1200: Close request – Online customer

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that is possible to close an open request.		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Select a tab "Requests"	Tab is displayed correctly.
2	Select a customer from "Request Accepted".	BO is populated with customer's information.
3	Click on "Close" button.	Customer is moved from "Request Accepted" to "Online". Conversation is archived on "History". Customer's widget removes operator's information.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	BSTC-OCVC-1100 or BSTC-OCVC-1400	
Keywords:	None	

Test Case BSTC-OCVC-1300: Close request – Offline customer

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that is possible to close an open request.		
Preconditions: None.		
#:	Step actions:	Expected Results:
1	Select a tab "Requests"	Tab is displayed correctly.
2	Select a customer from "Request Accepted".	BO is populated with customer's information.
3	Click on "Close" button.	Customer is removed from "Request Accepted".

		Conversation is archived on “History”.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	BSTC-OCVC-1100 or BSTC-OCVC-1400	
Keywords:	None	

Test Case BSTC-OCVC-1400: Proactive engagement

Author:	jgrilo	

Summary:
The purpose of this test case is to ensure that is possible to talk with a customer without him have made a request.

Preconditions:

1. Operator is not answering a customer yet.
2. At least one costumer is online and do not have made a request.

#:	Step actions:	Expected Results:
1	Select a tab “Online”	Tab is displayed correctly.
2	Select a customer.	BO is populated with customer’s information.
3	Select chat’s input box.	UI is displayed correctly.
4	Input a message	Message is displayed on the input box correctly.
5	Press “Enter”	Message is sent in the default delivery type (IM) and displayed at BO. Message is received by customer and displayed. Customer is moved from “Online” to “Request Accepted”. Customer’s widget displays operator’s information.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-400	
Keywords:	None	

Test Case BSTC-OCVC-1500: Switch message delivery mode		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that is possible to change message delivery mode between e-mail and instant message.		
Preconditions: 1. Operator is wants to send a message to a valid customer (a customer is already on conversation or a customer on “Online” tab is available)		
#:	Step actions:	Expected Results:
1	Select a tab with valid customer.	Tab is displayed correctly.
2	Select the customer.	BO is populated with customer’s information.
3	Click on message delivery switching buttons	Buttons UI is displayed correctly.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-200, BSTC-OCVC-300, BSTC-OCVC-400 or BSTC-OCVC-500	
Keywords:	None	

Test Case BSTC-OCVC-1600: Send new Message – Instant Message		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that is possible to talk with customers by chat.		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Select a tab with valid customer.	Tab is displayed correctly.
2	Select the customer.	BO is populated with customer’s information.
3	Click on IM delivery button.	Buttons UI is displayed correctly.
4	Select chat’s input box.	UI is displayed correctly.
5	Input a message	Message is displayed on the input box correctly.
6	Press “Enter”	Message is sent as an IM and displayed at BO.

		Message is received by customer and displayed.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-1100 or BSTC-OCVC-1400	
Keywords:	None	

Test Case BSTC-OCVC-1700: Send new Message – E-mail		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that is possible to talk with customers by e-mail.		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Select a tab with valid customer.	Tab is displayed correctly.
2	Select the customer.	BO is populated with customer's information.
3	Click on e-mail delivery button.	Buttons UI is displayed correctly.
4	Select chat's input box.	UI is displayed correctly.
5	Input a message	Message is displayed on the input box correctly.
6	Click on "Send" button	Message is sent in as an e-mail and displayed at BO. Message is received by customer and displayed.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-1100 or BSTC-OCVC-1400	
Keywords:	None	

Test Case BSTC-OCVC-1800: Send Messages to offline customers	
Author:	jgrilo
Summary: The purpose of this test case is to ensure that the only possible way to talk with offline customers by chat is sending e-mail.	

