

Mestrado em Engenharia Informática  
Dissertação/Estágio 2015/2016  
Relatório Final

# Solução de Chamadas em Conferência na Web usando WebRTC

André Perdigão da Costa de Sá Gonçalves  
apcosta@student.dei.uc.pt

Orientadores:  
Ernesto Costa (DEI)  
Tiago Leitão (WIT Software)

Data: 1 de Julho de 2016



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



**Departamento de Engenharia Informática**  
Faculdade de Ciências e Tecnologia  
Universidade de Coimbra  
Pólo II, Pinhal de Marrocos, 3030-290 Coimbra

Tel: +351239790000 | Fax: +351239701266 | [info@dei.uc.pt](mailto: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](mailto:info@wit-software.com)

**ESTAGIÁRIO:**

**André Perdigão da Costa de Sá Gonçalves**  
[apcosta@student.dei.uc.pt](mailto:apcosta@student.dei.uc.pt), [andre.goncalves@wit-software.com](mailto:andre.goncalves@wit-software.com)

**ORIENTADOR DEI:**

**Ernesto Costa**  
[esrnesto@dei.uc.pt](mailto:esrnesto@dei.uc.pt)

**ORIENTADOR WIT SOFTWARE:**

**Tiago Leitão**  
[tiago.leitao@wit-software.com](mailto:tiago.leitao@wit-software.com)

**JURI:**

**Luis Filipe C. Paquete**  
[paquete@dei.uc.pt](mailto:paquete@dei.uc.pt)

**Carlos Bento**  
[bento@dei.uc.pt](mailto:bento@dei.uc.pt)



## Agradecimentos

Gostaria de agradecer toda a ajuda disponibilizada a quem contribuiu diretamente para este estágio, Professor Ernest Costa, meu orientador no Departamento de Engenharia Informática, Engenheiros Tiago Leitão e André Silva da WIT Software que me seguiram ao longo do estágio e me aconselharam em alturas críticas.

A WIT Software também está de parabéns pelos estágios disponibilizados e pelas excelentes condições de trabalho que proporcionam, permitindo alcançar os objetivos do estágio.

Em último, mas não menos importante, um agradecimento especial à minha família que me apoia desde sempre e que me dá asas num mundo cheio de oportunidades. Aos meus amigos de longa data e a todos os restantes estagiários que percorreram também um caminho semelhante na empresa.



## Resumo

O uso de *software* nativo é cada vez menor devido à evolução contínua dos *web browsers*. Existe uma grande quantidade aplicativos que podem ser acedidos com recurso a esta ferramenta, sem a necessidade de *software* adicional. No mundo das videoconferências e da transmissão de multimédia, os navegadores também não ficaram para trás, e com recurso ao *WebRTC* já é possível realizar comunicação áudio e vídeo.

A tecnologia *WebRTC* veio oferecer novas possibilidades de comunicação, fazendo concorrência a serviços e plataformas já estabelecidos. Apesar da comunicação entre navegadores poder ser conseguida facilmente, seria vantajoso haver integração com estas plataformas, como *IP Phones* e dispositivos da rede celular ou terrestre, fazendo uso do protocolo *SIP* que já é utilizado por estas plataformas.

Neste relatório serão apresentados estudos das tecnologias, como *SIP* e *WebRTC*, comparações entre serviços de conferência já estabelecidos e quais os seus processos para a gestão de conferências, desenvolvendo um serviço com as mesmas capacidades, mas com a vantagem de permitir a participação na chamada em conferência a partir de múltiplas plataformas, facilitando o acesso a chamadas em conferência. O resultado é um produto novo e inovador dentro da empresa, que pode ajudar os trabalhadores a melhor realizarem o seu trabalho.



# Índice

Agradecimentos.....	4
Resumo .....	6
Índice.....	8
I. Índice de Figuras.....	11
II. Glossário .....	13
III. Acrónimos .....	14
1. Introdução .....	16
1.1. Contexto e motivação .....	16
1.2. Objetivos .....	16
1.2.1. Objetivos técnicos .....	16
1.2.2. Objetivos do Estágio.....	18
1.2.3. Objetivos da Empresa .....	18
1.3. Estágio.....	18
1.4. Organização do documento.....	19
2. Estado da Arte .....	20
2.1. WebRTC.....	20
2.1.1. <i>Signaling</i> .....	20
2.1.2. Protocolos.....	21
2.1.3. Suporte nos <i>web browsers</i> .....	23
2.1.4. Object RTC (ORTC) .....	26
2.2. Arquiteturas de conferências <i>web</i> .....	27
2.2.1. Topologia .....	27
2.2.2. Descoberta de pares .....	29
2.3. Serviços de conferências <i>web</i> .....	29
2.3.1. Funcionalidades .....	29
2.3.2. Competidores .....	30
2.3.3. Tabela comparativa .....	35
3. Abordagem .....	36
3.1. Metodologia .....	36
3.2. Planeamento.....	36
3.3. Sprints de <i>Scrum</i> .....	37



3.4.	Riscos .....	40
4.	Desenvolvimento.....	42
4.1.	Arquitetura de Software .....	42
4.1.1.	Servidor ( <i>backend</i> ) .....	43
4.1.2.	Aplicação web React ( <i>frontend</i> ) .....	44
4.2.	Tecnologias .....	46
4.2.1.	<i>Web sockets</i> .....	46
4.2.2.	WebRTC .....	47
4.2.3.	<i>Kurento Media Server</i> .....	49
4.2.4.	SIP .....	52
4.3.	Dificuldades .....	55
5.	Resultados .....	57
5.1.	Conferência entre clientes <i>web</i> .....	57
5.2.	Conferência entre clientes <i>web</i> e <i>VoIP</i> .....	58
5.3.	Funcionalidades não implementadas .....	59
6.	Qualidade de Software .....	61
6.1.	Testes Funcionais.....	61
6.1.1.	Testes de Aceitação .....	61
6.2.	Testes de Qualidade.....	62
6.2.1.	Testes de Usabilidade.....	62
6.2.2.	Testes de Performance .....	63
7.	Conclusão.....	68
7.1.	Visão geral .....	68
7.2.	Trabalho futuro .....	69
7.3.	Considerações finais .....	69
	Referências .....	71
	Apêndice.....	73
	A – <i>Screenshots</i> de aplicações de conferência .....	73
	B – Imagens e Diagramas.....	78
	Anexos .....	80
	A – Abordagem.....	80
	B – Arquitetura.....	80
	C – Qualidade de Software .....	80



## I. Índice de Figuras

Figura 1-1 - Diagrama da solução a desenvolver - conferências multiponto com suporte a SIP .....	17
Figura 2-1 - Sinalização entre dois pares.....	20
Figura 2-2 - Sinalização e NAT Traversal com recurso aos servidores STUN e TURN [4] .....	21
Figura 2-3 - Pacote SDP com a descrição das várias secções [5].....	22
Figura 2-4 - Cabeçalho de um pacote RTP .....	22
Figura 2-5 - Percentagem global de uso por navegador desktop (Setembro de 2015) [8]....	24
Figura 2-6 - Percentagem global de uso por navegador móvel (Setembro de 2015) [9] .....	24
Figura 2-7 - Funcionalidades implementadas da API WebRTC por browser (Janeiro de 2016) [10] .....	25
Figura 2-8 - Funcionalidades implementadas da API WebRTC por browser (Junho de 2016) [10] .....	26
Figura 2-9 - Composição de streams em apenas uma (composite) .....	28
Figura 3-1 - Diagrama Gantt do planeamento do estágio ao nível dos Sprints .....	38
Figura 4-1 - Diagrama da arquitetura ao nível dos contentores.....	43
Figura 4-2 - Descrição do componente Web Server.....	44
Figura 4-3 - Arquitetura genérica Flux usada em várias aplicações React.....	45
Figura 4-4 - Arquitetura da aplicação web em ReactJS (os diferentes Componentes das vistas encontram-se omitidos) .....	46
Figura 4-5 - Diagrama de fluxo para o estabelecimento de uma conferência com WebRTC [13] .....	48
Figura 4-6 - Media Pipeline conectando 3 clientes WebRTC, sendo o Master o emissor de média [15] .....	49
Figura 4-7 - Protocolo de comunicação em tempo real em Web Browsers e dispositivos celulares (ou IP Phones) .....	50
Figura 4-8 - Streams de media entre os vários participantes .....	50
Figura 4-9 – Pipeline com as ligações internas entre os endpoints, criadas com recurso à API do Kurento (não se encontram representadas as ligações do endpoint D) .....	51
Figura 4-10 - Múltiplas instâncias de servidores Kurento com lotação individual .....	52
Figura 4-11 - Diagrama de sequência para um pedido de INVITE.....	53
Figura 4-12 - Diagrama da comunicação entre a solução desenvolvida e participante SIP .....	54
Figura 4-13 - WTT WebRTC deployment architecture [16].....	54
Figura 4-14 - Diagrama de sequência de convite de um número de telefónico .....	55
Figura 5-1 - Sala de conferência resultante da implementação do 1º semestre .....	57
Figura 5-2 - Sala de conferência resultante da implementação do 2º semestre .....	58
Figura 5-3 - Ecrã de convite de números telefónicos .....	59
Figura 5-4 - Ecrã de informação/configuração da conferência.....	59
Figura 6-1 - Percentagem de uso de CPU do Chrome em função do número de participantes e topologia.....	65
Figura 6-2 - Largura de banda utilizada pelo servidor multimédia (em Mbps) com diferentes participantes conectados.....	66

Figura 6-3 – Percentagem de uso do cpu pelo processo Kurento em função do nº de participantes .....	66
Figura 0-1 - Apppear.in: Ecrã principal.....	73
Figura 0-2 - Apppear.in: sala de conferência .....	73
Figura 0-3 - Apppear.in: Sala de conferência com 2 participantes.....	74
Figura 0-4 - Jitsi Meet: sala de conferência.....	74
Figura 0-5 - Talky.io: ecrã principal .....	75
Figura 0-6 - Talky.io: ecrã pré-conferência.....	75
Figura 0-7 - Talky.io: sala de conferência.....	76
Figura 0-8 - GoToMeeting: ecrã principal.....	76
Figura 0-9 - GoToMeeting: sala de conferência .....	77

## II. Glossário

Termos	Definição
<b>API</b>	An application programming interface is a set of rules and specifications that software programs can follow to communicate with each other. It serves as an interface between different software programs and facilitates their interaction
<b>G.711/PCM</b>	Narrowband audio codec that provides toll-quality audio, primarily used in telephony
<b>H.264</b>	Also known as MPEG-4 AVC, it's a video coding format
<b>Kurento Media Server</b>	WebRTC media server that allows transcoding, recording, mixing, broadcasting and routing of audiovisual flows
<b>Long-polling</b>	Mechanism for the server to independently send, or push, data to the client without the client first making a request
<b>NAT traversal</b>	Mechanism to establish and maintain IP connections across gateways behind NAT. Required for realtime network applications, like VoIP
<b>Opus</b>	Lossy audio coding format that is particularly suitable for interactive real-time applications over the Internet
<b>SIP Trunking</b>	It is the use of VoIP to facilitate the connection of a PBX to the internet. It is a direct connection between your organization and an Internet telephony service provider
<b>VP8</b>	Video compression format developed by Google supported by the major browsers
<b>WebRTC</b>	Open project that provides browsers and mobile applications with Real-Time Communications (RTC) capabilities for the web

### III. Acrónimos

Termos	Definição
<b>ABR</b>	Adaptive bitrate streaming
<b>API</b>	Application Programming Interface
<b>DOM</b>	Document Object Model
<b>DTMF</b>	Dual-Tone Multi-Frequency
<b>ICE</b>	Interactive Connectivity Establishment
<b>MCU</b>	Multipoint Control Unit
<b>MVC</b>	Model View Controller
<b>NAT</b>	Network address translation
<b>OTT</b>	Over the Top
<b>P2P</b>	Peer to peer
<b>PBX</b>	Private Branch Exchange
<b>PSTN</b>	Public Switched Telephone Network
<b>RTCP</b>	RTP Control Protocol
<b>RTP</b>	Real-Time Transmission Protocol
<b>SaaS</b>	Software as a Service
<b>SDP</b>	Session Description Protocol
<b>SIP</b>	Session Initiation Protocol
<b>SRTP</b>	Secure Real-time Transport Protocol
<b>STUN</b>	Session Traversal Utilities for NAT
<b>TURN</b>	Traversal Using Relay NAT
<b>XMPP</b>	Extensible Messaging and Presence Protocol



## 1. Introdução

### 1.1. Contexto e motivação

Serviços de videoconferência existem desde inícios da década de 90 [1] e são, nos dias de hoje, bastante utilizados em diversas áreas, tanto a um nível pessoal como empresarial. A premissa do serviço é bastante simples, permitir conferências áudio e vídeo entre vários utilizadores por recurso a um *software* proprietário instalado no sistema ou um browser com a tecnologia Adobe Flash.

Apesar de existirem diversos serviços de videoconferência, as principais funcionalidades são partilhadas pela maioria, como vídeo em tempo real, *VoIP*, *chat* de texto, partilha de ecrã, entre outros.

WebRTC foi tornado público em 2011 pela gigante das pesquisas Google [2], e desde então que tem impulsionado novos tipos de serviços inteiramente funcionais a partir de um navegador *web*, assim como serviços de videoconferência.

Com o advento da internet surgiram novas formas de comunicação. Aplicações OTT utilizam as infraestruturas dos operadores para disponibilizar inúmeros serviços, muitos deles competindo contra serviços semelhantes prestados pelas mesmas operadoras.

Comunicar via áudio ou vídeo através da internet tornou-se uma tarefa recorrente para qualquer utilizador, seja numa comunicação entre 2 utilizadores ou mais. Este tipo de conferências na internet é mais barato para o utilizador, a infraestrutura já se encontra montada, os custos de operação são inferiores e permite disponibilizar ainda mais funcionalidades. Através da internet e de um simples navegador (*browser*), é possível usufruir de inúmeros serviços e as conferências são um deles.

Utilizando novas tecnologias no *browser* já é possível trocar dados media entre vários pontos sem qualquer recurso a software adicional ou *plugin*, através da utilização da tecnologia *WebRTC*.

### 1.2. Objetivos

Um estágio é, maioritariamente, o primeiro contato com o mercado de trabalho para um estudante. Este proporciona uma excelente oportunidade de o estudante consolidar os conhecimentos aprendidos ao longo dos anos e perceber como o desenvolvimento de *software* é de facto executado.

#### 1.2.1. Objetivos técnicos

O objetivo do projeto é criar um protótipo de um serviço de conferências de voz e vídeo que permita controlar os vários participantes e como estes interagem na conferência.



Utilizando tecnologias atuais, como o *WebRTC*, foram criados mecanismos que permitam a estes utilizadores conferenciarem independentemente da plataforma em que se encontram, integrando participantes de navegadores *web* a clientes SIP (ex. *IP Phones*). Foram também analisados vários modos de efetuar conferências, disponibilizando também interfaces gráficas para o controlo e gestão do serviço de conferências.

Como ponto de partida foram estudados serviços atuais de conferências na web como *appear.in*, *UberConference*, entre outros, juntamente com a tecnologia que suporta estes serviços, *WebRTC*. Foram definidas funcionalidades base para o serviço e posteriormente acrescentadas outras que contribuam mais para o valor deste projeto, como a integração *SIP*.

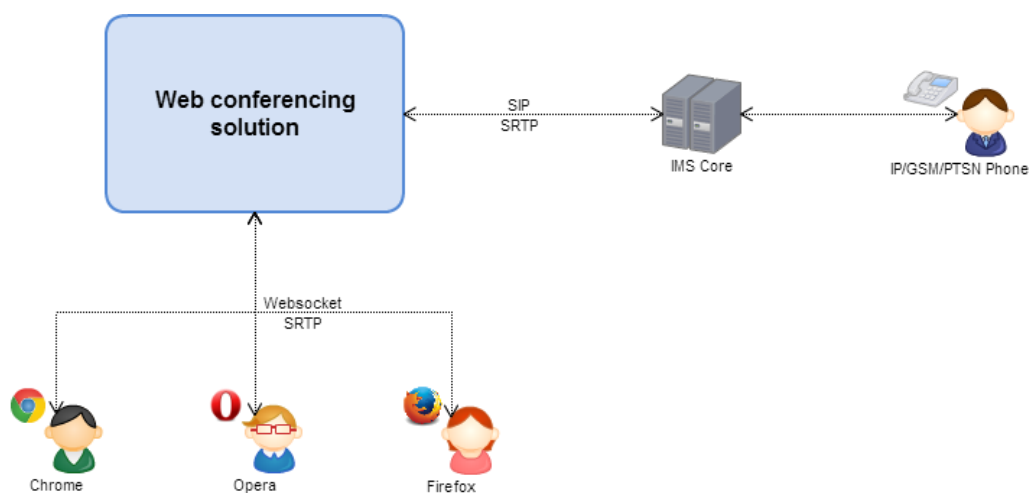


Figura 1-1 - Diagrama da solução a desenvolver - conferências multiponto com suporte a SIP

Como representado na Figura 1-1, pretende-se desenvolver um serviço de conferências que ofereça suporte a clientes em *web browsers* assim como a clientes fora da *web*, como telemóveis. Pretende-se criar um serviço de fácil acesso e utilização simplificada, mantendo as funcionalidades esperadas em conferências e inovando também ao integrar participantes fora da *web*. Este projeto, apesar de não ser totalmente inovador, pretende contribuir para o valor da empresa e facilitar chamadas em conferência primariamente dentro da empresa.

Desta forma, o conjunto de funcionalidades de alto nível propostas para este projeto são as seguintes:

- **Chamadas em conferência na web:** a partir de um navegador web (Chrome) é possível criar ou juntar a uma sala de conferência, sendo possível participar numa videoconferência com outros participantes (na web ou SIP)
- **Chamadas em conferência com suporte para SIP:** participantes num dispositivo celular ou IP podem juntar-se a uma conferência através de SIP. Estes participantes podem apenas realizar audioconferência
- **Funcionalidades de moderação:** Utilizadores com privilégio têm capacidades básicas de moderação, como a expulsão de participantes

- **Identificação em tempo-real de um participante em atividade:** Quando um participante se encontra em atividade (está a comunicar) é dada informação visual para ser facilmente identificado numa conferência em grupo

### 1.2.2. Objetivos do Estágio

O principal objetivo do estágio é consolidar os conhecimentos adquiridos ao longo do percurso académico decorrido no Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra, no mestrado de Engenharia Informática no ramo de Engenharia de Software. O estágio proporciona uma oportunidade de colocar em prática o conhecimento adquirido, desenvolvendo um *software* real com base numa necessidade real.

### 1.2.3. Objetivos da Empresa

A WIT Software tem como objetivo desenvolver os estagiário, passando conhecimentos de como desenvolver *software* com qualidade. Perante isto, o objetivo principal será a finalização do projeto com sucesso, ou seja, que os objetivos apresentados à Universidade tenham sido atingidos.

A empresa pretende investir no trabalho do estagiário de forma a enriquecer o seu leque de produtos, ajudando-a a diferenciar dos competidores.

## 1.3. Estágio

O estágio decorreu na empresa WIT Software, localizada em Coimbra, Portugal, especializada na área das telecomunicações móveis [3]. Foi fundada em 2001 com sede no Instituto Pedro Nunes (IPN). A WIT Software possui clientes em todo o mundo e está continuamente a desenvolver produtos e ideias inovadoras que possam criar uma vantagem competitiva.

Este estágio encontra-se dividido em duas componentes, uma de integração na empresa e aos produtos e tecnologias que são trabalhados e uma componente de desenvolvimento de projeto onde foram aplicadas várias técnicas e métodos de trabalho. Estas duas componentes são muito importantes e servirão para consolidar os conhecimentos adquiridos ao longo do meu percurso académico, mais especificamente aos processos de engenharia de *software*.

No percurso do estágio foram adquiridos conhecimentos sobre *WebRTC*, protocolos de transmissão e sinalização de *media*, *frameworks* de *frontend* para a web, e toda a infraestrutura que permite desenvolver um serviço de videoconferência. O objetivo do estágio será desenvolver um produto novo que expanda o leque já abrangente que a empresa tem, contribuindo para o seu crescimento.

Para o correto desenvolvimento do produto é necessário o planeamento de uma estratégia que considere os vários objetivos. Este planeamento inclui análise de requisitos e definição de arquitetura.

### 1.4. Organização do documento

Este documento encontra-se organizado com a seguinte estrutura:

2. **Estado da arte** – Análise das tecnologias existentes que suportam as conferências na web, as suas arquiteturas e protocolos, sendo apresentada uma comparação de vários serviços no mercado que possuem funcionalidades semelhantes
3. **Abordagem** – Descreve a abordagem utilizada, juntamente com o planeamento da análise e desenvolvimento do projeto, descrevendo o pretendido no conjunto dos dois semestres, fazendo uma breve referência às funcionalidades a implementar
4. **Desenvolvimento** – Descrição da arquitetura implementada, juntamente com as decisões que levaram à sua decisão assim como a descrição da implementação de alguns componentes
5. **Resultados** – Apresenta e descreve os resultados obtidos durante o estágio
6. **Qualidade de Software** – Descreve as estratégias utilizadas para avaliar e garantir a qualidade do *software* produzido, como testes funcionais e de usabilidade
7. **Conclusão** – Esta secção sumariza o trabalho realizado no semestre, colocando o foco nos pontos principais, fazendo também uma retrospectiva do estágio e trabalho futuro.