Preconditions: None		
#:	Step actions:	Expected Results:
1	Select a tab with valid customer.	Tab is displayed correctly.
2	Select the customer.	BO is populated with customer’s information. Message delivery type is set as E-mail and cannot be changed.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-1100	
Keywords:	None	

Test Case BSTC-OCVC-1900: Send Messages – Send URLs		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that sent URLs display as an image collected from meta-tag (if available), and clickable URL.		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Select a tab with valid customer.	Tab is displayed correctly.
2	Select the customer.	BO is populated with customer’s information.
3	Click on IM delivery button.	Buttons UI is displayed correctly.
4	Select chat’s input box.	UI is displayed correctly.
5	Input URL as message.	Message is displayed on the input box correctly.
6	Press “Enter”.	Message is sent as an IM and displayed at BO. Message is received by customer and displayed. URL is displayed correctly.
Execution type:	Manual	
Estimated exec.		

duration (min):	
Importance:	Medium
Requirements:	BSTC-OCVC-1100 or BSTC-OCVC-1400
Keywords:	None

Test Case BSTC-OCVC-2000: Escape characters

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that mal intended users cannot inject HTML or scripts.		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Select a tab with customers.	Tab is displayed correctly.
2	Select a customer.	BO is populated with customer’s information.
3	Select an input field	UI is displayed correctly
4	Try to inject a script/HTML	Message is displayed on input field
5	Press “Enter”	Characters on message are escaped.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	BSTC-OCVC-200, BSTC-OCVC-300, BSTC-OCVC-400, BSTC-OCVC-500, BSTC-OCVC-1100 or BSTC-OCVC-1400	
Keywords:	None	

Test Case BSTC-OCVC-2100: Group messages

Author:	jgrilo	
Summary: The purpose of this test case is to verify that messages sent by the same user on the same minute are grouped.		
Preconditions: 1. Customer has some messages on list.		
#:	Step actions:	Expected Results:
1	Select a tab with customers.	Tab is displayed correctly.
2	Select a customer.	BO is populated with customer’s information.

3	Click on IM delivery button.	Buttons UI is displayed correctly.
4	Select chat's input box.	UI is displayed correctly.
5	Input a message	Message is displayed on the input box correctly.
6	Press "Enter"	Message is sent as an IM and displayed at BO. Message is received by customer and displayed.
7	Repeat 5 and 6 several times	Messages on the same minute append payload to previous.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Low	
Requirements:	BSTC-OCVC-1100 or BSTC-OCVC-1400	
Keywords:	None	

Test Case BSTC-OCVC-2200: Open URL from customer's navigation history		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that operator can open customer's collected URLs.		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Select a tab with customers.	Tab is displayed correctly.
2	Select a customer.	BO is populated with customer's information.
3	Hover with mouse on customer's collected URLs.	URL is translated to left and "Open" button is displayed.
4	Click on "Open" button.	New tab opens on browser with selected URL.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-200, BSTC-OCVC-300, BSTC-OCVC-400, BSTC-OCVC-500	
Keywords:	None	

Test Case BSTC-OCVC-2300: Add comments		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that operator can append comments to customers		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Select a tab with customers.	Tab is displayed correctly.
2	Select a customer.	BO is populated with customer's information.
3	Select "Comments" tab.	Tab is displayed correctly.
4	Select comment's input box.	UI is displayed correctly.
5	Input comment	Comment is displayed at input box.
6	Press "Enter" or click on "Add" button	Comment is appended to tab.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-200, BSTC-OCVC-300, BSTC-OCVC-400, BSTC-OCVC-500	
Keywords:	None	

Test Case BSTC-OCVC-2400: Remove comments		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that operator can append comments to customers.		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Select a tab with customers.	Tab is displayed correctly.
2	Select a customer.	BO is populated with customer's information.
3	Select "Comments" tab.	Tab is displayed correctly.
4	Hover with mouse on customer's comments.	"Delete" button is displayed.
5	Click on "Delete" button.	Selected comment is removed from customer's list.
Execution type:	Manual	
Estimated exec.		

duration (min):	
Importance:	Low
Requirements:	BSTC-OCVC-200, BSTC-OCVC-300, BSTC-OCVC-400, BSTC-OCVC-500
Keywords:	None

Test Case BSTC-OCVC-2500: Answer call request – Browser request

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that operator can talk with customer by call.		
Preconditions: 1. Customer requested a voice call from the widget. 2. Give access to microphone.		
#:	Step actions:	Expected Results:
1	Select a "Requests" tab	Tab is displayed correctly.
2	Select customer on "Request Accepted".	BO is populated with customer's information.
3	Click on "Accept Call" button	BO's UI updates: Call management buttons are displayed; Call timer starts; Call area expands. Remote stream starts.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	BSTC-OCVC-1100 or BSTC-OCVC-1400	
Keywords:	None	

Test Case BSTC-OCVC-2600: Answer call request – Customer's app request

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that operator can talk with customer by call.		
Preconditions: 1. Operator is not answering a customer. 2. Customer made a call request from the mobile app. 3. Allow access to camera and microphone.		
#:	Step actions:	Expected Results:
1	Select a "Requests" tab	Tab is displayed correctly.
2	Select customer on "Requests Pending".	BO is populated with customer's information.