O trabalho descrito neste relatório é complementado com os seguintes anexos:

- **Anexo A – Abordagem:** descreve a metodologia utilizada neste projeto
- **Anexo B – Arquitetura:** contém a informação completa da arquitetura
- **Anexo C – Qualidade de Software:** descrição de testes funcionais e de usabilidade

## 2. Estado da Arte

### 2.1. WebRTC

WebRTC é uma API implementada pelos navegadores que permite transmissão de áudio, vídeo e outro tipo de dados, sem necessitar de serviços adicionais para o suportar, criando comunicações P2P. Apesar da implementação desta tecnologia ser ainda recente, a grande maioria dos navegadores atuais já suportam este tipo de comunicação.

Apesar das inúmeras possibilidades que esta tecnologia permite, a grande vantagem é não requerer qualquer *plugin* ou *software* adicional, permitindo que comunicações P2P em tempo real estejam ativas por defeito.

A implementação do WebRTC implica que os diversos navegadores utilizem *codecs* de áudio e vídeo específicos para transmitir a informação em tempo real.

#### 2.1.1. Signaling

Para que dois pares (*peers*) possam estabelecer uma ligação entre eles, é necessário recorrer a um servidor de *signaling* (Figura 2-1) para que as suas localizações na rede possam ser conhecidas por ambos. Desta forma os pares recebem o endereço na rede de cada um e podem agora negociar uma ligação direta.

Os navegadores utilizam a *framework* ICE para estabelecer esta ligação. O ICE tenta descobrir o melhor caminho na rede através de um servidor *STUN*.

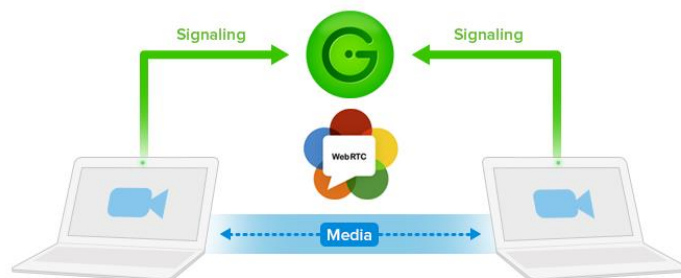


Figura 2-1 - Sinalização entre dois pares

Por vezes o canal de comunicação direto é impossível de ser estabelecido pelos pares e nestas situações é necessário fazer uso de um servidor adicional denominado de *TURN*. Servidores *TURN* atuam como intermediário na troca de dados entre os pares na eventual situação dos clientes estarem por detrás de uma *NAT* (Figura 2-2).

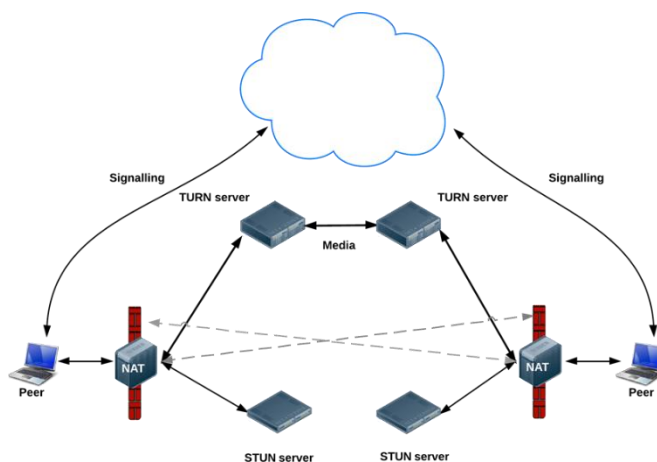


Figura 2-2 - Sinalização e NAT Traversal com recurso aos servidores STUN e TURN [4]

Esta troca de informação entre o cliente e o servidor de *signaling* é usualmente efetuada através de *websockets*, canais de comunicação bidirecionais e instantâneos.

## 2.1.2. Protocolos

### 2.1.2.1. SDP

*Session Description Protocol* é usado para descrever os parâmetros de inicialização de uma sessão entre dois pares. Os conteúdos transmitidos neste protocolo podem ser facilmente lidos pois são descritos em texto.

É usado para iniciar ou parar fluxos de media (*streams*) ou modificar os parâmetros da sessão, como *codecs*, endereços de IP e portas. Este protocolo não é usado para a transmissão da *media*, como o protocolo *RTP* descrito na secção seguinte.

Um pacote SDP descreve as capacidades de determinado dispositivo o que implica a troca de pelo menos dois pacotes SDP para o início de uma sessão entre dois pontos. A Figura 2-3 mostra um exemplo de um pacote *SDP*, contendo informação da rede, *media* a ser estabelecida e suas capacidades assim como candidatos *ICE*.

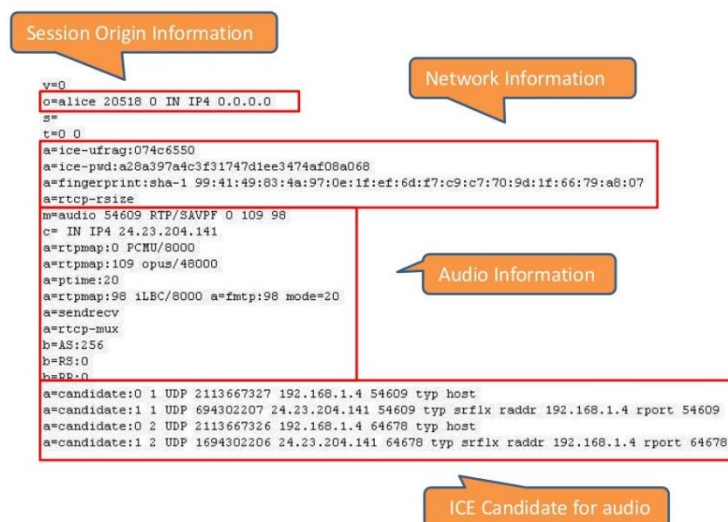


Figura 2-3 - Pacote SDP com a descrição das várias seções [5]

Aquando da receção do pacote SDP pelas duas entidades, ambas podem agora acordar nos *codecs* a utilizar pois têm conhecimento do que o participante remoto suporta e o seu endereço na rede. A partir deste momento é possível estabelecer as ligações que irão transportar a *media* entre os pares, através do protocolo RTP.

#### 2.1.2.2. RTP

RTP, ou *Real-Time Transport Protocol*, é um protocolo que especifica como é que transmissões de dados multimédia em tempo real devem ser geridos. Este protocolo encontra-se especificado no RFC 1889 [6].

A responsabilidade do protocolo é de apenas transmitir a informação, mas para garantir Qualidade de Serviço é usado juntamente com o protocolo *RTCP* (*RTP Control Protocol*), ajudando na sincronização das diferentes *streams*, gerindo a perda de pacotes e atrasos na recepção de informação (*jitter*).

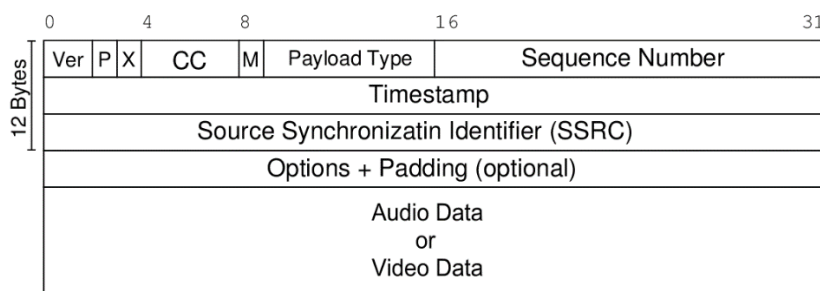


Figura 2-4 - Cabeçalho de um pacote RTP

A Figura 2-4 representa o cabeçalho de um pacote *RTP*. Este contém vários componentes como o número de sequência, usado para detetar pacotes perdidos, identificação do *payload*, descrevendo o tipo de codificação da media, identificação da origem da *stream*, entre outros.

### 2.1.2.3. *SIP*

*SIP* é um protocolo de sinalização usado para gerir sessões de comunicação numa rede IP. Uma sessão pode ser uma chamada entre dois participantes ou uma sessão de conferência colaborativa.

*SIP* é um protocolo bastante flexível e bastante detalhado. Foi desenhado com o propósito de estabelecer sessões multimédia entre um grupo de participantes. É possível estabelecer simples chamadas telefónicas como também sessões de videoconferência ou de mensagens instantâneas (IM). *SIP* é similar ao protocolo HTTP e permite gerir chamadas *VoIP* recorrendo a outros protocolos, como o *RTP* descrito anteriormente para a troca de pacotes em tempo real.

Neste projeto foi utilizado este protocolo para realizar comunicação com dispositivos fora da *web*, como *IP Phones* ou telemóveis que se encontram na rede celular. A secção 4.2.4 refere como é que este protocolo é usado no contexto do estágio.

### 2.1.3. Suporte nos *web browsers*

A tecnologia *WebRTC* ainda não se encontra disponível em todos os *browsers*. À data de escrita deste relatório, *WebRTC* é suportado por três *browsers desktop*: Chrome, Firefox e Opera. Esta tecnologia é suportada em navegadores móveis pelo *Chrome for Android* e *Android Browser* (> v4.4.4). Este suporte da tecnologia pelos *browsers* pode ser traduzido em percentagem para uma melhor compreensão, e o suporte global da tecnologia *WebRTC* é de 56.88% [7]. Mais de metade dos navegadores já suportam esta tecnologia e, referindo a figura seguinte (Figura 2-5), é possível confirmar que 70% dos utilizadores podem usufruir da tecnologia *WebRTC*, considerando a percentagem global de utilização dos navegadores Chrome, Firefox e Opera.

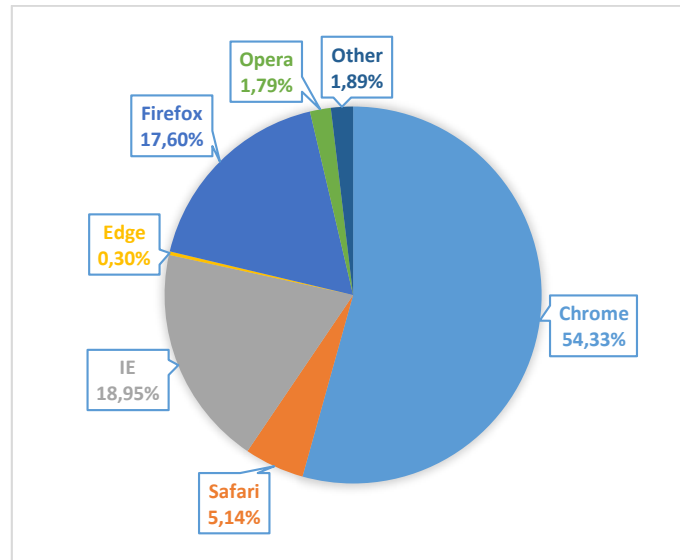


Figura 2-5 - Percentagem global de uso por navegador desktop (Setembro de 2015) [8]

Considerando apenas utilizadores em dispositivos móveis, confirmando com os dados da Figura 2-6, 36% utilizam *Chrome for Android* que já suporta WebRTC, assim como *Opera Mini*.

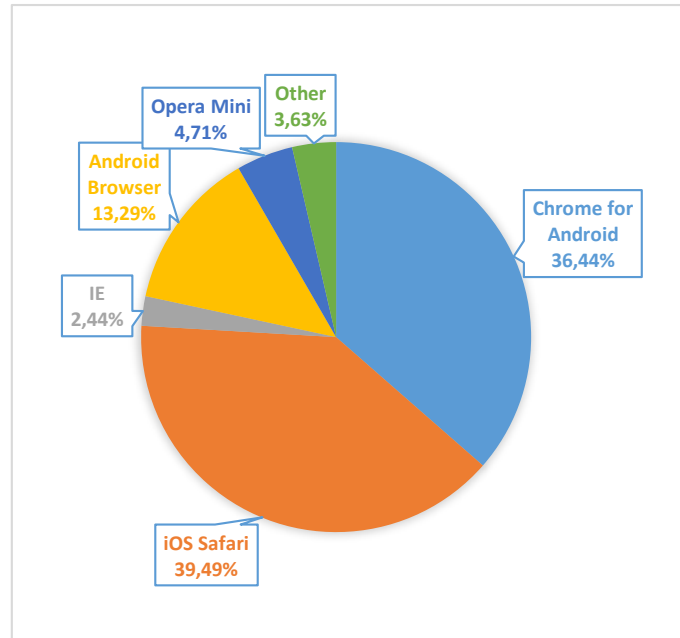
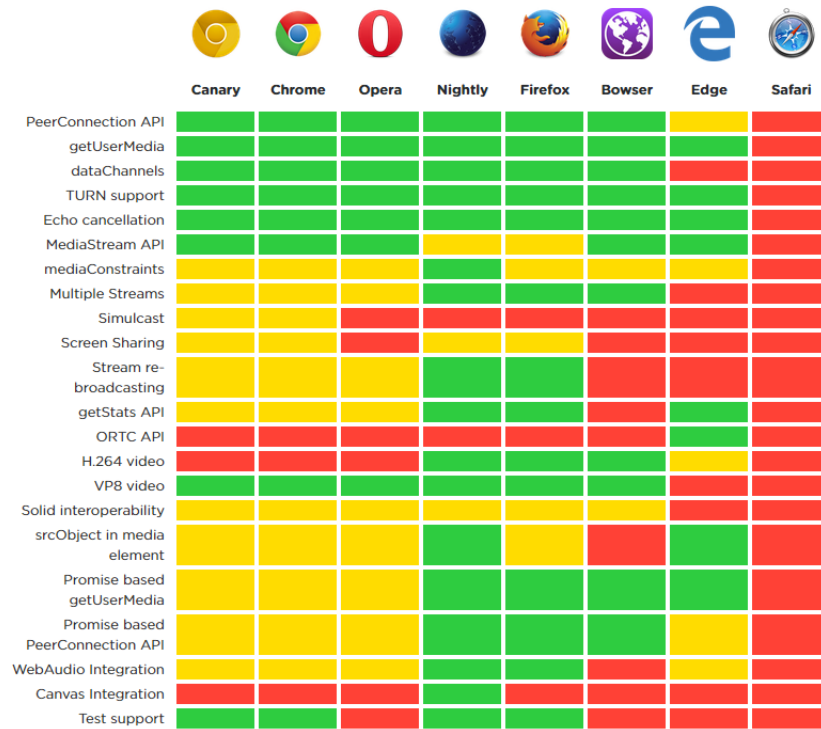


Figura 2-6 - Percentagem global de uso por navegador móvel (Setembro de 2015) [9]

Para utilizadores dos navegadores Safari e Internet Explorer, existem alternativas que permitem fazer uso de WebRTC, através da instalação de *plugins*. O serviço “Is WebRTC Ready Yet?” [10] fornece informação de como esta API se encontra disponível pelos



diferentes navegadores e quais as funcionalidades implementadas. A figura seguinte (Figura 2-7 e Figura 2-8) é um *snapshot* do serviço, à data de escrita do relatório.



**Completion Score: 54.5%**

Figura 2-7 - Funcionalidades implementadas da API WebRTC por browser (Janeiro de 2016) [10]

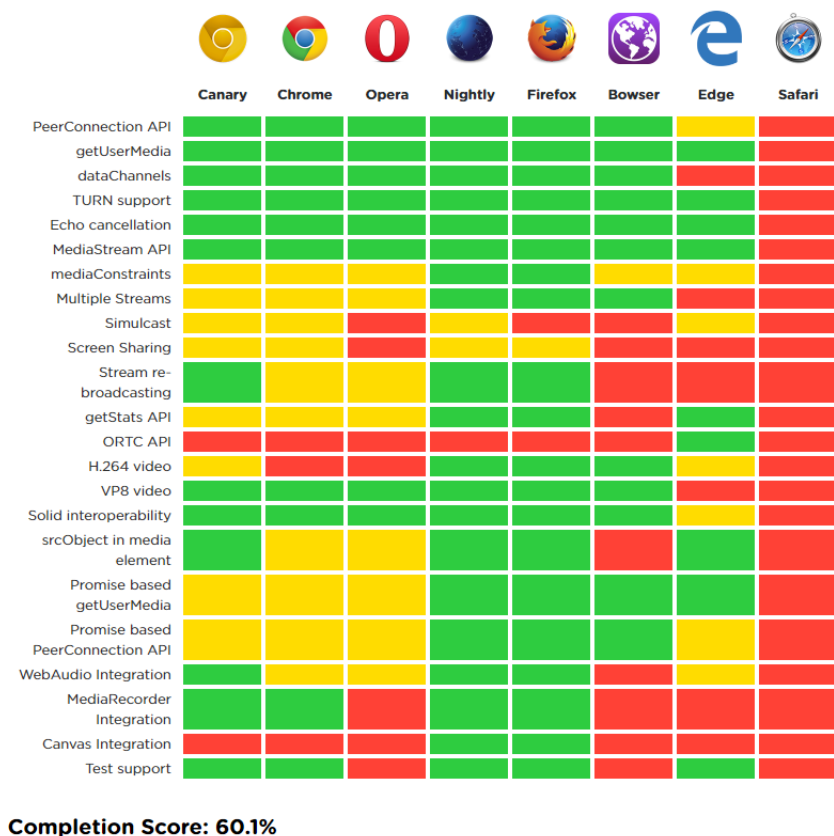


Figura 2-8 - Funcionalidades implementadas da API WebRTC por browser (Junho de 2016) [10]

#### 2.1.4.Object RTC (ORTC)

*Object RTC* é uma iniciativa da empresa Hookflash para criar mecanismos de comunicação em tempo real nos navegadores web, semelhante ao *WebRTC*. Estas duas APIs servem o mesmo propósito mas o seu funcionamento é ligeiramente diferente, estando o *ORTC* melhor otimizado, trocando a implementação de alguns mecanismos que assentam hoje por base em praticamente todos os sistemas de comunicação em tempo real, sendo um destes o protocolo SDP para a troca de informação entre os clientes. Robin Raymond, autor original do *ORTC* refere que o protocolo SDP é antiquado e problemático, tendo por base antigas tecnologias (*«The SDP format itself is arcane and rooted in old world legacy reasoning»*).

À data de escrita do relatório, o único browser que suporta esta nova API de comunicação em tempo real é o *Microsoft Edge*. A Google também pretende contribuir para o desenvolvimento desta nova API e o *WebRTC* não será substituído mas irá sim adotar internamente estes novos mecanismos [11]. *ORTC* é por vezes referido como *WebRTC 1.1* e considerado por muitos uma evolução ao *WebRTC 1.0*. Alguns dos conceitos de *ORTC* já foram transferidos para o *WebRTC 1.0* e a Google pretende acrescentar estas funcionalidades ao código fonte do Chrome.

## 2.2. Arquiteturas de conferências web

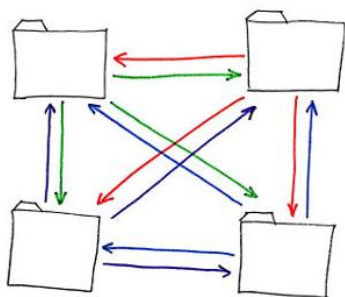
Para permitir realizar conferências na web, existem algumas técnicas e arquiteturas que suportam este tipo de eventos. A comunicação entre dois pares pode ser direta mas para permitir conferências multiponto pode ser adotada uma arquitetura diferente que ofereça melhor suporte a este tipo de conferências.

Esta secção irá descrever diferentes tipos de arquitetura e como é feita a descoberta dos pares.

### 2.2.1. Topologia

Nesta secção serão apresentadas duas topologias, as mais utilizadas, mas existem um conjunto vasto de outras topologias, sendo algumas dessas sub-topologias das apresentadas em baixo.

#### 2.2.1.1. Rede *mesh*



Uma rede *full mesh* é uma rede em que todos os nós se encontram conectados a todos os restantes nós. Do ponto de vista de conferências na web, significa que cada par é responsável por enviar a sua media a todos os restantes pares, assim como receber a media de cada um deles. Este tipo de redes funciona bem para um número reduzido de pares.

A primeira vantagem deste tipo de redes é não requerer um MCU (ver secção 2.2.1.2) para retransmitir a media dos pares, sendo uma abordagem pouco dispendiosa. Por não haver retransmissão dos dados por um servidor central, poderão surgir alguns problemas na qualidade da chamada por ser necessário uma maior largura de banda por cliente.

Neste tipo de situações, o serviço de conferências não permite uma grande variedade de clientes (plataformas). Se for pretendido um serviço de conferências que suporte navegadores como o Chrome, criando uma rede *mesh* completa irá restringir os clientes que se podem juntar a esta conferência pois terão de suportar os mesmos *codecs* de áudio e vídeo que WebRTC permite, assim como o mesmo protocolo de transporte.

#### **Simulação de número de transmissões de media (N participantes na conferência):**

Par n<sup>o</sup>1 envia própria *media* para N-1 pares

Par n<sup>o</sup>1 recebe media de N-1 pares

## 2.2.1.2. MCU

Uma outra solução para conectar diferentes clientes numa conferência web é através da utilização de um servidor MCU. Se um dos requisitos for escalabilidade (suportar um maior número de participantes) e suportar clientes em diferentes plataformas, a utilização de um MCU torna-se obrigatória.

Um MCU retransmite a media dos pares para todos os restantes. Neste tipo de arquitetura o cliente não necessita de tanta largura de banda como numa rede *mesh*. Cada par apenas transmite a sua *media* 1 vez para o MCU, sendo este responsável por fazer a retransmissão.

O MCU pode também juntar as diferentes *streams* de media em apenas uma (*composite*), sendo necessário ainda menos largura de banda. Estas abordagens trazem também algumas desvantagens como menos controlo sobre a conferência por parte dos participantes.

## 2.2.1.2.1. Composite

Existe também o mecanismo de *composite* que agrega várias *streams* (áudio ou vídeo) em apenas uma. Quando este mecanismo é utilizado em áudio denomina-se de *mixing* (ver capítulo 4.2.3). Uma das vantagens de utilizar este processo é a redução da largura de banda necessária dado que várias *streams* são agregadas em apenas uma em que o *bitrate* é muito inferior ao cumulativo das *streams*, podendo o serviço escalar melhor em conferências com mais participantes.

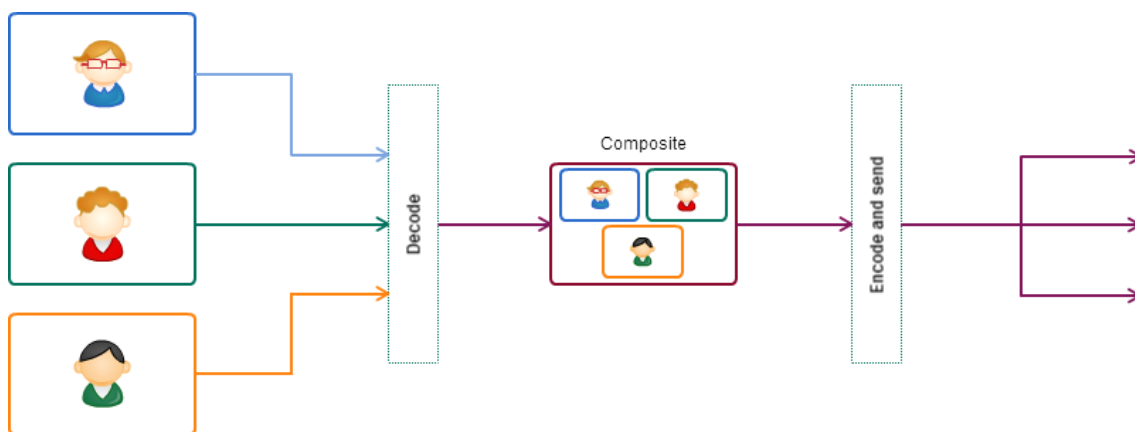


Figura 2-9 - Composição de streams em apenas uma (*composite*)

A Figura 2-9 representa como esta composição é realizada. O servidor recebe o vídeo de vários participantes, descodifica-os (*decode*) e faz a composição da imagem numa espécie de grelha. No final faz a codificação para o formato de cada participante (VP8 no Chrome) e envia. Todos os participantes recebem a mesma stream.

Desvantagens associadas a este mecanismo é a menor qualidade de imagem final e controlo sobre a imagem de cada participante.

### 2.2.2. Descoberta de pares

Como referido anteriormente, para um serviço de conferências é necessário um servidor que troque a informação entre os pares, permitindo-os estabelecer ligações ponto a ponto e comunicar informação. Este mecanismo não é definido pelo WebRTC e deve ser implementado de qualquer forma possível, sem restrições na linguagem de programação. Para concretizar este objetivo pode ser implementado um simples servidor de *websockets* que troca os pacotes SDP e candidatos ICE entre os participantes.

Em outros serviços de conferência, são usados protocolos como o XMPP assim como servidores *Jabber* para trocar toda a informação. Nestes casos, os serviços registam um identificador para a conferência (*jid* ou *jabber id*) e quando os participantes se conectam a informação é enviada para o *jid* da conferência. Usando este protocolo é possível tirar vantagem de servidores XMPP já disponíveis (e gratuitos) que criam uma enorme rede entre eles (protocolo descentralizado).

É necessário que o servidor central troque a informação necessária entre os pares mas este objetivo também pode ser atingido de forma manual, copiando e colando esta informação. Como exemplo temos o repositório *serverless-webrtc* [12] que disponibiliza uma demonstração.

## 2.3. Serviços de conferências web

É possível encontrar inúmeros serviços de conferência gratuitos que fazem uso da tecnologia WebRTC, permitindo videoconferências “*on the go*” (navegador web sem extensões adicionais). Existem outros tipos de serviços que fornecem uma API que tratam de todo o *backend* necessário para uma conferência se realizar, sendo necessário apenas implementar o cliente (*frontend*) da aplicação.

### 2.3.1. Funcionalidades

As principais funcionalidades encontradas em serviços de videoconferência na web são a facilidade de falar em grupo, parar de reproduzir áudio ou vídeo de um participante (assim como *mute* da própria media) e *chat* de texto. Outros serviços possuem funcionalidades mais avançadas como partilha de ficheiros, partilha de ecrã (necessitando de uma extensão) e *whiteboard* para desenhar. Serviços de conferência na web para o mundo empresarial possuem maior interoperabilidade entre participantes através da integração com o protocolo SIP, abrindo maiores possibilidades de conferência.

Relativamente a funcionalidades de moderação existem diferentes abordagens. Alguns serviços permitem que todos os participantes possam expulsar outro participante e outros em que a funcionalidade de expulsar não está presente.

### 2.3.2.Competidores

Para serviços de conferência na web já existem alguns serviços e empresas estabelecidos neste ecossistema. Conferências na web não são uma novidade mas a tecnologia WebRTC é recente e cada vez mais novos serviços são criados à sua volta.

Iremos olhar em maior detalhe em alguns serviços de conferências, começando com competidores diretos (que usam WebRTC) e posteriormente nos competidores indiretos (serviços de conferência legados ou que requerem software proprietário).

#### 2.3.2.1. Diretos

Apesar do WebRTC ser uma tecnologia recente, já existem inúmeros serviços de conferências na web que usufruem desta tecnologia para criar conferências na web fáceis de utilizar e que podem ser acedidas por qualquer utilizador em quase todas os dispositivos, não necessitando também de software específico para usufruir destes serviços. Existem alguns serviços que se encaixam nesta categoria, sendo alguns apenas demonstrações técnicas e não serviços inteiramente funcionais.

Nesta secção iremos analisar alguns destes serviços, descrevendo o seu funcionamento e comparar as suas funcionalidades.

##### 2.3.2.1.1. Appear.in



**URL:** <http://appear.in>

**Company:** Telenor Digital

Este serviço de conferências na web representa o primeiro protótipo que se pretende desenvolver.

É bastante fácil de utilizar o serviço, requerendo apenas o clique de um botão. O utilizador é apresentado com uma caixa de texto onde pode personalizar o nome da sala que deseja criar ou aceita um dos nomes aleatórios. Quando a sala é criada, o utilizador é redireccionado para outra página onde a sua câmara é mostrada, após permitir o acesso aos periféricos. O utilizador pode agora convidar outros participantes através do link único.

Este serviço permite também fechar a sala sendo necessária uma palavra-chave para aceder à conversação. Outra funcionalidade para utilizadores registados é a aquisição do nome da sala para que mais nenhum utilizador o possa usar, e o utilizador que a adquiriu tem alguns privilégios de moderação, como excluir participantes da conversa.

Possui também uma funcionalidade de controlo de qualidade com duas opções: alta ou baixa. Quando uma das opções é seleccionada a página é refrescada com as novas alterações em efeito.

## Funcionalidades

- Não é necessário registo
- Cancelar audio e video de um participante (incluindo o próprio microfone)
- Partilha de ecrã (necessária extensão)
- Fechar sala (palavra-chave para aceder)
- Adquirir sala (sala privada com controlos de moderação – excluir participantes)
- Controlo de qualidade da chamada (alta/baixa)
- Até 8 participantes

### 2.3.2.1.2. Jitsi Meet

**URL:** <https://meet.jit.si/>

**Company:** Jitsi

**License:** Open-source (LGPL)

Jitsi fornece vários *softwares open-source* para sistemas de conferência na web. O seu software principal é o *Jitsi VideoBridge* e é um excelente MCU. O *Jitsi VideoBridge* fornece inúmeras funcionalidades e é um MCU bastantamente escalável, por usar alguns truques para baixar a largura de banda necessária. A Jitsi possui também o seu serviço de conferências na web chamado *Jitsi Meet* e foi desenvolvido sobre o WebRTC e utilizando o seu próprio MCU para a transmissão da media da conversação.



## Funcionalidades

- Não é necessário registo
- Partilha de ecrã (necessária extensão)
- Salas privadas (palavra-chave para aceder)
- Partilha em tempo real de documentos (Etherpad)
- Participantes ilimitados
- *Chat* de texto
- Cancelar audio e video de um participante (incluindo o próprio microfone)

## 2.3.2.1.3. Rabbit



**URL:** <https://rabb.it/>

**Company:** Rabbit, Inc.

Rabbit é outra solução gratuita, inteiramente no *browser* e com a possibilidade de participar através de uma aplicação móvel. Em termos de funcionalidades é bastante semelhante à já discutida *appear.in*.

#### Funcionalidades

- Não é necessário registo (algumas funcionalidades requerem)
- Até 15 participantes
- *Selfcast* (foco principal num utilizador – ecrã maior para esse participante)
- Fechar sala (palavra-chave para aceder)
- *Rabbitcast* (Ecrã controlado remotamente com Netflix, Youtube e outros que são transmitidos para todos os utilizadores)
- *Chat* de texto
- Cancelar audio e video de um participante (incluindo o próprio microfone)
- Aplicação móvel gratuita

## 2.3.2.1.4. Talky



**URL:** <http://talky.io>

**Company:** &yet

Assim como o serviço *appear.in*, o esqueleto deste serviço é bastante semelhante. O foco principal é a facilidade de criar uma nova sala de conferência assim que a aplicação é aberta. É automaticamente sugerido um nome para a sala e basta entrar. Exatamente antes de entrar na página da conferência é mostrada ao utilizador uma previsualização da sua câmara, juntamente com a possibilidade de escolher os dispositivos media (câmara e microfone). Talky suporta até 15 participantes na videoconferência.

#### Funcionalidades

- Não é necessário registo
- Fechar sala (palavra-chave para aceder)
- Partilha de ecrã (necessária extensão)
- Cancelar audio e video de um participante (incluindo o próprio microfone)
- Escolher mostrar apenas o participante em actividade
- Até 15 participantes
- Aplicação iOS grátis



### 2.3.2.2. Indiretos

No contexto do projeto, competidores indiretos são considerados os que ainda não possuem integração com WebRTC, requerendo uma aplicação proprietária, assim como outros serviços que possuam menos funcionalidades.

#### 2.3.2.2.1. WebEx

**URL:** <http://www.webex.com>



**Company:** Cisco

WebEx é um outro excelente serviço de conferências web mais direcionado para o mundo empresarial. Apesar de ser um SaaS, existe também uma versão gratuita que suporta até 3 pessoas numa conferência, possuindo também uma solução On Premises.

Apesar de ser uma solução bastante profissional, é necessário descarregar um software para usufruir do serviço. Existe integração com a maioria dos produtos Cisco e permite que participantes se possam juntar a partir de uma rede celular, marcando um número específico ou recebendo uma chamada da sala.

- **Funcionalidades**
- *Whiteboard*
- Partilha de ecrã
- Gravar reuniões (áudio + vídeo)
- Função de “Apresentador”
- VoIP com *call-in*
- “Call me” para qualquer número
- Sala protegida com palavra-chave
- Controlo remoto de computador
- Aplicação móvel

#### 2.3.2.2.2. UberConference

**URL:** <http://www.uberconference.com>



**Company:** Switch Communications, Inc

UberConference é um serviço de conferências direcionado para o mundo empresarial. Fornece soluções profissionais pagas, sendo um SaaS, mas possui também uma versão gratuita.

Uma das maiores desvantagens deste serviço é a possibilidade de conferenciar apenas por voz. A única funcionalidade de vídeo é através da partilha de ecrã.

## Funcionalidades

- Proteger sala
- Partilha de ecrã (necessária extensão)
- Cancelar audio e video de um participante (incluindo o próprio microfone)
- Join conference from phone (call)
- Gravação de chamadas
- Integração com Dropbox, Google Drive, Box e Evernote
- *Chat* de texto
- Detecção do participante em actividade com *feedback* visual
- Agendamento de reuniões

### 2.3.2.2.3. GoToMeeting

**URL:** <http://www.gotomeeting.com>

**Company:** Citrix Systems, Inc.



Citrix Systems has a few products directed to communication and GoToMeeting is one of them. It is a professional solution and has different tiers for its product according to the number of participants supported. To be able to participate in a conference the user must download a software. As with almost all conference services they offer private rooms, screen sharing VoIP and phone audio and ability to record the sessions as well as a free mobile app.

They also have a free tier for their service and it does not require any software to be installed, it's working fully in-browser. It supports up to 3 people and has the same features as *Appear.in*, making this free version a direct competitor.

## Funcionalidades (versão profissional)

- Salas privadas
- Partilha de ecrã
- VoIP & audio por telemóvel
- Gravar sessões
- Aplicação móvel gratuita









## Funcionalidades (versão gratuita no *browser*)

- Login não necessário
- Até 3 participantes
- Partilha de ecrã (necessária extensão)
- Integração com aplicações de terceiros (Outlook, Slack, etc.)
- Cancelar áudio/vídeo
- *Chat* de texto
- Sala pessoal

### 2.3.3. Tabela comparativa

De forma a melhor comparar os diferentes serviços a tabela seguinte mostra as diferentes funcionalidades disponíveis ou não ao longo destes serviços.

Tabela 1 - Tabela comparativa das funcionalidades de diferentes serviços de conferências

Funcionalidade								
Requer registo	✗	✗	✗	✗	✓	✓	✓	✗
Requer extensão/software	✗	✗	✗	✗	✓	✗	✓	✗
Áudio conferência	✓	✓	✓	✓	✓	✓	✓	✓
Vídeo conferência	✓	✓	✓	✓	✓	✗	✓	✓
Límite <i>users</i>	8	∞	25	15	2 <sup>1</sup> /7 <sup>2</sup>	10 <sup>1</sup> /100 <sup>2</sup>	>5	8
Salas privadas	✓	✓	✓	✓	✓	✓	✓ <sup>2</sup>	✓
Partilha de ecrã	✓	✓	✗	✓	✓	✓	✓	✗
Chat	✓	✓	✓	✓	✓	✓	✓	✓
<i>Call-out</i>	✗	✗	✗	✗	✓ <sup>2</sup>	✓ <sup>2</sup>	✓	✓
<i>Call-in</i>	✗	✗	✗	✗	✓	✓	✓	✓ <sup>3</sup>
Deteção de voz	✗	✓	✗	✗	✗	✓	✗	✓
Gravação	✗	✗	✗	✗	✓	✓	✓ <sup>2</sup>	✗
Mute áudio	✓	✓	✓	✓	✓	✓	✓	✓
Parar vídeo	✓	✓	✓	✓	✓	✗	✓	✓
Controlo de qualidade	✓	✗	✗	✗	✗	✗	✗	✗
Excluir participante	✗	✓	✓	✓	✓	✓	✓	✓
Participante principal	✓	✓	✓	✗	✓	✗	✗	✗

<sup>1</sup> Versão gratuita

<sup>2</sup> Versão paga

<sup>3</sup> Funcionalidade não completada na totalidade

### 3. Abordagem

Como referido anteriormente, pretende-se desenvolver um protótipo que permita criar salas de conferência que possam ser acedidas com um simples *web browser*. A integração com protocolos como o SIP permite criar um protótipo inovador que possibilita comunicações entre múltiplos participantes, tanto a partir de navegadores web como dispositivos celulares ou telefones *IP*.

Esta secção permite perceber, de uma forma breve, que tipo de funcionalidades são esperadas nos dois semestres de estágio e o planeamento dos *sprints* de *Scrum*.

#### 3.1. Metodologia

Este projeto será desenvolvimento utilizando um processo incremental e iterativa, baseado em metodologias ágeis. Esta metodologia opõe-se ao tradicional modelo *Waterfall*, em que existe um planeamento completo de todas as fases de desenvolvimento desde o início do projeto. Como as estimativas para a completação de uma tarefa são, de um modo geral, sobrestimadas ou subestimadas, pode levar a incumprimentos de prazos e a um percurso de desenvolvimento não desejado. No caso de situações não previstas, como aquando o uso de novas ferramentas e tecnologias, pode também resultar em alterações ao plano traçado.

De forma a poder mitigar um imprevisto com maior rapidez e controlo, como o caso de alterações nos requisitos ou dificuldades de desenvolvimento, será adotada uma metodologia ágil, *Scrum*, facilitando o desenvolvimento e acompanhamento do projeto.

#### 3.2. Planeamento

As funcionalidades a desenvolver foram definidas e posteriormente adicionadas ao *Product Backlog* que pode ser consultado no *Anexo A – Abordagem*. Para a definição das funcionalidades utilizou-se a ferramenta de *User Stories* da abordagem ágil. Estas histórias permitem capturar a descrição de uma funcionalidade do ponto de vista do utilizador, de uma forma simples esclarecedora.

Estas funcionalidades foram definidas com base em atuais serviços de conferências *web* de forma a poder fornecer, inicialmente, um serviço semelhante e estudar as suas vantagens e desvantagens. No segundo semestre foram adicionadas as funcionalidades que oferecem ainda mais valor a este protótipo. Consultar o *Anexo B – Arquitetura* que descreve também os requisitos funcionais e não funcionais.

Como referido na secção 2.1.4, existem diferentes topologias implementadas em serviços de conferência *web*, sendo a rede *Mesh* a mais popular nos serviços disponíveis *online*. Apesar das limitações associadas, foi implementada a mesma topologia (1º semestre), permitindo um desenvolvimento mais acelerado, sendo possível mostrar uma conferência em funcionamento atempadamente.

O segundo semestre foi dedicado à integração das funcionalidades mais avançadas (SIP) que permitem integrar qualquer tipo de telefone na conferência. Para esta integração foi necessário um maior estudo das tecnologias e tempo de desenvolvimento.

A seguinte lista mostra as funcionalidades de alto trabalhadas em cada semestre:

### 1º Semestre

- Salas de conferência para participantes na web
- Conferenciar com múltiplos utilizadores (web browser)
- Acesso à conferência através da partilha de um *link*
- Ponto de vista técnico: Java *backend*, React *frontend*, WebRTC (signaling, SDP, Ice Candidates, etc)

### 2º Semestre

- Integração com *Media Server*
- Diferentes topologias para uma sala de conferência (P2P ou *Media Server*)
- Finalização da interface
- Integração do protocolo SIP
  - *Break-out* e *break-out* de chamadas

## 3.3. Sprints de *Scrum*

Seguindo uma metodologia de desenvolvimento ágil como o *Scrum*, os *Sprints* podem ser agendados previamente, oferecendo uma vista geral das várias etapas.

Em cada reunião de *Sprint* realizada antes do seu começo, são definidas as funcionalidades a desenvolver a partir do já existente *Backlog*. Com base na velocidade de desenvolvimento do programador, a quantidade de funcionalidades pode ser estimada, tendo em conta a sua complexidade. Estas tarefas são escolhidas dependendo dos requisitos da empresa, prioridade das tarefas e o tempo disponível. Em cada *Sprint*, o programador inicia o desenvolvimento pegando em uma das tarefas definidas para o *Sprint*.

Os *Sprints* para o estágio foram definidos pela WTT e na figura seguinte (Figura 3-1) é possível ver o seu planeamento.

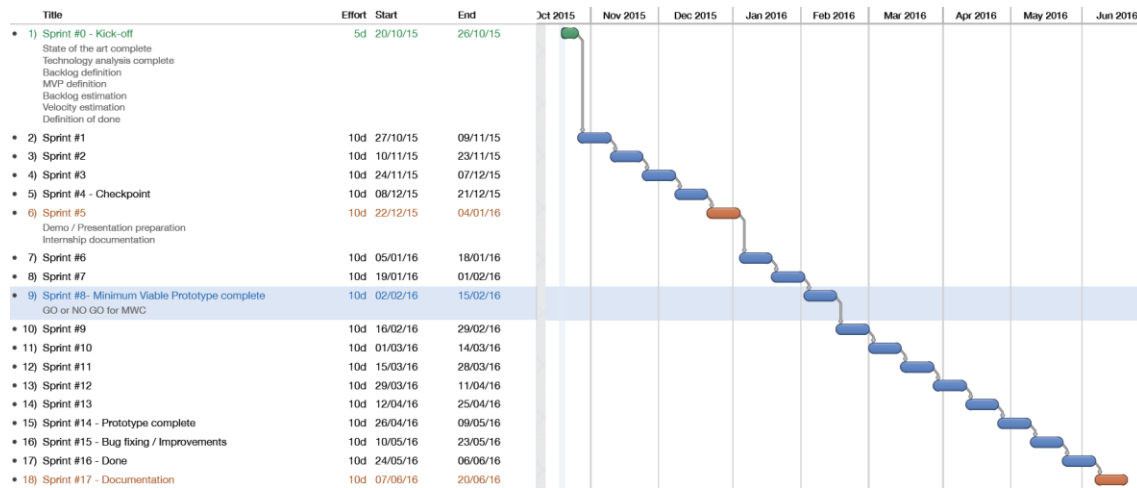


Figura 3-1 - Diagrama Gantt do planeamento do estágio ao nível dos Sprints

As funcionalidades trabalhadas em cada *Sprint* foram registadas e a seguinte lista descreve, de uma forma sucinta, o trabalho realizado ao longo do ano.