3	Click on “Answer” button	BO’s UI updates: Call management buttons are displayed; Draw buttons are displayed; Blur button is displayed; Mobile image is displayed. Remote stream starts.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	BSTC-OCVC-1100	
Keywords:	None	

Test Case BSTC-OCVC-2700: Call customer		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that operator can talk with customer by call.		
Preconditions: 1. Customer has a phone number associated. 2. Allow access to microphone.		
#:	Step actions:	Expected Results:
1	Select a “Requests” tab	Tab is displayed correctly.
2	Select customer on “Request Accepted”.	BO is populated with customer’s information.
3	Click on “Make Call” button	BO’s UI updates: Call management buttons are displayed.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	BSTC-OCVC-1100 or BSTC-OCVC-1400	
Keywords:	None	

Test Case BSTC-OCVC-2800: Reject call	
Author:	jgrilo
Summary: The purpose of this test case is to ensure that operator can end a call.	

Preconditions:

1. Customer request voice call from widget.

#:	Step actions:	Expected Results:
1	Select a “Requests” tab	Tab is displayed correctly.
2	Select customer on “Request Accepted”.	BO is populated with customer’s information.
3	Click on “End Call” button	BO’s UI updates: Call management buttons are removed.

Execution type:	Manual
Estimated exec. duration (min):	
Importance:	High
Requirements:	BSTC-OCVC-1100 or BSTC-OCVC-1400
Keywords:	None

Test Case BSTC-OCVC-2900: End call

Author:	jgrilo

Summary:

The purpose of this test case is to ensure that operator can end a call with a customer.

Preconditions:

#:	Step actions:	Expected Results:
1	Select a “Requests” tab	Tab is displayed correctly.
2	Select customer on “Request Accepted”.	BO is populated with customer’s information.
3	Click on “End call” button or “Close Conversation” button.	BO’s UI updates: Call management buttons are removed.

Execution type:	Manual
Estimated exec. duration (min):	
Importance:	Low
Requirements:	BSTC-OCVC-2500, BSTC-OCVC-2600 or BSTC-OCVC-2700
Keywords:	None

Test Case BSTC-OCVC-3000: Collapse call area

Author:	jgrilo

Summary:

The purpose of this test case is to ensure that operator switch call area mode.

Preconditions:

1. Call area is expanded.

#:	Step actions:	Expected Results:
1	Select a “Requests” tab	Tab is displayed correctly.
2	Select customer on “Request Accepted”.	BO is populated with customer’s information.
3	Click on “Expand/Collapse area” button	BO’s UI updates: Button UI updates; Call area collapses.

Execution type:	Manual
Estimated exec. duration (min):	
Importance:	Low
Requirements:	BSTC-OCVC-2500 or BSTC-OCVC-2700
Keywords:	None

Test Case BSTC-OCVC-3100: Expand call area

Author:	jgrilo

Summary:
The purpose of this test case is to ensure that operator switch call area mode.

Preconditions:
1. Call area is collapsed.

#:	Step actions:	Expected Results:
1	Select a “Requests” tab	Tab is displayed correctly.
2	Select customer on “Request Accepted”.	BO is populated with customer’s information.
3	Click on “Expand/Collapse area” button	BO’s UI updates: Button UI updates; Call area Expand.

Execution type:	Manual
Estimated exec. duration (min):	
Importance:	Low
Requirements:	BSTC-OCVC-2500 or BSTC-OCVC-2700
Keywords:	None

Test Case BSTC-OCVC-3200: Mute stream – Audio

Author:	jgrilo

Summary:
The purpose of this test case is to ensure that operator can mute his audio stream.

Preconditions:
1. Audio stream is unmuted.