- Sprint #0:**  
**Data:** 20-10-2015 a 27-10-2015  
 Análise de tecnologias, definição do MVP juntamente com o *backlog* e estimativa das tarefas
- Sprint #1:**  
**Data:** 27-10-2015 a 10-11-2015  
 Análise de arquitetura e criação do projeto. Página inicial de *mockup* para a criação de salas de conferência. Aplicação em *React* inicializada
- Sprint #2:**  
**Data:** 10-11-2015 a 24-11-2015  
 Definição da interface. Trabalho no servidor de *backend* para suportar a criação de salas. Início do componente de *websockets*.
- Sprint #3:**  
**Data:** 24-11-2015 a 8-12-2015  
 Lógica de controlo de *WebRTC* para permitir conferências entre 2 participantes. Continuação da aplicação em *React*, suportando agora ecrãs de loading e conferências entre 2 participantes.
- Sprint #4:**  
**Data:** 8-12-2015 a 22-12-2015

Possibilidade de definir nome de utilizador, desligar áudio de um participante, moderador excluir participante e mudar tópico. Utilizadores podem agora escolher os dispositivos de média antes de entrar na conferência.

- **Sprint #5:**

**Data:** 22-12-2015 a 5-1-2016

Deteção de voz no cliente, sendo propagado um evento para os restantes participantes pelo servidor.

Finalização da documentação e apresentação intermédia.

- **Sprint #6:**

**Data:** 05-1-2016 a 19-1-2016

Resumo xyz

- **Sprint #7:**

**Data:** 19-1-2016 a 2-2-2016

Investigação de tecnologias de *Media Server*. *Kurento Media Server* foi a tecnologia escolhida para suportar as funcionalidades mais avançadas do serviço de conferências

- **Sprint #8:**

**Data:** 2-6-2016 a 16-2-2016

Integração do *Kurento Media Server* com o serviço (*KurentoService*).

- **Sprint #9:**

**Data:** 16-2-2016 a 1-3-2016

Possibilidade de criar salas de conferência de diferentes topologias.

- **Sprint #10:**

**Data:** 1-3-2016 a 15-3-2016

Estudo da tecnologia SIP e integração do módulo *SipService*. É possível agora fazer chamadas para números de telefone e estes participantes são adicionados à sala de conferência de onde foram convidados

- **Sprint #11:**

**Data:** 15-3-2016 a 29-3-2016

Possibilidade de enviar mensagens de texto dentro de uma conferência, através do chat. Volume dos participantes que se encontram em inatividade é reduzido. Melhorias na interface quando vários clientes se encontram em conferência

- **Sprint #12:**

**Data:** 29-3-2016 a 12-4-2016

Utilizadores podem agora entrar numa conferência em modo de áudio, não enviando o seu vídeo. *Upgrade* automático de uma conferência em topologia P2P para topologia de Media Server quando um participante *SIP* é convidado.

- **Sprint #13:**  
**Data:** 12-4-2016 a 26-4-2016  
Mensagem de alerta quando browser não é suportado. Resolução de *bugs* e melhoramentos na estabilidade da aplicação
- **Sprint #14:**  
**Data:** 26-4-2016 a 10-5-2016  
Análise de *break-in* de chamadas (possibilidade de chamar o número do serviço e juntar a uma sala de conferência).
- **Sprint #15:**  
**Data:** 10-5-2016 a 24-5-2016  
Salas protegidas por password podem ser agora criadas.
- **Sprint #16:**  
**Data:** 24-5-2016 a 7-6-2016  
Correções de erros e melhorias na estabilidade da aplicação. Melhorias na usabilidade da interface.
- **Sprint #17:**  
**Data:** 7-6-2016 a 22-6-2016  
Finalização da aplicação e documentação

### 3.4. Riscos

#### R.1. Arquitetura não atinge a performance ideal

- **Descrição:** A arquitetura não apresenta capacidades de atingir o desempenho e a escalabilidade previstos
- **Impacto:** Médio
- **Probabilidade:** Baixa
- **Mitigação:** Adoção de uma arquitetura diferente (existe uma arquitetura de *backup*)

#### R.2. Fracas estimativas de desenvolvimento

- **Descrição:** o desenvolvimento em tecnologias nunca antes trabalhadas poderá resultar em atrasos no desenvolvimento de tarefas
- **Impacto:** Médio-Alto
- **Probabilidade:** Média
- **Mitigação:** Diálogo com utilizadores já experientes na tecnologia e uma contínua aprendizagem através da busca de informação *online*



### R.3. Dependências de bibliotecas

- **Descrição:** O projeto depende de bibliotecas externas que podem sofrer alterações ou a depreciações, podendo resultar no funcionamento incorreto da aplicação
- **Impacto:** Médio-baixo
- **Probabilidade:** Baixa
- **Mitigação:** Uso de bibliotecas que se encontrem apenas em produção, tendo cuidado nas atualizações para versões mais recentes

### R.4. WebRTC deprecado

- **Descrição:** O projeto depende da tecnologia WebRTC que poderá ser deprecada e substituída por uma tecnologia concorrente (como ORTC). Apesar do WebRTC vir a adotar os mecanismos e APIs de ORTC, consideramos a possibilidade de depreciação.
- **Impacto:** Alto
- **Probabilidade:** Muito baixa
- **Mitigação:** Atualizar os mecanismos de estabelecimento de comunicação em tempo real para tecnologias atuais. O *media-server* tem de suportar também os novos mecanismos de comunicação para as funcionalidades desenvolvidas se manterem funcionais.

### R.5. Manutenibilidade reduzida

- **Descrição:** Por vezes o software desenvolvido torna-se difícil de manter por ser complexo, usar tecnologias antigas, entre outros. A própria estruturação do código e do projeto é um fator determinante na criação de software fácil de manter.
- **Impacto:** Médio
- **Probabilidade:** Média-Baixa
- **Mitigação:** A utilização de paradigmas de programação já definidos, seguindo boas práticas para o desenvolvimento de software. A WIT possui já um guia de como criar bom software e serão usadas estas práticas

### R.6. Competição no mercado

- **Descrição:** Existem inúmeros serviços que permitem conferências, sendo que a maioria são desenvolvidos por grandes empresas já estabelecidas no contexto das comunicações em tempo real
- **Impacto:** Médio
- **Probabilidade:** Média-Baixa
- **Mitigação:** É necessário estar à frente na inovação e manter esta visão ao longo do estágio. Tendo isto em conta, é necessário manter as funcionalidades atualizadas estudando continuamente os competidores

## 4. Desenvolvimento

Este capítulo descreve o desenvolvimento dos vários componentes constituintes do projeto.

O serviço é constituído por três componentes principais:

- **Aplicação Web:** este componente é o que fornece a interface ao cliente, permitindo-o interagir com o serviço. Apesar de estar separado, este componente é servido pelo servidor de *backend*. A separação foi feita tendo em conta a sua complexidade.
- **Servidor de *backend* (ou servidor web):** possui todas as funcionalidades de gestão que oferecem à aplicação web as capacidades de interação entre múltiplos clientes remotos
- **Servidor de multimédia (*Kurento*):** responsável por redistribuir a *media* dos vários participantes, permitindo ligações de diferentes dispositivos (*web browser*, *IP phones*, etc)

Para tornar este capítulo mais breve, a arquitetura será descrita de uma forma mais superficial, sendo possível obter informação mais detalhada no documento anexado *Anexo B – Arquitetura*.

### 4.1. Arquitetura de Software

Como referido anteriormente, a solução é composta por dois componentes principais: a aplicação web e o servidor que possui a lógica de negócio.

A aplicação web é responsável por fornecer ao utilizador a interface que lhe permite interagir com o serviço, contendo também lógica que lhe permite usufruir das funcionalidades planeadas.

O servidor é responsável por manter o serviço operacional, atuando como um servidor web com os devidos controladores, modelos e vistas. Este servidor possui também um componente vital ao funcionamento do serviço que são os *websockets*. Este componente é responsável por fazer a comunicação entre servidor e cliente de forma a que a aplicação web possa realizar as devidas operações, como o estabelecimento de uma videochamada entre os participantes.

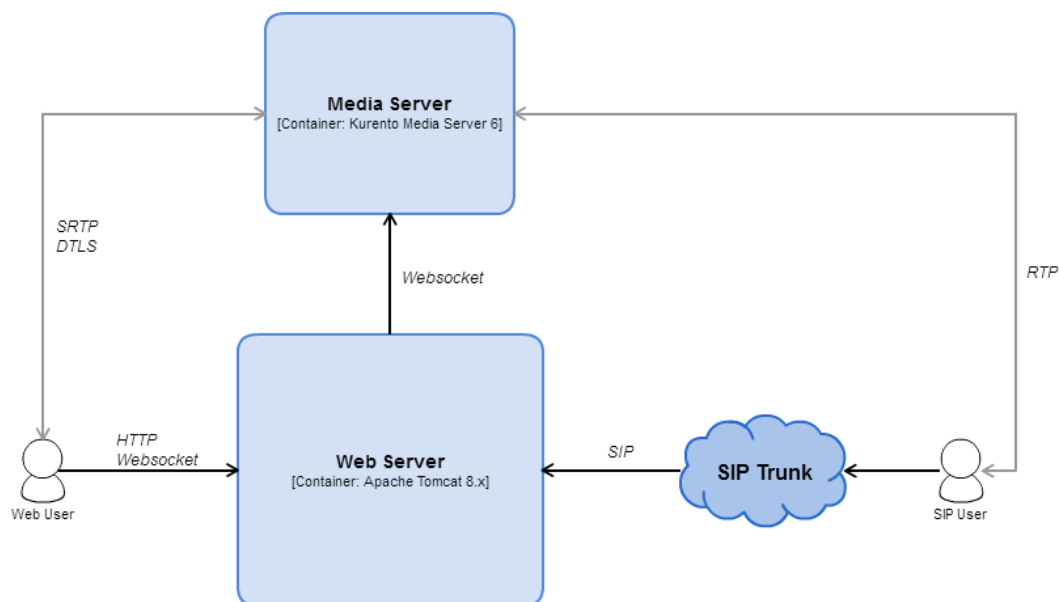
4.1.1. Servidor (*backend*)

Figura 4-1 - Diagrama da arquitetura ao nível dos contentores

A Figura 4-1 representa a arquitetura do sistema ao nível dos contentores. A aplicação web não é representada neste diagrama por ser um componente do servidor.

Entremos agora em maior detalhe na interação utilizador – aplicação web. A comunicação é feita por HTTP, acedendo ao controlador inicial que devolve a página inicial da aplicação. Nesta interface existe um pequeno formulário que permite, também por HTTP, realizar um pedido *POST* para criar uma sala de conferências. O utilizador é redirecionado para o controlador da aplicação que devolve a vista principal da aplicação *web*. Desde que esta aplicação é carregada, toda a comunicação passa a realizar-se através de *websockets*.

A aplicação web interage apenas com o componente *Websocket Handler* do servidor de *backend*. Este componente é responsável por receber e enviar mensagens de e para os clientes *web*. O *Websocket Handler* processa as mensagens em formato *JSON* que têm origem nos clientes, convertendo-as para uma classe Java com os mesmos atributos.

Os componentes *KurentoService* e *SipService* foram acrescentados na segunda metade do trabalho e permite ao serviço ter salas de conferência onde o servidor de multimédia atua como intermediário na entrega desta média e também a possibilidade de dialogar para qualquer número de telefone.

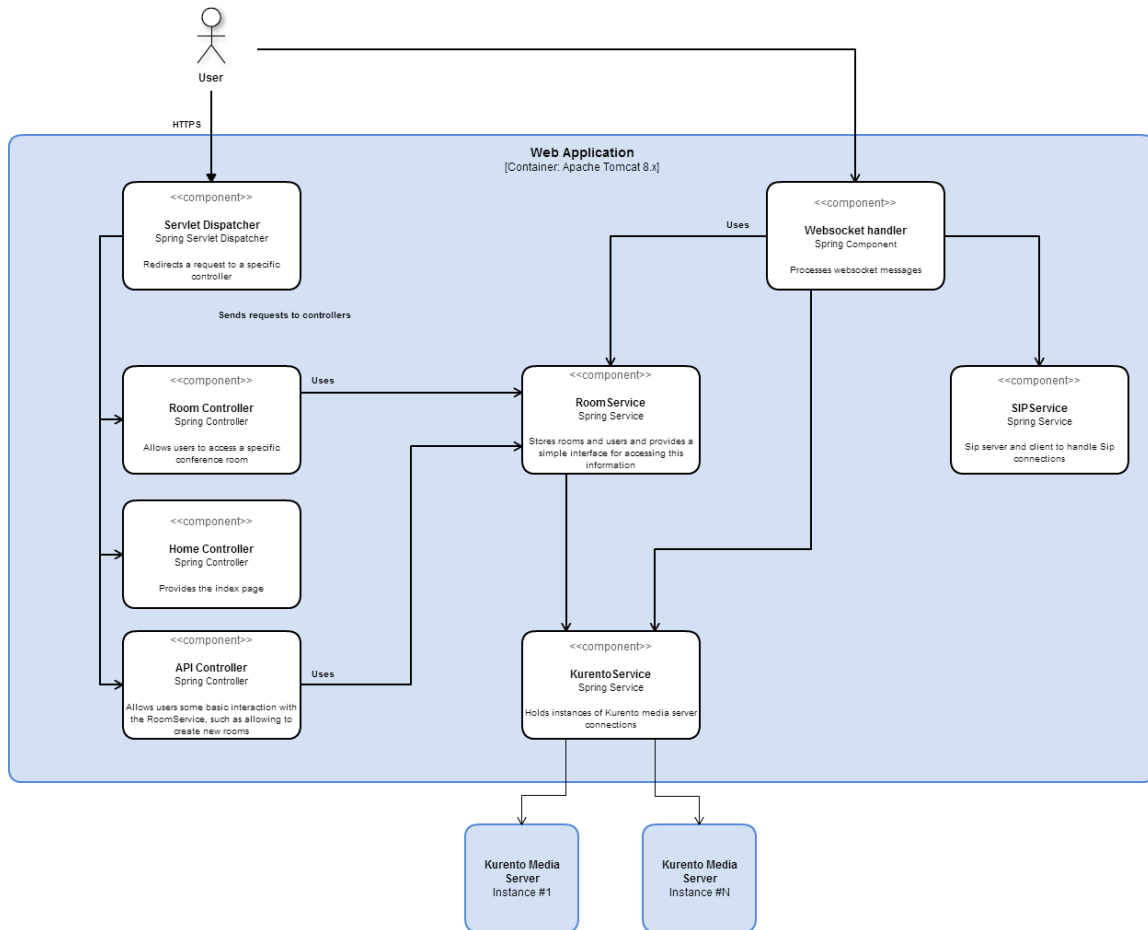


Figura 4-2 - Descrição do componente Web Server

A Figura 4-2 representa a vista de componentes do servidor de *backend*. *RoomService* é o módulo principal que gere as conferências e é utilizado por outros componentes do servidor, como o *Websocket Handler*.

O servidor de *backend* contém todos componentes necessários a um serviço de conferências P2P ou *Media Server* completo, recorrendo a tecnologias como *WebRTC*, *websockets* e *SIP*.

#### 4.1.2. Aplicação web React (*frontend*)

A interface com o utilizador é uma aplicação web que pode ser acedida através de um endereço disponibilizado pelo servidor. Esta aplicação foi desenvolvida recorrendo a uma *framework* própria para aplicações de *frontend* na *web*, *React*. Este tipo de aplicações web são executadas localmente no *browser*, ou seja, o servidor não tem de gerar as vistas e enviá-las para o cliente. O cliente (*browser*) possui toda a lógica necessária de controlo e visualização de forma a mostrar uma interface fluída e consistente.

Das inúmeras *frameworks* de *frontend* para a web foi escolhido *React* por ser uma *framework* inovadora, capaz, e com bastante suporte da comunidade. Para mais detalhes sobre este tipo de *frameworks* consultar o Anexo B – Arquitetura.

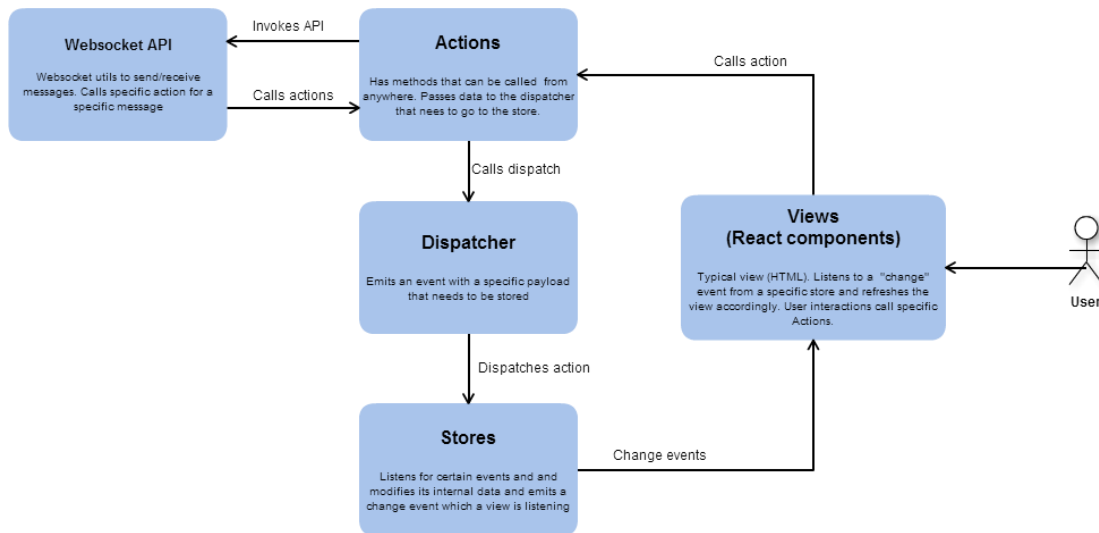


Figura 4-3 - Arquitetura genérica Flux usada em várias aplicações React

Apesar de *React* ser apenas uma *framework* para controle das *views*, é possível criar aplicações estáveis. É considerado por muitos como o “V” no “MVC”. Desta forma, o *Facebook*, criador da *framework*, definiu uma arquitetura que permite criar aplicações interativas com facilidade. Esta arquitetura foi denominada de *Flux*, encontrando-se representada na Figura 4-3. *Flux* não é uma arquitetura MVC mas uma em que existe um fluxo unidirecional de dados. Este fluxo ocorre entre os três componentes principais: *Dispatcher*, *Stores* e *Views*. Quando um utilizador interage com uma *View React*, a *view* propaga uma ação através do *Dispatcher*, que termina nas várias *Stores* que estão à escuta de determinadas ações. Estas *Stores* contêm os dados da aplicação e a lógica de negócio. Por sua vez, as várias *Views* que seguem várias *Stores* recebem a informação e atualizam as vistas.

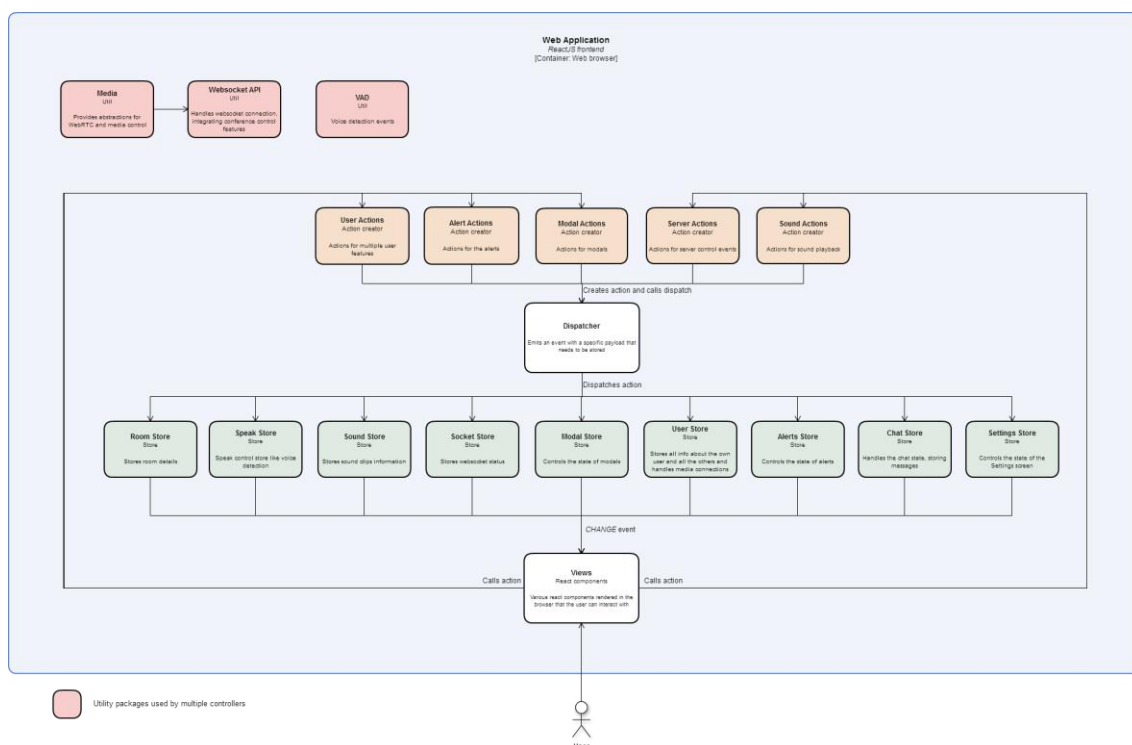


Figura 4-4 - Arquitetura da aplicação web em ReactJS (os diferentes Componentes das vistas encontram-se omitidos)

A Figura 4-4 representa a arquitetura geral da aplicação web, como se encontram conectados os diferentes componentes. Podemos facilmente confirmar como a arquitetura Flux se encontra integrada na aplicação, pelo fluxo unidirecional entre as *actions*, *dispatcher*, *stores* e *views*. De forma a não prolongar esta secção, mais detalhes sobre os componentes individuais da aplicação web podem ser consultadas no Anexo B – Arquitetura.

## 4.2. Tecnologias

Esta secção descreve como tecnologias como o WebRTC foram usadas, ou implementadas, no produto e como contribuem para o valor do mesmo.

### 4.2.1. Web sockets

*Web socket* é um protocolo que permite a comunicação entre servidor e cliente, num canal bidirecional. Esta tecnologia é maioritariamente utilizada na *web* (*browsers*) e facilita a comunicação de informação em tempo real e com uma latência muito reduzida entre os dois pontos referidos.

A comunicação é estabelecida inicialmente recorrendo ao protocolo *HTTP* e, após o pedido ser recebido pelo servidor, é feito o *upgrade* da ligação para *websocket*.

Esta tecnologia é indispensável ao correto funcionamento do serviço porque existe bastante informação que é transmitida entre cliente – servidor. A informação é passada em formato

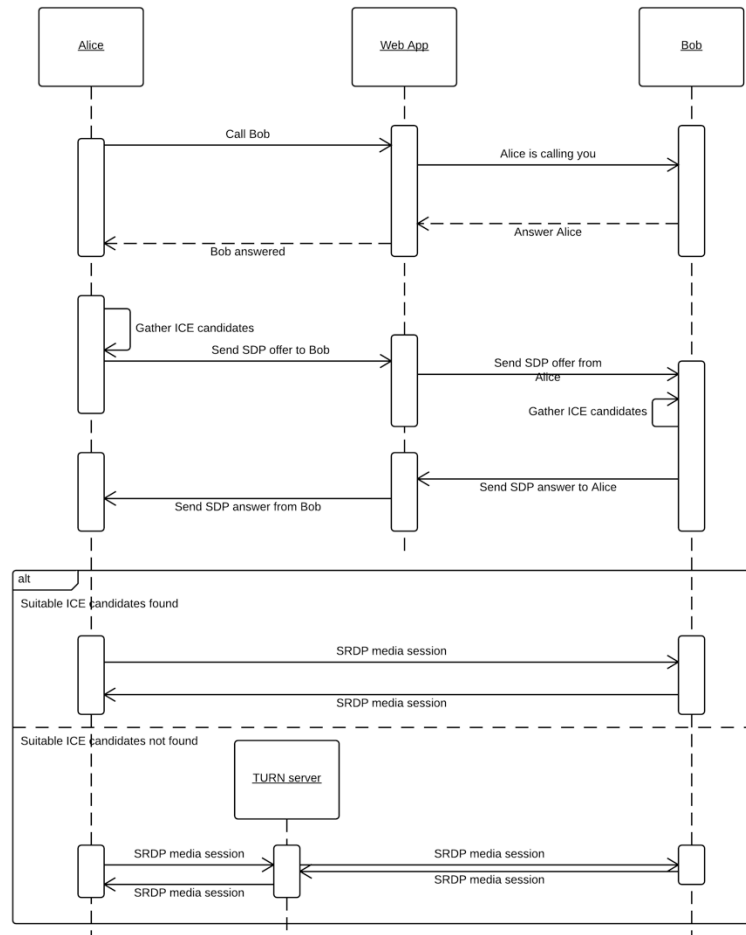
JSON que pode ser facilmente processada em qualquer linguagem. Quando um utilizador pretende entrar numa conferência, o *handshake* é feito através de *web sockets* e todas as respostas a pedidos são também transmitidas neste canal. O *handshake* serve para processar inicialmente um novo utilizador e verificar se tem permissões para entrar na sala (palavra-chave correta). Se o *handshake* for realizado com sucesso, o cliente recebe uma mensagem de sucesso juntamente com toda a informação da sala, como o tópico, os detalhes de todos os participantes conectados, entre outros.

Todas as funcionalidades que necessitam de *broadcast* (funcionalidades que requerem os restantes participantes receber uma notificação do evento) são transmitidas neste protocolo para o servidor de *backend* notificar de imediato os restantes participantes. O servidor atua como intermediário, sendo este responsável de processar a mensagem e verificar se necessita de ser reencaminhada para um ou mais utilizadores.

#### 4.2.2. WebRTC

Como referido anteriormente, WebRTC é a tecnologia vital do projeto, é a ferramenta que permite aos navegadores web como o Chrome e Firefox possuírem mecanismos de receção e transmissão de media.

Após o utilizador entrar numa sala de conferência, é-lhe automaticamente pedido permissão para aceder aos dispositivos multimédia, como o microfone e câmara. A *stream* de media que resulta deste pedido é gerida pelo navegador, mas através de *Javascript* é possível controlá-la. Para um participante poder entrar na conferência tem de estabelecer contacto com os restantes participantes, de forma a que se possa criar uma conferência P2P. O diagrama da Figura 4-5 mostra o fluxo de informação para o cliente *Alice* estabelecer uma conferência com o cliente *Bob*.


 Figura 4-5 - Diagrama de fluxo para o estabelecimento de uma conferência com *WebRTC* [13]

Como em outros mecanismos de conferência, o *WebRTC* usa o protocolo *SDP* para a descrição dos atributos da conferência. Após a *Alice* gerar o seu pacote *SDP*, este é encaminhado para o utilizador *Bob*, através de *websockets*. Por sua vez *Bob* responde com o seu pacote *SDP*, com destino à *Alice*. Os parâmetros da conferência são acordados, resta verificar se a ligação entre os dois clientes pode ser estabelecida, com recurso aos servidores *STUN* e *TURN*. Se tudo se verificar, é então estabelecida uma ligação entre os dois clientes e as *streams* de media fluem agora entre eles.

Na segunda metade do trabalho foi estudado o protocolo *SIP* (4.2.4 *SIP*) para fazer integração com clientes que se encontrem fora da plataforma online. Desta vez a ligação já não será estabelecida entre cada participante (*P2P*, *mesh network*) mas com recurso a um servidor que faça a retransmissão das *streams* (4.2.3 *Kurento Media Server*). Apesar da arquitetura da conferência mudar, o mecanismo de estabelecimento da chamada será igual, com recurso ao *WebRTC*.



#### 4.2.3. Kurento Media Server

Como referido anteriormente, a necessidade de um *Media Server* é vital para a integração de clientes fora de navegadores web. Esta necessidade existe pelo facto de haver limitação por parte dos dispositivos móveis ou de *IP Phones*. É impossível implementar a mesma lógica que existe na plataforma *online* sem recorrer a uma aplicação dedicada ao efeito.

*Kurento Media Server* [14] é um servidor multimédia que possui integração com WebRTC e disponibiliza API para Java, sendo adequado ao projeto desenvolvido. Além de suportar WebRTC oferece também suporte a clientes que possuam o protocolo RTP como método de transmissão de média, como clientes SIP. Desta forma é possível interligar diferentes tipos de clientes com diferentes tipos de codificação de média (*codecs*). A Figura 4-7 exemplifica a ligação entre dois clientes com capacidades diferentes através do servidor *Kurento*. Através da sua API, Kurento disponibiliza vários *Endpoints*, como o *WebRtcEndpoint*, *RtpEndpoint*, *RecorderEndpoint*, entre outros, permitindo criar inúmeros tipos de *pipelines*.

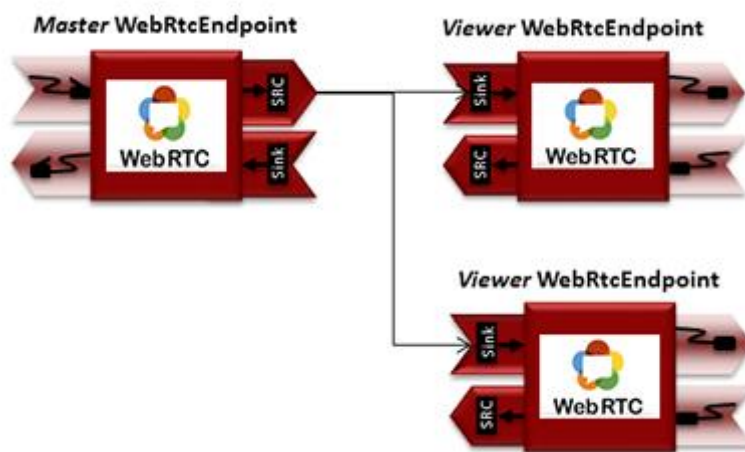


Figura 4-6 - Media Pipeline conectando 3 clientes WebRTC, sendo o Master o emissor de média [15]

A Figura 4-6 exemplifica uma *pipeline* que conecta 3 clientes WebRTC. O objetivo desta *pipeline* é encaminhar a média do cliente *Master* (áudio e/ou vídeo) para os 2 clientes *Viewers*. Internamente, o servidor automaticamente divide a *stream* para os dois clientes, sendo que o *Master* tem de enviar apenas uma vez. Aqui podemos verificar a diferença entre as topologias *Mesh* e *MCU*.

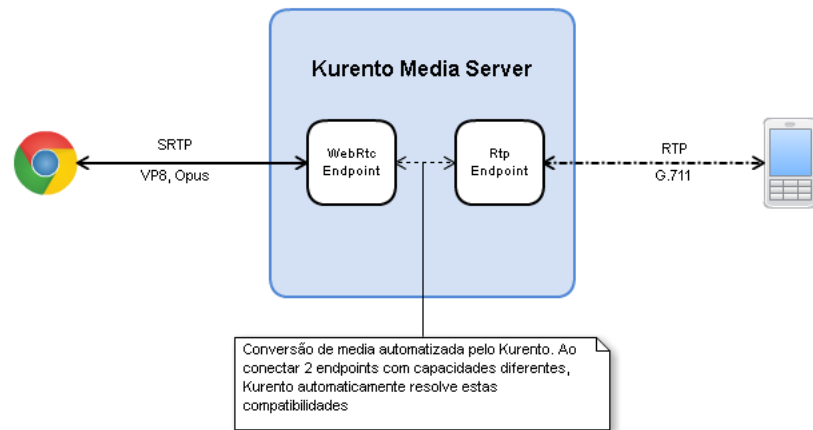


Figura 4-7 - Protocolo de comunicação em tempo real em Web Browsers e dispositivos celulares (ou IP Phones)

Como os clientes *SIP* suportam apenas uma *stream* única para transporte da mídia (*stream* única com *IN* e *OUT*), é necessário um mecanismo que junte as *streams* dos vários participantes em apenas uma. Considerando apenas áudio, este mecanismo é denominado de *mixing*. Kurento possui uma ferramenta que pode ser usada na *pipeline* que faz o *mixing* das *streams* definidas e permite conectar a *stream* já com o áudio agregado em qualquer *Endpoint*.

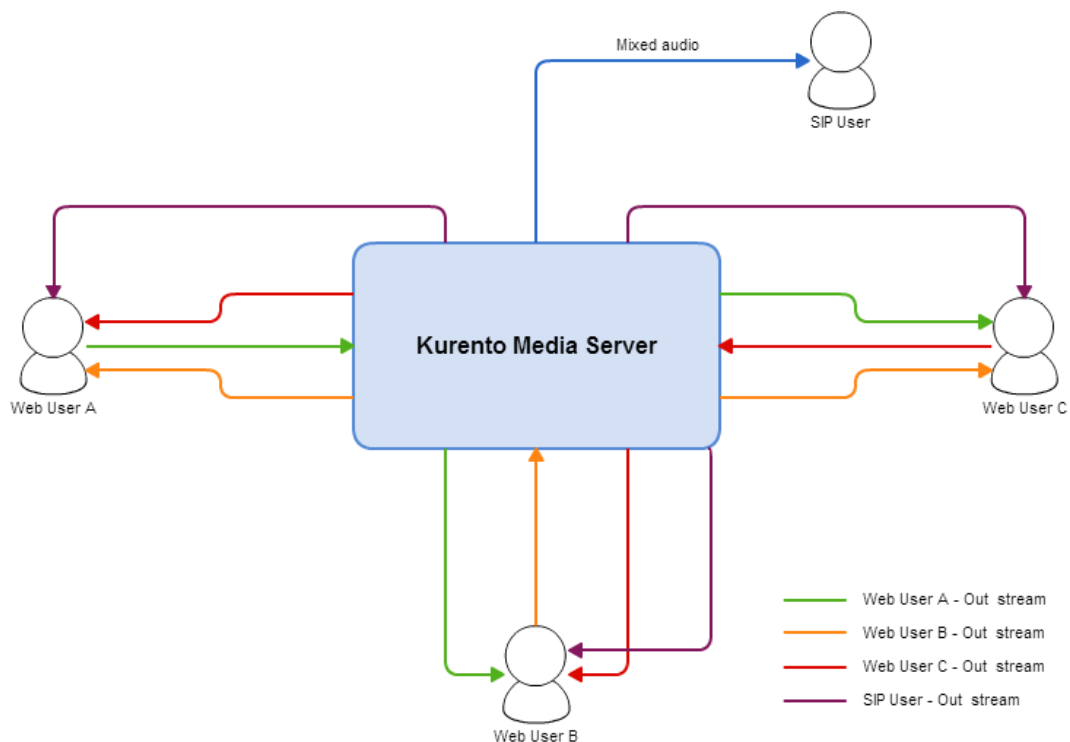


Figura 4-8 - Streams de mídia entre os vários participantes

As *streams* de cada cliente podem ser consultadas na Figura 4-8. Cada cliente web envia a sua média uma vez para o servidor e recebe as *streams* dos restantes participantes (N-1). Neste diagrama as ligações do *Kurento* encontram-se abstraídas, mas o diagrama seguinte encontra-se mais detalhado no sentido de mostrar as ligações internas criadas com recurso à API Java da biblioteca do *Kurento*.

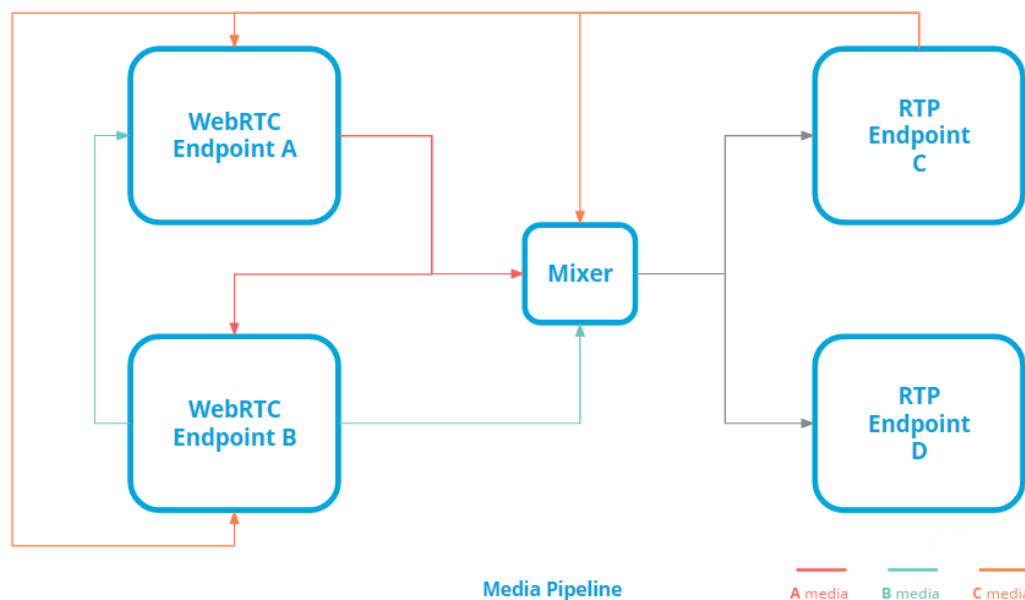


Figura 4-9 – Pipeline com as ligações internas entre os endpoints, criadas com recurso à API do Kurento (não se encontram representadas as ligações do endpoint D)

Do diagrama da Figura 4-9 está representada uma *pipeline* com 4 *endpoints*, sendo 2 *WebRtcEndpoint* (fazendo uso da tecnologia WebRTC) e 2 clientes *RtpEndpoint*. De notar que este diagrama representa as ligações internas do *Kurento* e não os próprios clientes físicos (browser ou telefone). A média de cada um dos *endpoints* tem de chegar a todos os restantes para poder existir uma conferência. O *endpoint* A está conectado ao *endpoint* B diretamente assim como ao *Mixer*. O *endpoint* B possui ligações semelhantes, estando conectado ao *endpoint* A e ao *Mixer*. O *endpoint* C está conectado a todos os *WebRtcEndpoints* e também ao *Mixer*. Como os *endpoints* RTP permitem apenas 1 *stream* em cada sentido, o *output* do *Mixer* está conectado a estes *endpoints*. Como o áudio destes participantes necessita de chegar aos restantes participantes do mesmo tipo, o *output* encontra-se também conectado ao *Mixer*. Internamente o próprio *Kurento* faz *mixing* excluindo a própria média, com o objetivo de não criar *feedback loop*<sup>4</sup>. O *Mixer* internamente junta as *streams* que recebe e faz *output* de uma *stream* que contém o áudio de todos os *endpoints*, mecanismo necessário para poder permitir clientes que suportam RTP (como SIP).

<sup>4</sup> O *endpoint* C necessita de receber o áudio de A, B e D já mixado. Mas C também se encontra conectado ao *Mixer*. Em teoria, isto iria criar *feedback* neste participante, ouvindo-se a si mesmo. O *Kurento* já resolve esta situação excluindo do *Mixer* a própria *stream* quando se encontra conectado a si próprio.

#### 4.2.3.1. Escalabilidade

Um servidor multimédia é usualmente um servidor que possui bastante carga dado que tem de processar uma elevada quantidade de dados em tempo real. Com melhor *hardware* é possível escalar a performance deste tipo de servidores, mas existirá sempre um limite para o escalonamento vertical, como por exemplo a carga na rede (*bandwidth*).

De forma a suportar um maior número de participantes conectados em simultâneo ao serviço de conferências, foi criado suporte para múltiplas instâncias de servidores *Kurento*.

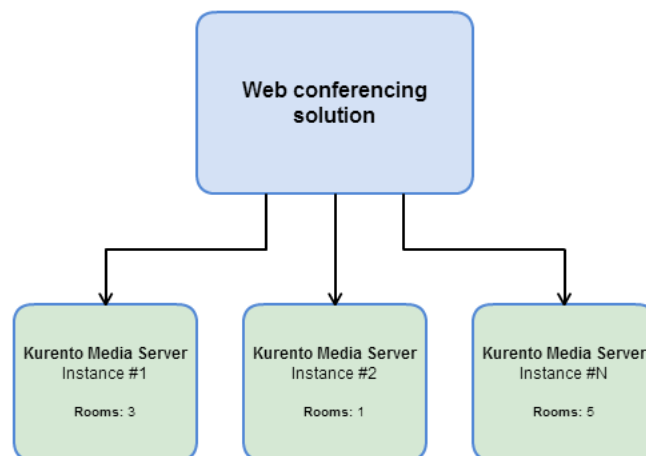


Figura 4-10 - Múltiplas instâncias de servidores *Kurento* com lotação individual

No processo de criar uma nova sala de conferência, a pedido de um utilizador, é atribuído um servidor *Kurento* específico para processar os dados multimédia desta sala, sendo escolhido o servidor com menor lotação. Desta forma a carga é distribuída pelos diversos servidores.