#:	Step actions:	Expected Results:

1	Select a “Requests” tab	Tab is displayed correctly.
2	Select customer on “Request Accepted”.	BO is populated with customer’s information.
3	Click on “Mute/Unmute audio” button	Button UI updates. Customer stops receiving audio.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-2500 , BSTC-OCVC-2600 or BSTC-OCVC-2700	
Keywords:	None	

Test Case BSTC-OCVC-3300: Mute stream – Video

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that operator can mute his video stream.		
Preconditions: 1. Video stream is unmuted.		
#:	Step actions:	Expected Results:
1	Select a “Requests” tab	Tab is displayed correctly.
2	Select customer on “Request Accepted”.	BO is populated with customer’s information.
3	Click on “Mute/Unmute video” button	Button UI updates. Customer stops receiving video.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-2600	
Keywords:	None	

Test Case BSTC-OCVC-3400: Unmute stream – Audio

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that operator can unmute his audio stream.		
Preconditions: 1. Audio stream is muted.		
#:	Step actions:	Expected Results:
1	Select a “Requests” tab	Tab is displayed correctly.

2	Select customer on "Request Accepted".	BO is populated with customer's information.
3	Click on "Mute/Unmute audio" button	Button UI updates. Customer starts receiving audio.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-2500 , BSTC-OCVC-2600 or BSTC-OCVC-2700	
Keywords:	None	

Test Case BSTC-OCVC-3500: Unmute stream – Video

Author:	jgrilo	

Summary:
The purpose of this test case is to ensure that operator can unmute his video stream.

Preconditions:
1. Video stream is muted.

#:	Step actions:	Expected Results:
1	Select a "Requests" tab	Tab is displayed correctly.
2	Select customer on "Request Accepted".	BO is populated with customer's information.
3	Click on "Mute/Unmute video" button	Button UI updates. Customer starts receiving video.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-2600	
Keywords:	None	

Test Case BSTC-OCVC-3600: Select stroke size

Author:	jgrilo	

Summary:
The purpose of this test case is to ensure that operator select the stroke size to draw with successfully.

Preconditions:
None

#:	Step actions:	Expected Results:
1	Select a "Requests" tab	Tab is displayed correctly.

2	Select customer on “Request Accepted”.	BO is populated with customer’s information.
3	Click on “Stroke size changing” button	Button UI updates.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-2600	
Keywords:	None	

Test Case BSTC-OCVC-3700: Select stroke color

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that operator select which color to draw with successfully.		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Select a “Requests” tab	Tab is displayed correctly.
2	Select customer on “Request Accepted”.	BO is populated with customer’s information.
3	Click on “Color changing” button	Button UI updates.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-2600	
Keywords:	None	

Test Case BSTC-OCVC-3800: Draw on mobile device

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that operator send drawings to mobile device successfully.		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Select a “Requests” tab	Tab is displayed correctly.
2	Select customer on “Request Accepted”.	BO is populated with customer’s information.
3	Click on BO’s screen and drag.	Stroke is displayed on BO’s mobile

		screen and on mobile app.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	BSTC-OCVC-2600	
Keywords:	None	

Test Case BSTC-OCVC-3900: Clear drawings

Author:	jgrilo

Summary:
The purpose of this test case is to ensure that operator can clear the drawings sent to mobile device successfully.

Preconditions:

1. Operator is logged.
2. Operator is on call (mobile) with a customer.

#:	Step actions:	Expected Results:
1	Select a “Requests” tab	Tab is displayed correctly.
2	Select customer on “Request Accepted”.	BO is populated with customer’s information.
3	Click on “Clean” button.	Drawings are removed from BO’s mobile screen and mobile app.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-2600	
Keywords:	None	

Test Case BSTC-OCVC-4000: Blur screen

Author:	jgrilo

Summary:
The purpose of this test case is to ensure that operator can blur the image received from the mobile device.

Preconditions:

1. Image is clear.

#:	Step actions:	Expected Results:
1	Select a “Requests” tab	Tab is displayed correctly.
2	Select customer on “Request Accepted”.	BO is populated with customer’s information.

3	Click on “Blur/Clear” button.	Image received from mobile device become blurred.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-2600	
Keywords:	None	

Test Case BSTC-OCVC-4100: Clear screen

Author:	jgrilo

Summary:
The purpose of this test case is to ensure that operator can blur the image received from the mobile device.

Preconditions:

1. Image is blurred.

#:	Step actions:	Expected Results:
1	Select a “Requests” tab	Tab is displayed correctly.
2	Select customer on “Request Accepted”.	BO is populated with customer’s information.
3	Click on “Blur/Clear” button.	Image received from mobile device become blurred.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	BSTC-OCVC-2600	
Keywords:	None	

6.1.4 Test Suite: Widget’s features

Test Case WSTC-OCVC-0100: Access widget – Operator is online

Author:	jgrilo

Summary:
The purpose of this test case is to ensure that the widget is injected correctly.

Preconditions:

2. At least one operator is online.
3. Internet connection.
4. Widget must be injected in site.

#:	Step actions:	Expected Results:
1	Visit an URL where the snippet is injected.	Widget’s UI is displayed correctly (chat).

		At BO customer is displayed as online.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	None	
Keywords:	None	

Test Case WSTC-OCVC-0200: Access widget – Operator is not online		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that the widget is injected correctly.		
Preconditions: 1. Operators are not online on back office. 2. Internet connection. 3. Widget must be injected in site.		
#:	Step actions:	Expected Results:
1	Visit an URL where the snippet is injected.	Widget’s UI is displayed correctly (request form).
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	None	
Keywords:	None	

Test Case WSTC-OCVC-0300: Change widget state – Operator is not online and logs in after		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that the widget state changes when operator change his state.		
Preconditions: 1. Operator logs in sometime after accessing widget.		
#:	Step actions:	Expected Results:
1	Visit an URL where the snippet is injected.	Widget’s UI is displayed correctly (request form).
2	Operator logs in.	Widget display as WSTC-OCVC-0100
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	WSTC-OCVC-0200	

Keywords:	None

Test Case WSTC-OCVC-0400: Change widget state – Operator online and logs out after		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that the widget state changes when operator change his state.		
Preconditions: 1. Operator logs out sometime after accessing widget.		
#:	Step actions:	Expected Results:
1	Visit an URL where the snippet is injected.	Widget's UI is displayed correctly (request form).
2	Operator login.	Widget display as WSTC-OCVC-0200
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	WSTC-OCVC-0100	
Keywords:	None	

Test Case WSTC-OCVC-0500: Request help – Operator is online – Message		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that customer can ask for help by chat to operators correctly.		
Preconditions: 1. At least one operator is online.		
#:	Step actions:	Expected Results:
1	Select message input box	Widget's UI is displayed correctly.
2	Input message	Message is displayed at input box.
3	Press "Enter"	Message is sent as an IM and displayed at BO. Message is received by customer and displayed. At BO customer is moved from "Online" to "Requests Pending".
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	

Requirements:	WSTC-OCVC-0100
Keywords:	None

Test Case WSTC-OCVC-0600: Call button – Show it

Author:	jgrilo

Summary:
The purpose of this test case is to ensure that customers can only call from browsers that support WebRTC.

Preconditions:
1. At least one operator is online.
2. Browser supports WebRTC.

#:	Step actions:	Expected Results:
1	Access widget.	"Call" button is displayed.

Execution type:	Manual
Estimated exec. duration (min):	
Importance:	High
Requirements:	WSTC-OCVC-0100
Keywords:	None

Test Case WSTC-OCVC-0700: Call button – Do not show it

Author:	jgrilo

Summary:
The purpose of this test case is to ensure that customers can only call from browsers that support WebRTC.

Preconditions:
1. At least one operator is online.
2. Browser does not supports WebRTC.

#:	Step actions:	Expected Results:
1	Access widget.	"Call" button is not displayed.

Execution type:	Manual
Estimated exec. duration (min):	
Importance:	High
Requirements:	WSTC-OCVC-0100
Keywords:	None

Test Case WSTC-OCVC-0800: Request help – Operator is online – Call

Author:	jgrilo

Summary:
The purpose of this test case is to ensure that customer can ask for help by chat to operators correctly.

Preconditions:

None		
#:	Step actions:	Expected Results:
1	Click on “Call” button.	Call window opens. At BO customer is moved from “Online” to “Requests Pending”.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	WSTC-OCVC-0600	
Keywords:	None	

Test Case WSTC-OCVC-0900: Request help – Operator is not online – Select contact type		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that customer can ask for help to operators correctly.		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Click on “Contact me by” dropdown	Widget’s UI is displayed correctly.
2	Select one contact method	Input method is displayed.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	WSTC-OCVC-0200	
Keywords:	None	

Test Case WSTC-OCVC-1000: Request help – Operator is not online – Form filled correctly		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that customer can ask for help to operators correctly.		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Fill request form correctly.	Widget’s UI is displayed correctly.
3	Press “Enter” or “Send Button”.	Widget’s UI display correctly. At BO customer is moved from “Online” to “Requests Pending”.