#### 4.2.4.SIP

SIP é um protocolo de comunicação semelhante ao HTTP (*request-response*) e é usado para a iniciar sessões de comunicação em tempo real entre participantes. É um protocolo essencial no *VoIP* pois permite o estabelecimento de chamadas áudio e vídeo via *IP*. É baseado em texto e pode ser facilmente inspecionado, assim como o HTTP. Para estabelecer uma chamada entre 2 utilizadores, o protocolo SIP faz uso do protocolo SDP para descrever a media suportada por cada participante de forma a que se possa encontrar um *codec* comum aos participantes.

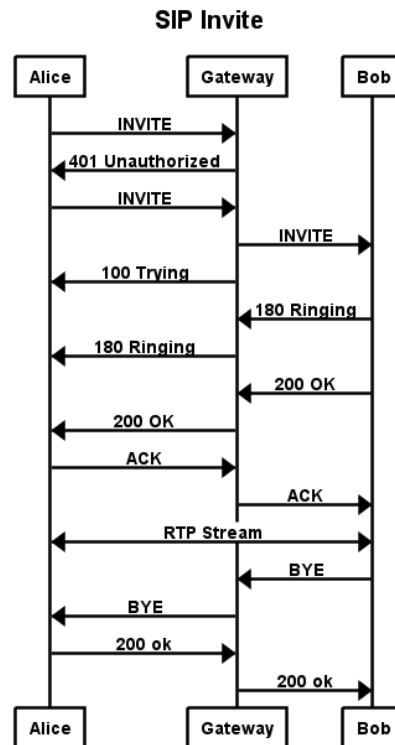


Figura 4-11 - Diagrama de sequência para um pedido de INVITE

O funcionamento do SIP é semelhante ao HTTP no sentido em que é esperada uma resposta depois de um pedido. Na Figura 4-11 encontra-se representada a sequência de pedidos e respostas que sucedem quando um cliente SIP pretende estabelecer uma chamada com outro cliente SIP. Como podemos reparar, os códigos de resposta são semelhantes ao HTTP no caso de “200 OK”. A *gateway* é um servidor que funciona como intermediário e lista telefónica, sabendo onde se encontram todos os clientes registados e é responsável por encaminhar todos os pedidos de controlo de chamada entre os participantes.

A utilização deste protocolo é fulcral para o projecto pois permite o estabelecimento de comunicações com dispositivos móveis, através de um *SIP Trunk* disponibilizado pela *IP-Brick*. A comunicação entre a solução desenvolvida e o *Trunk* é feita através de SIP sendo que foi implementado um serviço que permite facilmente iniciar uma chamada e receber o resultado desta operação, nomeadamente uma resposta do destinatário com a sua informação. *SIP Trunks* são servidores que permitem a ligação entre VoIP e redes PSTN, ou seja, é possível fazer um pedido de início de chamada a um endereço SIP que pode estar ligado via rede celular. O *Trunk* interage com a rede PSTN e realiza todo o tipo de lógica necessária à intercomunicação entre os dois serviços. Esta interação está representada na Figura 4-12.

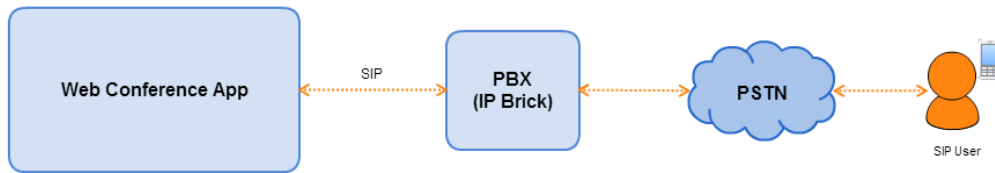


Figura 4-12 - Diagrama da comunicação entre a solução desenvolvida e participante SIP

Do ponto de vista do projecto, foi estudado o protocolo SIP para se poder criar um serviço interno que permita estabelecer uma chamada para um destinatário SIP, independentemente da rede em que se encontre. A WIT Software já possui uma solução que permite facilmente acrescentar funcionalidades de comunicação em tempo real, como a que se pretende desenvolver, através de SDK's para Javascript, mas por ser necessário um maior controlo sobre a comunicação foi então implementado um serviço semelhante que implementa o protocolo SIP e faz a comunicação com o *IP-Brick*.

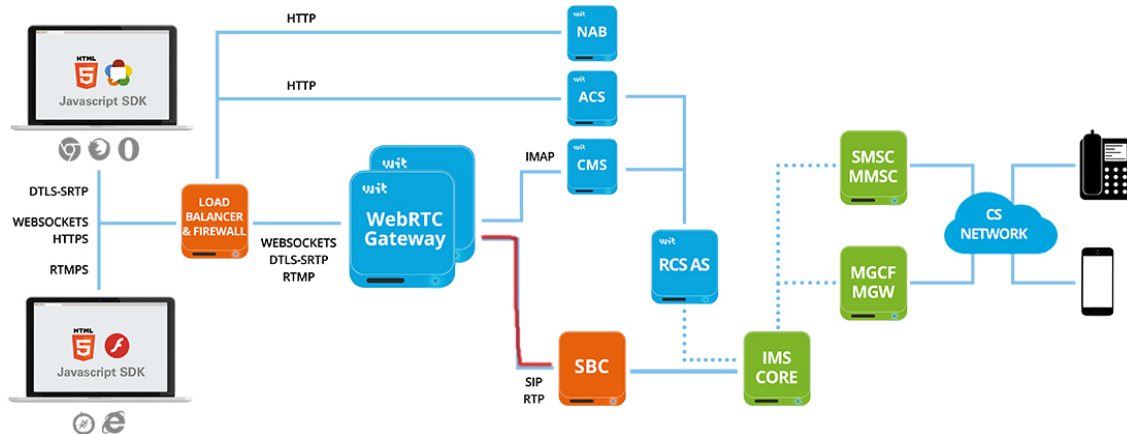


Figura 4-13 - WIT WebRTC deployment architecture [16]

A Figura 4-13 representa a arquitetura de WebRTC disponibilizada pela WIT. Como podemos reparar, existe um SDK que possuiu toda a lógica de comunicação com a WebRTC Gateway sendo que esta é responsável por fazer a comunicação com outros serviços, permitindo chamadas (entre muitas outras funcionalidades) com dispositivos na rede móvel. Como referido anteriormente, apesar da WebRTC Gateway já possuir estas funcionalidades de chamadas, foi desenvolvido um pequeno módulo similar ao que a Gateway já possui, sendo este módulo responsável por realizar a comunicação por SIP (linha a vermelho no diagrama).

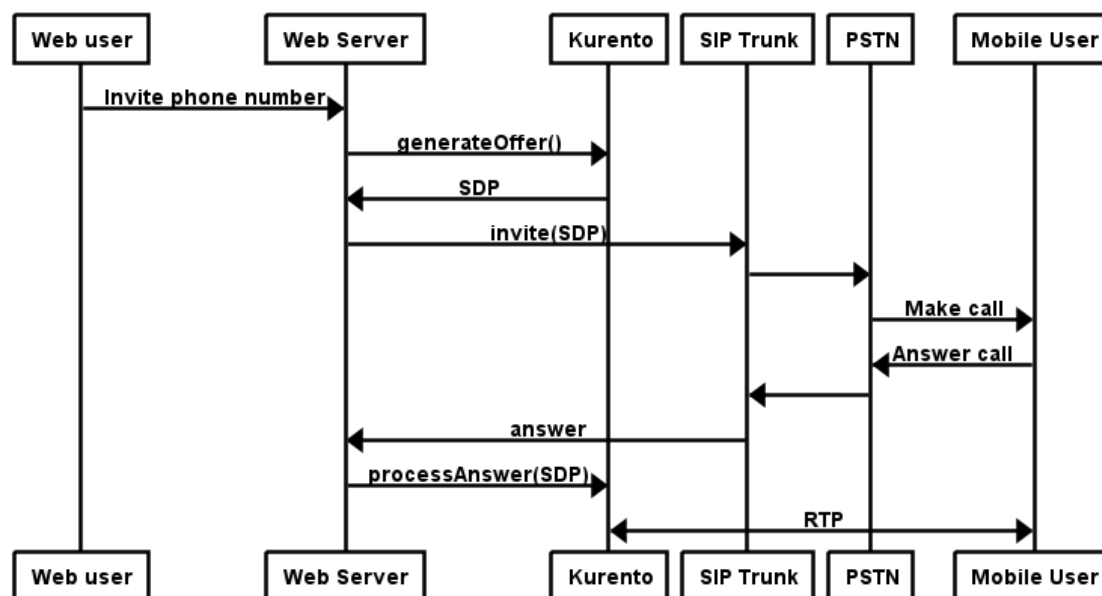


Figura 4-14 - Diagrama de sequência de convite de um número telefónico

A Figura 4-14 exemplifica a sequência de eventos da funcionalidade de convidar de um participante móvel para uma conferência. O participante na web faz o pedido de chamada e o servidor, recorrendo ao componente *SIP*, envia um *INVITE* para o número pretendido. O servidor multimédia *Kurento* é utilizado para gerar o *SDP* de oferta, que é enviado juntamente com o *INVITE*, e processar a resposta do cliente móvel que contém também o seu *SDP*. Assim que ambos os intervenientes (*Kurento* e cliente móvel) possuem o *SDP* de ambos, é possível estabelecer a ligação RTP entre eles.

### 4.3. Dificuldades

O projeto revelou-se desafiante e determinadas funcionalidades provaram-se mais difíceis de ultrapassar.

O início do trabalho foi o mais desafiante pois a aplicação a desenvolver parecia ser de grande complexidade e a definição da arquitetura tornou-se mais difícil que o esperado mas com a ajuda dos orientadores para me indicar o caminho e eventuais ferramentas e *frameworks* o processo tornou-se mais simplificado.

O desenvolvimento numa nova *framework* para a web, *React*, foi necessário compreender o seu mecanismo de funcionamento, como e quando são realizadas as atualizações nos componentes. *React* processa os vários componentes e realiza atualizações na *DOM* apenas quando esta difere. A estruturação da aplicação web e perceber como os vários componentes da aplicação interagem entre si foi um processo com alguma dificuldade mas ultrapassado com sucesso. A integração da interface também foi um processo contínuo, sendo otimizada ao longo do estágio.

Relativamente a conferências, a sincronização da troca de ofertas entre clientes foi um processo exaustivo pois foi necessário trabalho no cliente como no servidor, possuindo os dois componentes lógica para lidar com esta troca e processamento.

A integração com o servidor multimédia *Kurento* foi menos complexa do que o esperado em grande parte por já possuir um sistema de conferências funcional onde as bases do serviço já se encontravam em funcionamento. Agora além de suportar conferências P2P onde os clientes trocam as ofertas entre si, o servidor integra a API do *Kurento* e este processa as ofertas e devolve a resposta ao cliente.

O último ponto foi a integração com SIP e, apesar do protocolo ser simplificado, foi necessário inspecionar os pacotes trocados entre um *IP Phone* e o servidor multimédia para verificar os pacotes *SDP*, confirmando que os *codecs* necessários estão disponíveis e os *IP's* dos *proxies SIP* corretos.



## 5. Resultados

Este capítulo resume o trabalho realizado ao longo do estágio. Será feito um pequeno resumo juntamente com algumas imagens que demonstram o produto final desenvolvido.

Referir ao Anexo A – Abordagem para uma descrição completa das *User Stories* implementadas durante os dois semestres de desenvolvimento.

### 5.1. Conferência entre clientes web

No primeiro semestre foi dado início ao serviço de conferências e o resultado final foi uma aplicação capaz de conferenciar com múltiplos participantes numa topologia *P2P*. Para permitir este tipo de conferências foi necessário o servidor de *signaling* com a componente de *websockets* para permitir a troca de informação de *WebRTC* em tempo real assim como uma interface web inicial permitindo visualizar o vídeo e escutar a voz dos participantes em conferência.

O resultado deste semestre foi uma aplicação web onde era possível conferenciar entre vários utilizadores, sendo esta sala de conferência acedida através de um *URL*. Como é possível verificar na Figura 5-1, a sala de conferência de tópico “andre’s room” possui 4 participantes (incluindo o próprio). O participante que se encontra num fundo azul está-se a conectar à conferência e ainda não passou o ecrã de configuração onde é possível escolher a câmara, microfone e definir um nome de utilizador.

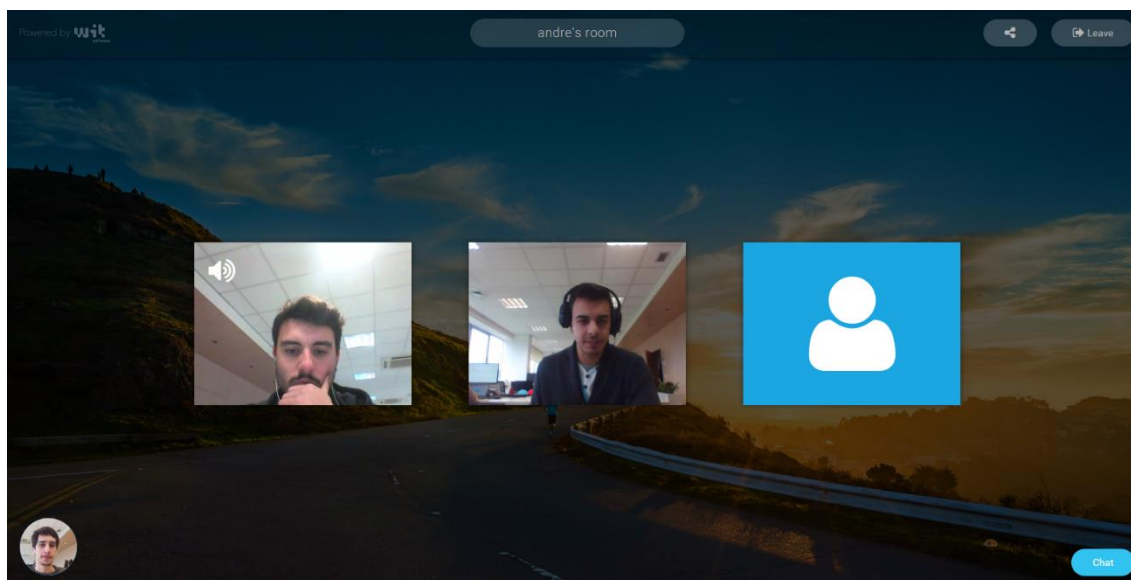


Figura 5-1 - Sala de conferência resultante da implementação do 1º semestre

Neste semestre foi também implementada a detecção de som e é possível verificar que o primeiro participante possui um símbolo de uma coluna indicando que se encontra em atividade.

## 5.2. Conferência entre clientes *web* e *VoIP*

No segundo semestre foram trabalhadas as funcionalidades que acrescentam mais valor ao produto, nomeadamente uma maior escalabilidade de participantes e a possibilidade de integrar participantes fora da *web*.

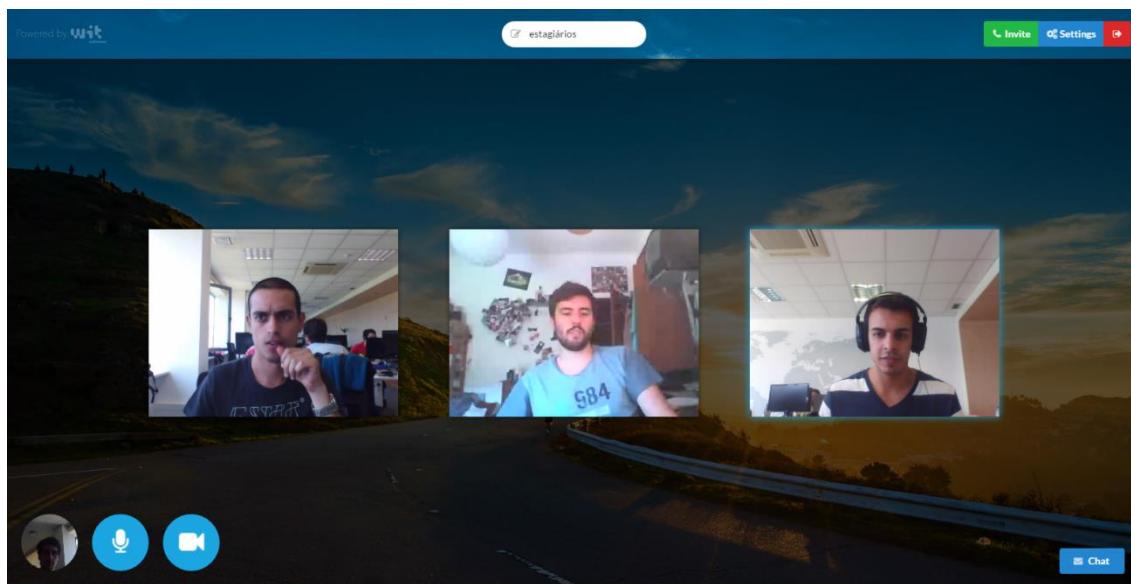


Figura 5-2 - Sala de conferência resultante da implementação do 2º semestre

A Figura 5-2 mostra o resultado final do produto, numa conferência com 4 participantes. O ícone de notificação de mensagens no *chat* e barra de ferramentas são visíveis. O participante à direita possui uma cor azul no rebordo indicando que se encontra em atividade. O símbolo de uma coluna na Figura 5-1 foi substituído por esta cor, simplificando a interface. Neste semestre, além da integração SIP, foi adicionado o ecrã de convite e de definições (*Invite* e *Settings*). O ecrã de convite, representado pela Figura 5-3, permite dialogar um número telefónico que será adicionado à conferência. Como a sala de conferência necessitava de ser transitada para o servidor multimédia, é apresentado um alerta que a sala será atualizada.

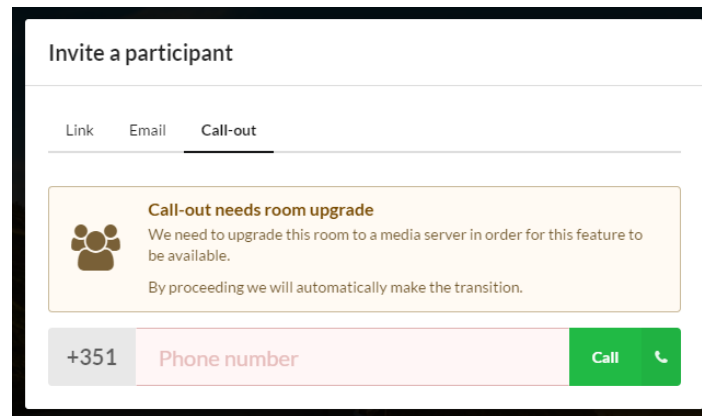


Figura 5-3 - Ecrã de convite de números telefónicos

O ecrã de configurações da Figura 5-4 permite acesso a informação mais técnica da sala de conferência, como atualização manual da topologia, ver os participantes e as suas capacidades, alteração dos dispositivos de *media* (câmara e microfone) e configuração do microfone e limiar de deteção.

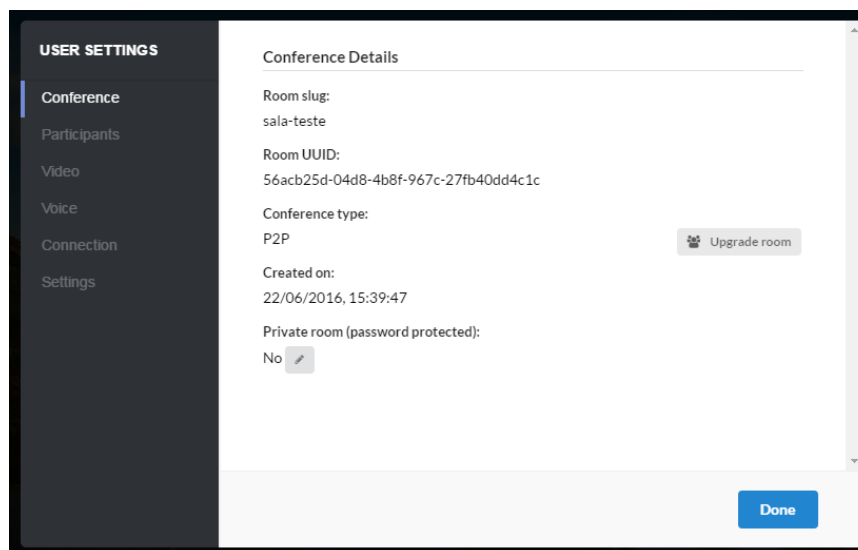


Figura 5-4 - Ecrã de informação/ configuração da conferência

### 5.3. Funcionalidades não implementadas

Do conjunto de funcionalidades planeadas no início do semestre, algumas não foram implementadas por estarem fora do âmbito do projeto ou por dificuldades na implementação. De um modo geral, estas funcionalidades que não foram implementadas não são críticas e, apesar de criarem valor no produto, o serviço como existe atualmente está inteiramente funcional do ponto de vista geral das conferências.

Nesta seção encontram-se listadas as funcionalidades não implementadas, juntamente com a razão para não se incluírem no produto final.