Execution type:	Manual
Estimated exec. duration (min):	
Importance:	High
Requirements:	WSTC-OCVC-0400
Keywords:	None

Test Case WSTC-OCVC-1100: Request help – Operator is not online – Form filled correctly

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that customer cannot ask for help to operators if form is not filled correctly.		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Fill request form incorrectly.	Widget’s UI is displayed correctly.
3	Press “Enter” or “Send Button”.	Widget’s UI display correctly and present which fields are incorrect.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	WSTC-OCVC-0400	
Keywords:	None	

Test Case WSTC-OCVC-1200: Request help – Operator is not online – Phone number filling

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that customer cannot input an invalid phone number.		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Select to be contacted by phone	Widget’s UI is displayed correctly.
2	Select phone input.	Input UI displays correctly.
3	Fill form with valid phone number.	Value is displayed at input. Valid characters are: Number; ‘+’ at the end of string.
Execution type:	Manual	

Estimated exec. duration (min):	
Importance:	Medium
Requirements:	WSTC-OCVC-0400
Keywords:	None

Test Case WSTC-OCVC-1300: Request help – Operator is not online – Phone number incorrect filling

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that customer cannot input an invalid phone number.		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Select to be contacted by phone	Widget's UI is displayed correctly.
2	Select phone input.	Input UI displays correctly.
3	Fill form with invalid phone number.	Invalid characters are rejected. Valid characters are: Number; '+' at the end of string.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	WSTC-OCVC-0400	
Keywords:	None	

Test Case WSTC-OCVC-1400: Send Message

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that customer can send messages to the operators.		
Preconditions: None.		
#:	Step actions:	Expected Results:
1	Select message input box	Widget's UI is displayed correctly.
2	Input message	Message is displayed at input box.
3	Press "Enter"	Message is sent as an IM and displayed at BO.

		Message is received by customer and displayed.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	WSTC-OCVC-0100	
Keywords:	None	

Test Case WSTC-OCVC-1500: Escape characters

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that mal intended users cannot inject HTML or scripts.		
Preconditions: None		
#:	Step actions:	Expected Results:
3	Select an input field	UI is displayed correctly
4	Try to inject a script/HTML	Message is displayed on input field
5	Press “Enter”	Characters on message are escaped.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	WSTC-OCVC-0100 or WSTC-OCVC-0200	
Keywords:	None	

Test Case WSTC-OCVC-1600: Open URLs – URL pointing outside the site

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that customers can open received URLs.		
Preconditions: 1. Customer have received an URL		
#:	Step actions:	Expected Results:
1	Click on received URL	New tab opens with URL.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	WSTC-OCVC-0100	

Keywords:	None

Test Case WSTC-OCVC-1700: Open URLs – URL pointing inside the site

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that customers can open previously received URLs.		
Preconditions: 1. Customer have received an URL		
#:	Step actions:	Expected Results:
1	Click on received URL	Page reload to the URL.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	WSTC-OCVC-0100	
Keywords:	None	

Test Case WSTC-OCVC-1800: URL push

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that customers is redirected to URL automatically when receive it if it points inside the site.		
Preconditions: 1. Customer receive an URL that points inside the site.		
#:	Step actions:	Expected Results:
1	Wait 5 seconds after receiving URL	Page reload to the URL.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	WSTC-OCVC-0100	
Keywords:	None	

Test Case WSTC-OCVC-1900: Group messages

Author:	jgrilo	
Summary: The purpose of this test case is to verify that messages sent by the same user on the same minute are grouped.		
Preconditions: 1. Customer has some messages on list.		
#:	Step actions:	Expected Results:
1	Select chat’s input box.	UI is displayed correctly.

2	Input a message.	Message is displayed on the input box correctly.
3	Press “Enter”.	Message is sent and displayed at widget. Message is received by operator and displayed.
4	Repeat 2 and 3 several times.	Messages on the same minute append payload to previous.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Low	
Requirements:	WSTC-OCVC-0100	
Keywords:	None	

6.1.5 Test Suite: Call window’s features

Test Case CSTC-OCVC-0100: Open call window		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that call window opens correctly.		
Preconditions: None		
#:	Step actions:	Expected Results:
1	Access call window	Window UI is displayed correctly
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	WSTC-OCVC-0800	
Keywords:	None	

Test Case CSTC-OCVC-0200: Open chat column		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that call window animations work correctly.		
Preconditions: Chat column is collapsed.		
#:	Step actions:	Expected Results:
1	Click on “Expand/Collapse chat”	Button UI displays. Chat column opens with previous conversations.
Execution type:	Manual	

Estimated exec. duration (min):	
Importance:	Medium
Requirements:	CSTC-OCVC-0100
Keywords:	None

Test Case CSTC-OCVC-0300: Close chat column

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that call window animations work correctly.		
Preconditions: Chat column is expanded.		
#:	Step actions:	Expected Results:
1	Click on “Expand/Collapse chat”	Button UI displays. Chat column closes.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	CSTC-OCVC-0100	
Keywords:	None	

Test Case CSTC-OCVC-0400: Send Message

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that customer can send messages to the operators.		
Preconditions: None.		
#:	Step actions:	Expected Results:
1	Select message input box	Widget’s UI is displayed correctly.
2	Input message	Message is displayed at input box.
3	Press “Enter”	Message is sent as an IM and displayed to customer. Message is received by operator and displayed.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	CSTC-OCVC-0200	
Keywords:	None	

Test Case WSTC-OCVC-0500: Escape characters		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that mal intended users cannot inject HTML or scripts.		
Preconditions: None		
#:	Step actions:	Expected Results:
3	Select an input field	UI is displayed correctly
4	Try to inject a script/HTML	Message is displayed on input field
5	Press “Enter”	Characters on message are escaped.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	CSTC-OCVC-0400	
Keywords:	None	

Test Case WSTC-OCVC-0600: Open URLs – URL pointing outside the site		
Author:	jgrilo	
Summary: The purpose of this test case is to ensure that customers can open received URLs.		
Preconditions: 1. Customer have received an URL		
#:	Step actions:	Expected Results:
1	Click on received URL	New tab opens with URL.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	CSTC-OCVC-0200	
Keywords:	None	

Test Case WSTC-OCVC-0700: Group messages		
Author:	jgrilo	
Summary: The purpose of this test case is to verify that messages sent by the same user on the same minute are grouped.		
Preconditions: 1. Customer has some messages on list.		
#:	Step actions:	Expected Results:

1	Select chat’s input box.	UI is displayed correctly.
2	Input a message.	Message is displayed on the input box correctly.
3	Press “Enter”.	Message is sent and displayed at widget. Message is received by operator and displayed.
4	Repeat 2 and 3 several times.	Messages on the same minute append payload to previous.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Low	
Requirements:	CSTC-OCVC-0400	
Keywords:	None	

Test Case WSTC-OCVC-0800: Call operator		
Author:	jgrilo	
Summary: The purpose of this test case is to verify that customer can talk with operator by call.		
Preconditions: 1. Customer is being answered by operator. 2. Browser supports WebRTC.		
#:	Step actions:	Expected Results:
1	Click on “Call” button	Call management buttons display. Operator is notified.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	High	
Requirements:	CSTC-OCVC-0100	
Keywords:	None	

Test Case WSTC-OCVC-0900: End call		
Author:	jgrilo	
Summary: The purpose of this test case is to verify that customer can end a call.		
Preconditions: 1. Operator answered the call request		
#:	Step actions:	Expected Results:

1	Click on “End” button	Call management buttons are removed.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	CSTC-OCVC-0800	
Keywords:	None	

Test Case WSTC-OCVC-1000: Mute audio stream

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that customers can mute and unmute his audio stream correctly.		
Preconditions: 1. Operator answered the call request. 2. Audio stream is unmuted.		
#:	Step actions:	Expected Results:
1	Click on “Mute/Unmute” button.	Button UI is updated. Audio stream stops.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	CSTC-OCVC-0800	
Keywords:	None	

Test Case WSTC-OCVC-1100: Unmute audio stream

Author:	jgrilo	
Summary: The purpose of this test case is to ensure that customers can mute and unmute his audio stream correctly.		
Preconditions: 1. Operator answered the call request. 2. Audio stream is muted.		
#:	Step actions:	Expected Results:
1	Click on “Mute/Unmute” button.	Button UI is updated. Audio stream starts.
Execution type:	Manual	
Estimated exec. duration (min):		
Importance:	Medium	
Requirements:	CSTC-OCVC-0800	
Keywords:	None	