- ***Break-in* de chamadas**

Esta funcionalidade permitiria qualquer telefone poder contactar o número atribuído ao serviço, através do teclado digitar o número da sala e entrar assim numa conferência. O problema surgiu na deteção das teclas que o utilizador móvel clica, não sendo possível fazer esta deteção. Esta funcionalidade pode ser implementada, mas não o foi em tempo útil do estágio. Seria necessário adicionar suporte a *DTMF* no servidor multimédia *Kurento* ou na *gateway* e esta enviaria mensagens *SIP INFO* com as teclas premidas.

Como não é possível saber que sala o participante pretende entrar, foi especificado uma sala por defeito para efeitos de teste, ou seja, esta funcionalidade está implementada, sendo impossível para utilizador a partir do telemóvel escolher em que sala entrar.

- **Participante “apresentador”**

A possibilidade de definir um participante principal, tanto por um moderador como por um participante normal, não foi finalizada. O objetivo seria permitir a um moderador definir um participante como participante principal e todos os restantes participantes recebiam essa informação e aumentariam o tamanho do vídeo do participante respetivo. A outra forma seria permitir a um participante que não seja moderador de escolher, localmente, qual seria o participante que desejaria observar com maior resolução. A funcionalidade não foi implementada por terem surgido alguns problemas quanto à disposição dos participantes na conferência e não foi possível em tempo útil permitir que esta disposição fosse mutável.

- **Desligar vídeo de um participante**

Um participante poderia escolher se pretendia receber o vídeo de qualquer participante à escolha com o objetivo de reduzir a largura de banda em participantes que o desejassem. A funcionalidade não foi implementada pois para cancelar (com o objetivo de não a receber, ao contrário de *mute* que simplesmente ignora a *stream* localmente, depois de a receber) uma *stream* de áudio ou vídeo é necessário renegociar a ligação com o participante (ou servidor multimédia). Para realizar esta renegociação é necessária lógica adicional, tanto no cliente como no servidor, e não fui capaz de solucionar os erros que surgiram em tempo útil.

## 6. Qualidade de Software

O processo de validação do produto desenvolvido é fundamental em qualquer projeto de *software*. É esperado que a aplicação seja entregue sem erros, e com a melhor interface possível de formar a criar uma boa experiência de uso.

Foram desenhados e executados vários testes de forma a validar as funcionalidades implementadas. É recomendada a leitura do Anexo C – Qualidade de Software pois contém uma descrição extensiva dos testes executados.

### 6.1. Testes Funcionais

Esta secção do documento descreve os testes de aceitação para validar e garantir qualidade no *software*, através de testes *black-box*.

#### 6.1.1. Testes de Aceitação

O seguinte conjunto de testes contém todos os testes de aceitação criados e executados de forma a garantir o correto funcionamento dos requisitos funcionais. Desta forma, todos os testes não possuem imenso detalhe dado que são para ser executados por uma equipa dedicada a garantir a qualidade de *software*. O testador deve saber qual é a resposta do sistema a determinada ação, ignorando qualquer lógica interna do processo.

Apesar dos testes de aceitação serem idealmente realizados por uma equipa dedicada, todos os testes descritos foram realizados por mim.

Foram definidos uma totalidade de 19 testes e executados de forma a garantir o correto funcionamento das funcionalidades implementados no produto. Estes testes pretendem avaliar o funcionamento da interface web.

O processo de testar o *software* foi contínuo tendo decorrido ao longo da implementação. Os mesmos testes podem ter sido executados múltiplas vezes até que todas as falhas estivessem corrigidas e a funcionalidade se comportasse como esperado. A Tabela 2 descreve os resultados obtidos. Para mais detalhes sobre os testes de aceitação realizados, consultar o Anexo C – Qualidade de Software.

Tabela 2 - Resultados dos testes de aceitação

Categoria	Primeiro Resultado		Último Resultado	
	Falhou	Passou	Falhou	Passou
Chat	1	1	0	2
Ecrã de configuração	2	4	0	6
Videoconferência	2	3	0	5
SIP	1	1	1	1
Deteção de voz	1	1	0	2
Gestão de salas	0	2	0	2

## 6.2. Testes de Qualidade

Os testes funcionais realizados anteriormente avaliam o comportamento da aplicação através na análise das respostas do sistema perante os pedidos. Nesta secção avaliaremos a usabilidade da aplicação, estando mais focada nos requisitos não-funcionais.

### 6.2.1. Testes de Usabilidade

Este capítulo contém uma breve análise aos testes de usabilidade feitos à interface web.

Testes de usabilidade são uma técnica focada em avaliar um produto. São testes feitos por utilizadores reais num ambiente controlado e fornecem informação extremamente relevante sobre o comportamento do produto e como outros utilizadores se irão comportar perante uma determinada interface.

Em primeiro lugar os testes foram realizados por um conjunto de cinco indivíduos e os resultados obtidos da experiência serão analisados e comentados de seguida. Numa segunda parte será feita uma um exercício de introspecção sobre usabilidade usando as dez heurísticas fundamentais de usabilidade, denominadas de as 10 heurísticas de Nielsen. Este último processo ajudou-nos a detetar problemas de usabilidade na interface, fazendo uma comparação com estes 10 princípios.

De seguida, as observações adquiridas pela execução dos testes serão brevemente descritas. Para maior detalhe sobre o processo e testes realizados, consultar o Anexo C – Qualidade de Software.

Os testes de usabilidade revelaram que, de uma forma geral, a interface é fácil de utilizar dado que aplica conceitos já familiares para a maioria dos utilizadores. Como qualquer interface, existe sempre espaço para melhoramentos. A barra de ações do participante (barra no canto inferior esquerdo) possui também a imagem do próprio utilizador. Este mecanismo causou algum tipo de confusão nos participantes por não saberem que se encontravam na conferência e pelo vídeo ser pequeno e se encontrar dentro de um círculo, cortando grande

parte da imagem. As principais funcionalidades da aplicação encontram-se bastante acessíveis a partir do ecrã de conferência, havendo um diálogo de definições que contém algumas configurações mais técnicas para utilizadores mais avançados

Através da análise dos princípios de Nielsen concluiu-se que nem em todas as situações a aplicação fornece *feedback* de uma ação, podendo induzir o utilizador em erro. A simplicidade da interface pretende ajudar o utilizador na sua interação com a aplicação, no entanto em ecrãs mais complexos não existe qualquer guia, ou ajuda. A prevenção de erros é também um ponto importante, evitando que o utilizador introduza informação ou que realize uma ação que possa resultar em erro. Desta forma alguns campos encontram-se protegidos, no entanto não existe qualquer tipo de ecrã de confirmação caso o utilizador pretenda retroceder.

Com o objetivo de mitigar alguns dos problemas encontrados, a renovação da interface para uma interface mais compacta e com informação sobre as possíveis ações disponíveis pode melhorar a experiência de uso. A barra superior pode conter mais informação sobre a conferência, a barra de ferramentas no canto inferior esquerdo pode ser redesenhada com o objetivo de ser mais compacta e estar sempre visível, alterando o local onde a imagem do utilizador é visualizada. No *chat* de texto poderia haver a distinção entre utilizadores simplesmente adicionando uma cor específica por utilizador ou utilizando uma imagem *avatar*. O ecrã de configurações poderia conter alguma informação de como o usar, especialmente na aba da Voz, explicando ao utilizador o processo de deteção de som.

### 6.2.2. Testes de Performance

Uma componente importante do software é o seu desempenho nas plataformas alvo. Esta seção pretende descrever como se comportam os vários componentes do software desenvolvido, aplicação *web*, servidor e *media server*.

Para isso foi monitorizado o uso de *cpu* e largura de banda tanto do cliente web como do servidor multimédia, para um conjunto diferente de configurações, fazendo variar o número de clientes assim como a topologia da conferência (P2P ou MCU).

#### 6.2.2.1. Configuração das Máquinas Virtuais

##### Aplicação Web

OS: Ubuntu 14.04.4 LTS  
CPU: 2 Core Intel(R) Xeon(R) X3363 @ 2.83GHz  
RAM: 16 GB

##### Kurento Media Server

OS: Ubuntu 14.04.4 LTS  
CPU: 2 Core Intel(R) Xeon(R) X3363 @ 2.83GHz

RAM: 16 GB

#### Cliente

OS: Microsoft Windows 7

CPU: Intel Core i5-3437U @ 1.9 GHz

RAM: 8 GB

Browser: Google Chrome 51.0.2704.103 m

#### 6.2.2.2. Aplicação web - WebRTC

Lidar com mídia é uma atividade que faz grande uso do processador pois é necessário codificar o áudio e o vídeo em tempo real. Recomenda-se também que a largura de banda disponível seja o maior possível. WebRTC é um protocolo adaptativo no sentido em que se auto ajusta consoante os limites de rede e de poder de processamento (*Adaptive bitrate streaming*).

Quando é estabelecida uma ligação WebRTC, este tenta manter um *bitrate* de vídeo e áudio consoante as capacidades locais. Nas experimentações realizadas, com 1 Gb de rede disponível, foi mantido um *bitrate* de 500 *kbps* para o vídeo. Este valor ajusta-se instantaneamente quando necessário, permitindo uma visualização fluida. Como o serviço de conferências desenvolvido permite duas topologias, podemos fazer alguns cálculos para perceber a largura de banda utilizada.

##### Topologia P2P

- 1 Ligação de envio e receção por participante
- N Participantes na conferência (inclusive)
- $N * 500 \text{ kbps}$  envio
- $N * 500 \text{ kbps}$  receção

Quando a conferência possui 4 participantes e assumindo uma taxa de 500 *kbps* fixos por ligação, um cliente necessita de  $4 \times (500 + 500) = 4\text{Mb/s}$ .

##### Topologia MCU (Media server)

- 1 Ligação de envio
- N Ligações de receção
- N Participantes na conferência (inclusive)
- 500 *kbps* envio
- $N * 500 \text{ kbps}$  receção

Quando a conferência possui 4 participantes e assumindo uma taxa de 500 *kbps* fixos por ligação, um cliente necessita de  $500 + 4 \times 500 = 2.5\text{Mb/s}$ , em média, de largura de banda.

Na prática os 500 *kbps* não são estáticos e quanto mais participantes existirem na conferência menor será o *bitrate* do vídeo e consequentemente a taxa de receção por participante. O fator



que maior contribui para uma menor performance é a decodificação do vídeo que requer boa capacidade de processamento. Por cada ligação WebRTC é necessário fazer *decode* e este processo é exaustivo pois necessita de ocorrer em tempo real.

Através de experimentação prática, foram registados os valores de uso do *CPU* pelo navegador *Chrome* em função do número de participantes de uma conferência. A Figura 6-1 mostra os resultados obtidos.

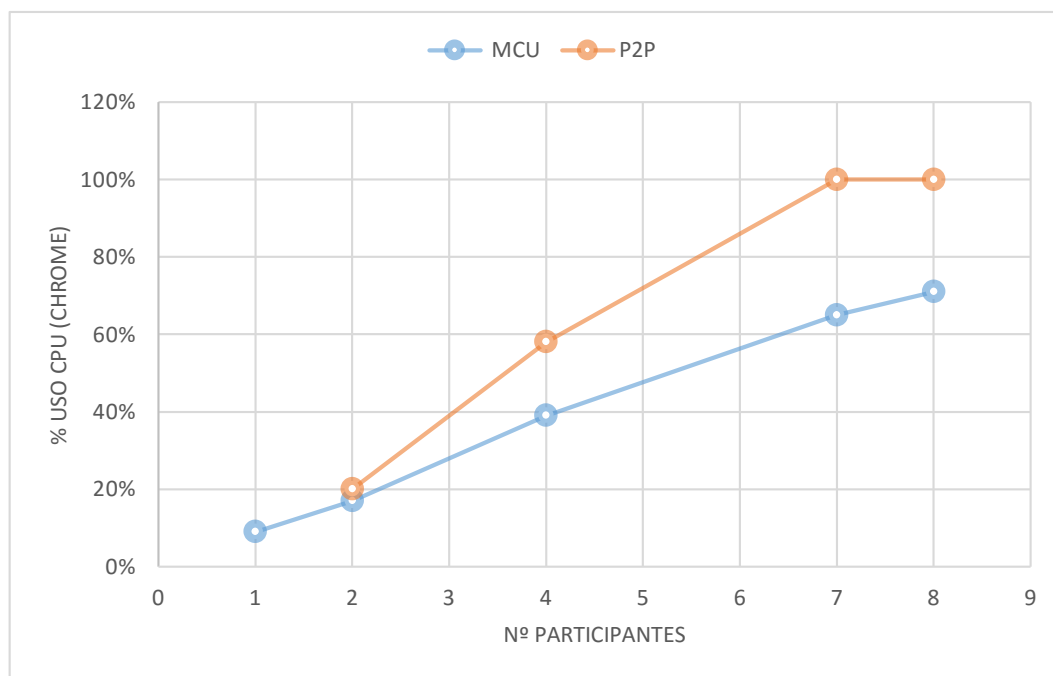


Figura 6-1 - Percentagem de uso de CPU do Chrome em função do número de participantes e topologia

É possível confirmar que o uso de *cpu* é bastante intenso e quanto mais participantes estiverem em simultâneo, maior a atividade no *cpu*. Foram testadas as duas topologias e é possível confirmar que por usar um servidor multimédia a escalabilidade da conferência no cliente é superior.

#### 6.2.2.3. Kurento Media Server

De forma a verificar a escalabilidade vertical do servidor multimédia foi monitorizada a sua carga e largura de banda em diferentes condições.

A Figura 6-2 representa a taxa de receção e envio, em *Mbps*, quando um conjunto de participantes se encontra conectado em conferência.

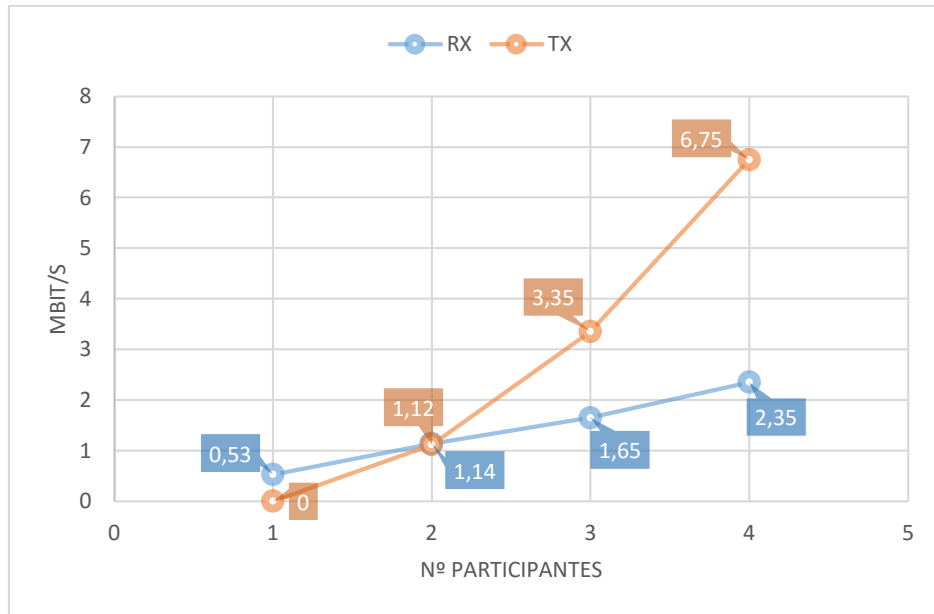


Figura 6-2 - Largura de banda utilizada pelo servidor multimídia (em Mbps) com diferentes participantes conectados

Quando existe um participante ativo, o servidor já se encontra a receber a sua *media* pelo que a taxa de recepção é igual à taxa de envio do cliente web, sensivelmente 530 Kbps. Quando mais clientes se conectam, a taxa de recepção do servidor aumenta linearmente, sendo que por cada participante a taxa de recepção aumenta em sensivelmente 540 Kbps. A taxa de envio do servidor já não aumenta linearmente em grande parte por haver uma boa quantidade de largura de banda para envio disponível assim como uma maior largura de banda de recepção nos clientes. De notar que estas taxas de transferência estão sujeitas a inúmeras variáveis que o protocolo WebRTC tem em conta por utilizar um protocolo de *streaming* denominado de *Adaptive Bitrate Streaming*, que tem em conta a largura de banda disponível e ajusta a qualidade da *stream*.

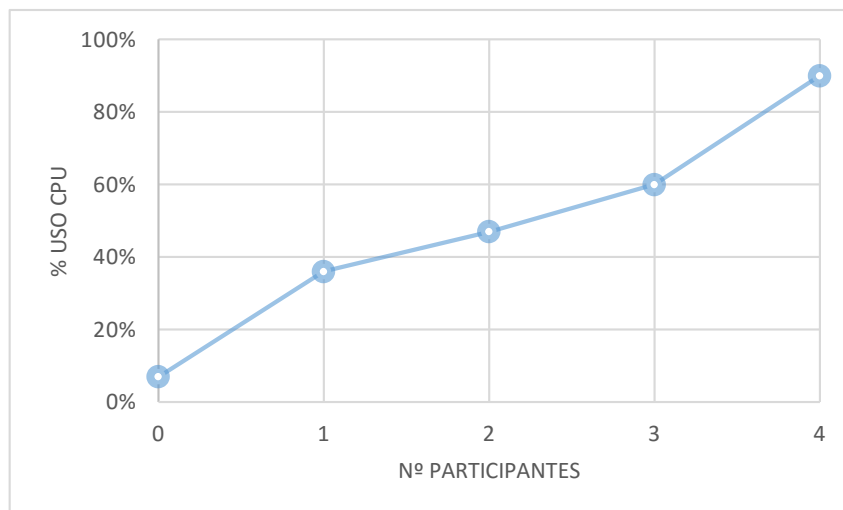


Figura 6-3 – Percentagem de uso do cpu pelo processo Kurento em função do nº de participantes

Na figura anterior encontra-se representado a percentagem de uso do *cpu* pelo processo *kaurento-media-server* em função do número de participantes. Os valores apresentados foram anotados depois de estabilizarem. Podemos confirmar que o processamento da *media* é exaustivo e com 4 participantes a taxa de uso já se encontra nos 90%. De notar que o processador possui dois núcleos, podendo a taxa de uso subir acima dos 100%.

## 7. Conclusão

Este capítulo conclui o relatório com uma visão geral do trabalho realizado durante o estágio. Encontra-se dividido em três secções diferentes para um sumário mais sucinto. A primeira secção apresenta uma visão geral do estágio de uma perspectiva dos objetivos. De seguida, uma pequena análise sobre o trabalho futuro incluindo algumas sugestões de melhorias nas funcionalidades presentes e outras funcionalidades interessantes a explorar. A última secção apresenta reflexões finais do trabalho realizado assim com da experiência de estágio.

### 7.1. Visão geral

De uma perspectiva dos objetivos a atingir podemos dizer que o projeto desenvolvido atingiu as metas propostas. O objetivo era criar um serviço de conferências que permitisse gerir chamadas em conferência na web e a integração com uma *gateway* de voz e vídeo de forma a permitir a comunicação com participantes fora da *web*, como por exemplo telemóveis.

Estes objetivos foram cumpridos e foi criado um serviço de conferências que permite criar uma sala de conferência para múltiplos participantes com bastante facilidade. Partilhando o endereço específico de uma sala é possível convidar qualquer participante que possua um computador e um *web browser*. Através de uma interface desenvolvida em *React*, é possível interagir nesta conferência de uma forma usual, estando ao dispor do utilizador várias funcionalidades esperadas neste tipo de ferramentas, como o cancelamento de áudio e vídeo, janela de conversação, entre outros. Do ponto de vista técnico, para a gestão de uma conferência com participantes web é apenas necessário um servidor que atue como intermediário na troca de informação entre cada cliente, nomeadamente informação do *WebRTC* necessária ao estabelecimento de uma chamada.

A segunda fase deste projeto também foi cumprida com sucesso que seria a integração de participantes fora do mundo *web*. Estes participantes não possuem a mesma flexibilidade que clientes web possuem, sem a criação de uma aplicação móvel específica. Desta forma foi necessário integrar chamadas celulares com o sistema já desenvolvido. Para isso foi feito uso de um servidor multimédia e de uma *gateway* que permite, através de *SIP*, gerir chamadas telefónicas.

Algumas funcionalidades de menor importância acabaram por não ser incluídas no desenvolvimento por estarem fora do âmbito do estágio. Quando foi elaborado o *Product Backlog* foram definidas um conjunto de funcionalidades e à medida que o desenvolvimento ocorreu a visão final do produto tornou-se mais clara e incluir estas funcionalidades do produto final implicaria maior tempo de desenvolvimento para finalizar todas as funcionalidades. Desta forma, é importante referir que o conjunto de funcionalidades de maior importância, as que contribuem para o valor do produto, foram feitas com sucesso.

## 7.2. Trabalho futuro

O projeto revelou-se interessante e, apesar de ter sido bem sucedido em desenvolver um produto funcional que possa fazer a diferença dentro da empresa, existem ainda oportunidades para melhorar o produto existente.

A interface foi esboçada por um *designer* sendo que eu fiz a minha implementação desse esboço. De forma a tornar o produto mais profissional, uma sugestão seria o esboço de todos os ecrãs por parte de um designer de forma a obter uma interface mais consistente e profissional.

Do ponto de vista das conferências foram encontrados, em sessão de *brainstorming*, vários problemas ou mecanismos que não funcionam tão bem em serviços de conferências gerais, como por exemplo atrasos dos participantes, agendamento de conferências, lidando com diferentes participantes em *timezones* diferentes. Quando, no caso de reuniões, os participantes entram é difícil perceber o contexto desta conferência, quem é quem, quem se encontra a falar e o que foi discutido anteriormente. Problemas de áudio e vídeo como falar em simultâneo ou fraca qualidade no som pode resultar também em más experiências de utilização. A possibilidade de ter participantes em modo “ouvinte” pode ser uma excelente funcionalidade. Estes participantes ouvem apenas, sendo apenas recetores e, de forma a permitir um maior número de participantes, estes podem receber áudio com menor qualidade e com atraso, caso necessário.

O acesso a funcionalidades que permitam conferências mais convenientes é também um bom ponto de partida, como o exemplo da partilha de documentos, transcrição de fala para texto e temporizador de conferência. Poder definir tópicos de discussão ao longo da conferência pode facilitar reuniões

## 7.3. Considerações finais

Decorreu quase um ano de estágio e foi sem dúvida um trabalho bastante desafiante, provando ser um projeto interessante e inovador. Diariamente fui enfrentando vários desafios, grande parte deles ultrapassados. As primeiras semanas do estágio poderão ter sido as mais intensas no sentido em que é necessário iniciar um projeto com alguma complexidade de raiz. Inicialmente, o âmbito do projeto era-me desconhecido e necessitei de ajuda na sua compreensão. Após analisar o âmbito do estágio com maior detalhe juntamente com os orientadores e definir as funcionalidades, este processo tornou-se mais simplificado.

O projeto desenvolvido é um projeto de *full-stack*, sendo que foi realizado trabalho no *frontend* juntamente com *backend*. Trabalhar em simultâneo destas duas componentes vitais revelou-se um desafio dado que o trabalho era extenso. Algumas das funcionalidades planeadas no início do estágio não foram implementadas por se terem revelado mais complexas e morosas. Mesmo assim as componentes principais do trabalho foram implementadas com sucesso e sinto-me satisfeito com o meu trabalho.

O ambiente de trabalho foi o adequado, obtendo suporte dos meus orientadores e trabalhadores da empresa sempre que necessitava e em bastantes ocasiões ajudaram-me a ultrapassar problemas tanto a nível da gestão do projeto como na implementação de funcionalidades. Senti-me respeitado e que podia expressar a minha opinião em diferentes matérias.

## Referências

- [1] nefsis, “Video conferencing history,” [Online]. Available: <http://www.nefsis.com/best-video-conferencing-software/video-conferencing-history.html>. [Acedido em Outubro 2015].
- [2] H. Alvestrand, ““Google release of WebRTC source code,”” 1 Junho 2011. [Online]. Available: <http://lists.w3.org/Archives/Public/public-webrtc/2011May/0022.html>.
- [3] WIT Software, S.A., “Why WIT,” [Online]. Available: <https://www.wit-software.com/company/why-wit/>. [Acedido em Junho 2016].
- [4] S. Dutton, “Getting Started with WebRTC,” 2014. [Online]. Available: <http://www.html5rocks.com/en/tutorials/webrtc/basics/>.
- [5] R. Pan, “WebRTC overview,” 2014. [Online]. Available: <http://pt.slideshare.net/rouyunpan/web-rtc-overview>.
- [6] Audio-Video Transport Working Group, “RTP: A Transport Protocol for Real-Time Applications,” 1996.
- [7] “Can I use... WebRTC,” [Online]. Available: <http://caniuse.com/#search=webrtc>.
- [8] StatCounter, “Web browser global usage share,” [Online]. Available: <http://gs.statcounter.com/>.
- [9] NetMarketShare, “Mobile web browsers global usage share,” [Online]. Available: <https://www.netmarketshare.com/>.
- [10] &yet, “Is WebRTC ready yet?,” [Online]. Available: <http://iswebrtcreadyyet.com/>.
- [11] Google Developers, “WebRTC Update,” [Online]. Available: [https://www.youtube.com/watch?v=GBAEG\\_RuqE&t=29m20s](https://www.youtube.com/watch?v=GBAEG_RuqE&t=29m20s).
- [12] C. Ball, “Serverless WebRTC,” [Online]. Available: <https://github.com/cjb/serverless-webrtc/>.
- [13] M. Gechev, “Multi-User Video Conference with WebRTC,” 2014. [Online]. Available: <http://blog.mgechev.com/2014/12/26/multi-user-video-conference-webrtc-angularjs-yeoman/>.
- [14] Kurento Technologies, “Kurento,” [Online]. Available: <https://www.kurento.org/>.
- [15] Kurento Technologies, “One to many video call,” 2015. [Online]. Available: <http://doc-kurento.readthedocs.io/en/stable/tutorials/java/tutorial-one2many.html>.

- [16] WIT Software, S.A., “WIT WebRTC Gateway,” 2016. [Online]. Available: <https://www.wit-software.com/products/webrtc/>. [Acedido em Junho 2016].
- [17] Facebook, “Flux Architecture Overview,” [Online]. Available: <https://facebook.github.io/flux/docs/overview.html>.
- [18] J. Maddalone, “React Fundamentals course,” egghead.io, [Online]. Available: <https://egghead.io/series/react-fundamentals>.
- [19] Pivotal Software, “Spring Framework,” [Online]. Available: <https://spring.io>.
- [20] G. G. Bernardo, “WebRTC beyond one-to-one communication,” [Online]. Available: <https://webrtcchacks.com/webrtc-beyond-one-one/>.
- [21] J. Ott, “SIP Conferencing,” em *IIR - SIP Congress 2001*, Stockholm, 2001.
- [22] J. T. Johnson, “SIP: Videoconferencing's secret sauce?,” Network World, 2009. [Online]. Available: <http://www.networkworld.com/article/2236430/sip--videoconferencing-s-secret-sauce-.html>.
- [23] S. Dutton, “WebRTC in the real world: STUN, TURN and signaling,” HTML5 Rocks, 2013. [Online]. Available: <http://www.html5rocks.com/en/tutorials/webrtc/infrastructure/>.
- [24] A. Zeidan, A. Lehmann e U. Trick, “WebRTC enabled multimedia conferencing and collaboration solution,” Frankfurt, Germany, 2014.
- [25] M. Khan, “WebRTC Experiments & Demos,” [Online]. Available: <https://www.webrtc-experiment.com>.
- [26] G. Politis e M. Reavy, “Firefox multistream and renegotiation for Jitsi Videobridge,” Mozilla, 2015. [Online]. Available: <https://hacks.mozilla.org/2015/06/firefox-multistream-and-renegotiation-for-jitsi-videobridge/>.
- [27] BlogGeek.me, “How To Implement Multipoint Video Using WebRTC: Introduction,” 2013. [Online]. Available: <https://bloggeek.me/multipoint-webrtc-intro/>.
- [28] “Spring by Example,” 2015. [Online]. Available: <http://www.springbyexample.org>.
- [29] M. Rouse, “Real-Time Transport Protocol (RTP) definition,” TechTarget, 2007. [Online]. Available: <http://searchnetworking.techtarget.com/definition/Real-Time-Transport-Protocol>.



## Apêndice

### A – Screenshots de aplicações de conferência

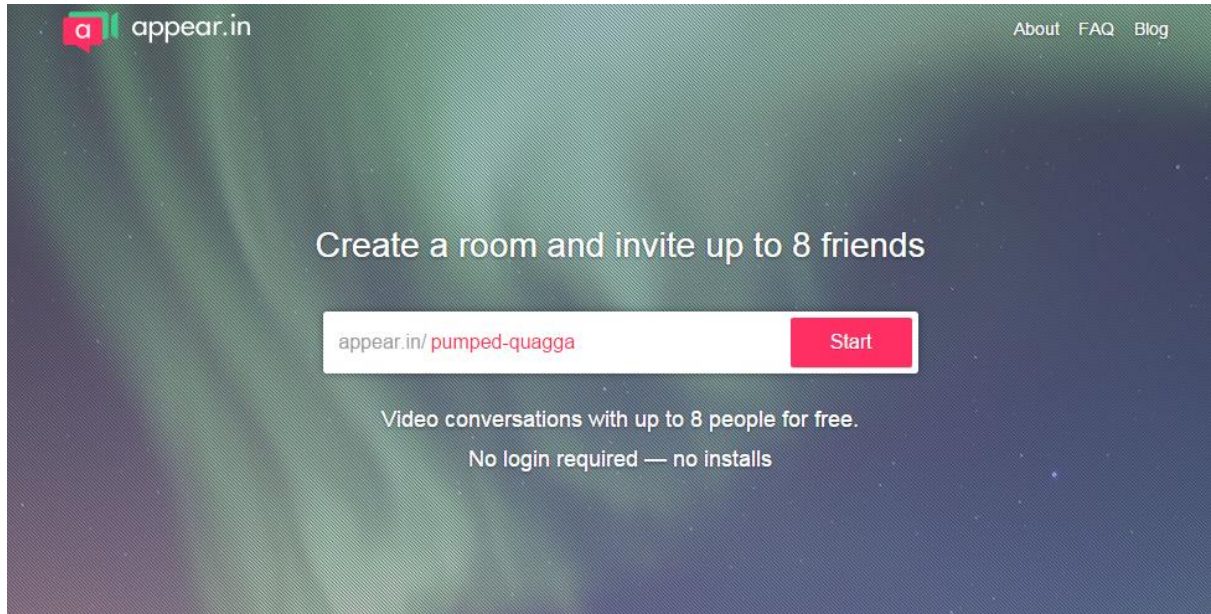


Figura 0-1 - *Appear.in*: Ecrã principal

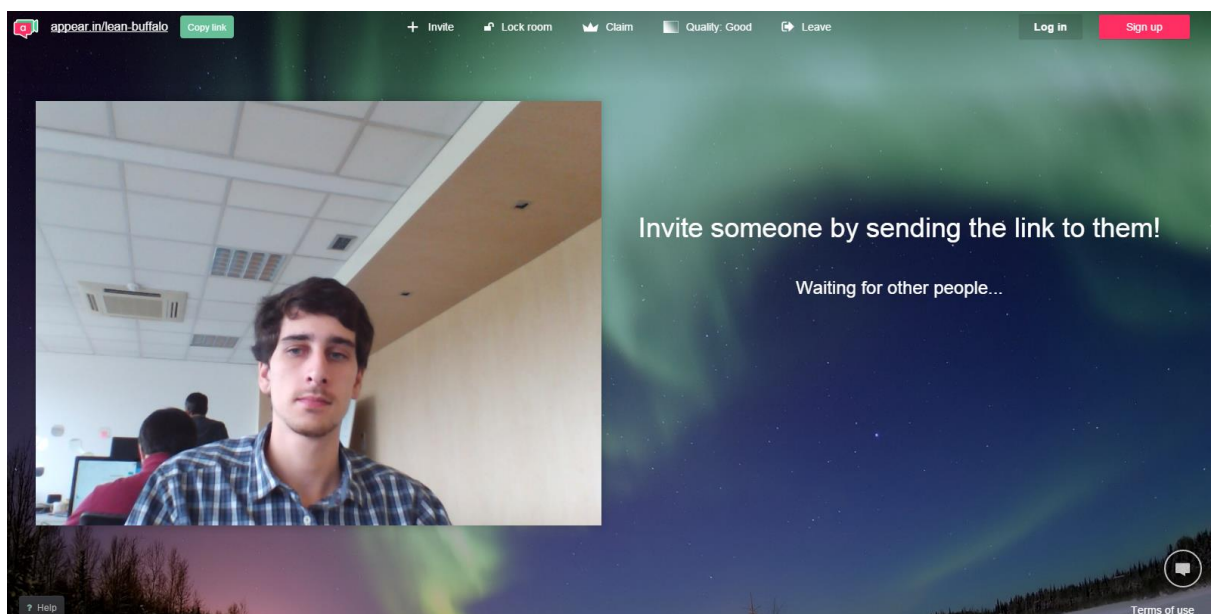


Figura 0-2 - *Appear.in*: sala de conferência

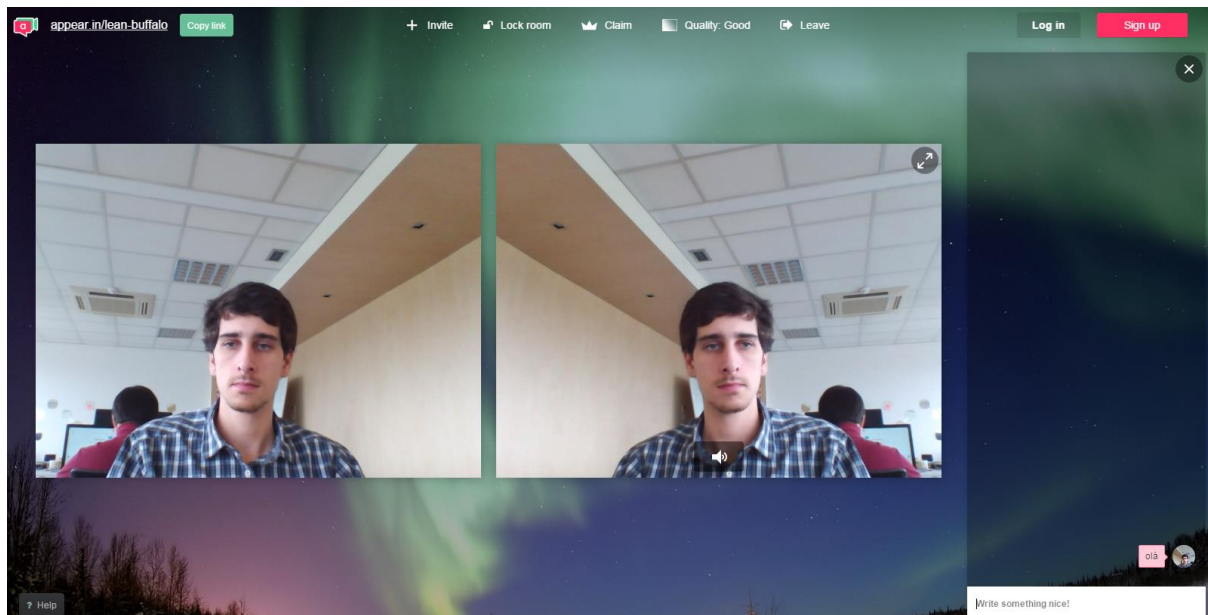


Figura 0-3 - *Appear.in: Sala de conferência com 2 participantes*

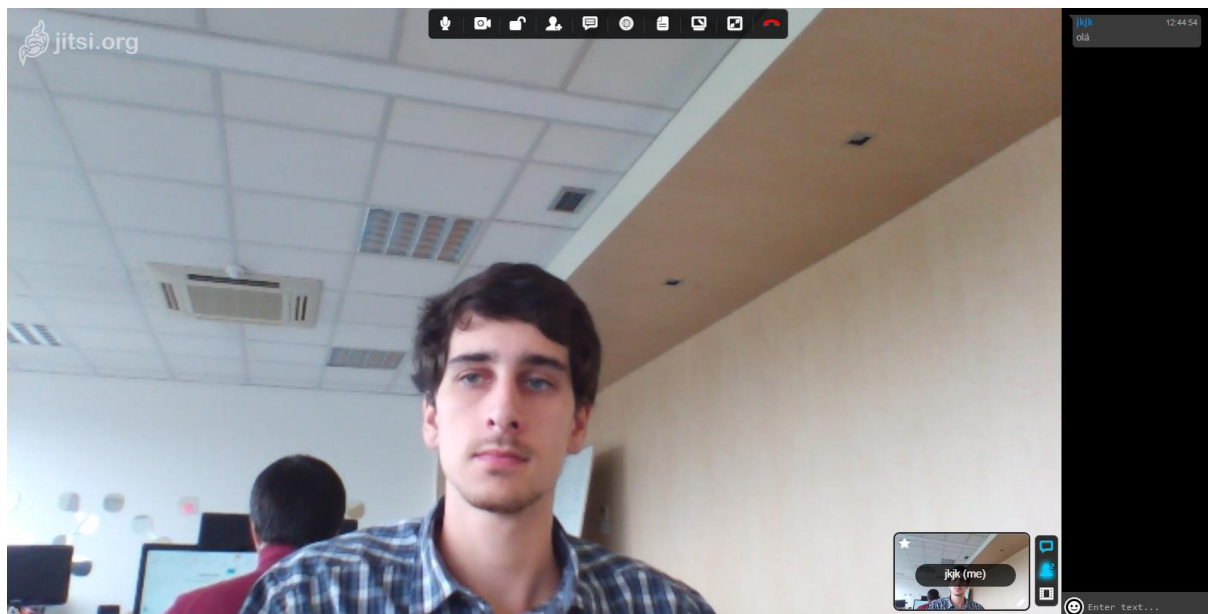


Figura 0-4 - *Jitsi Meet: sala de conferência*

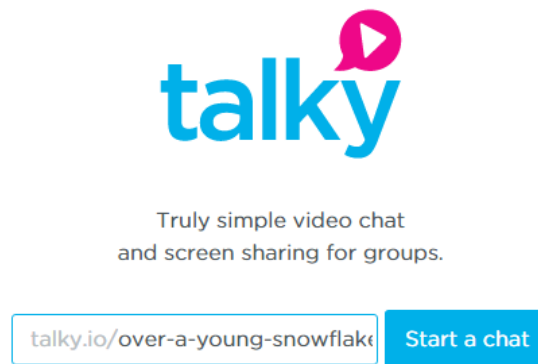


Figura 0-5 - Talky.io: ecrã principal

## Ready to join a video chat?

Talky is truly simple video chat and screen sharing for groups.  
[Learn more about talky.](#)

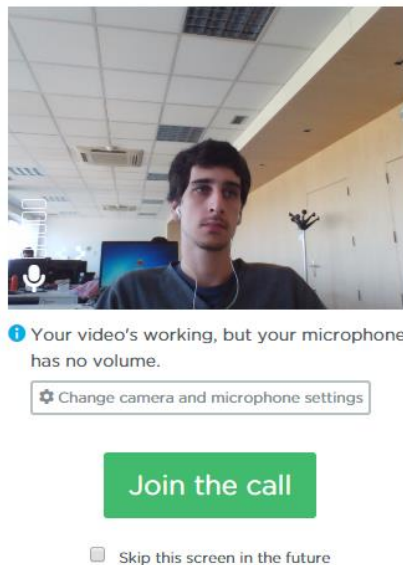


Figura 0-6 - Talky.io: ecrã pré-conferência

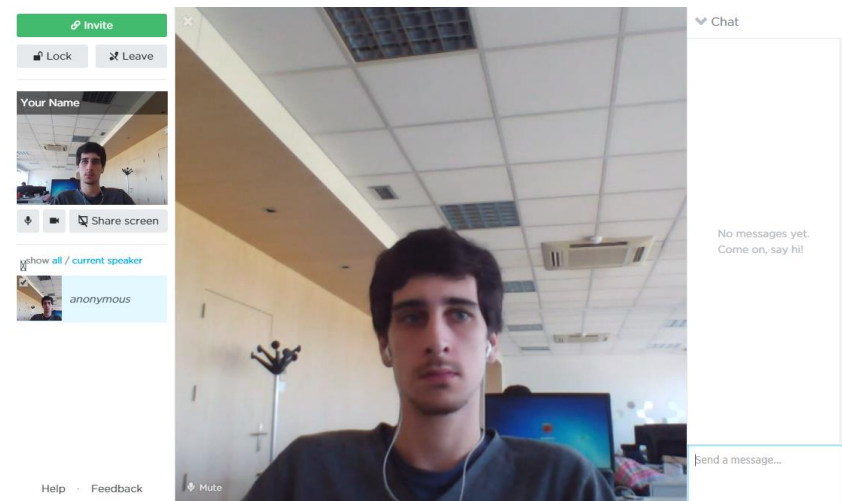


Figura 0-7 - Talky.io: sala de conferência

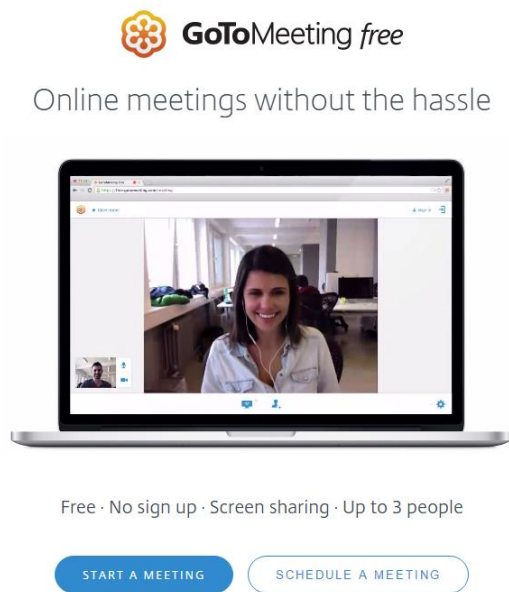


Figura 0-8 - GoToMeeting: ecrã principal



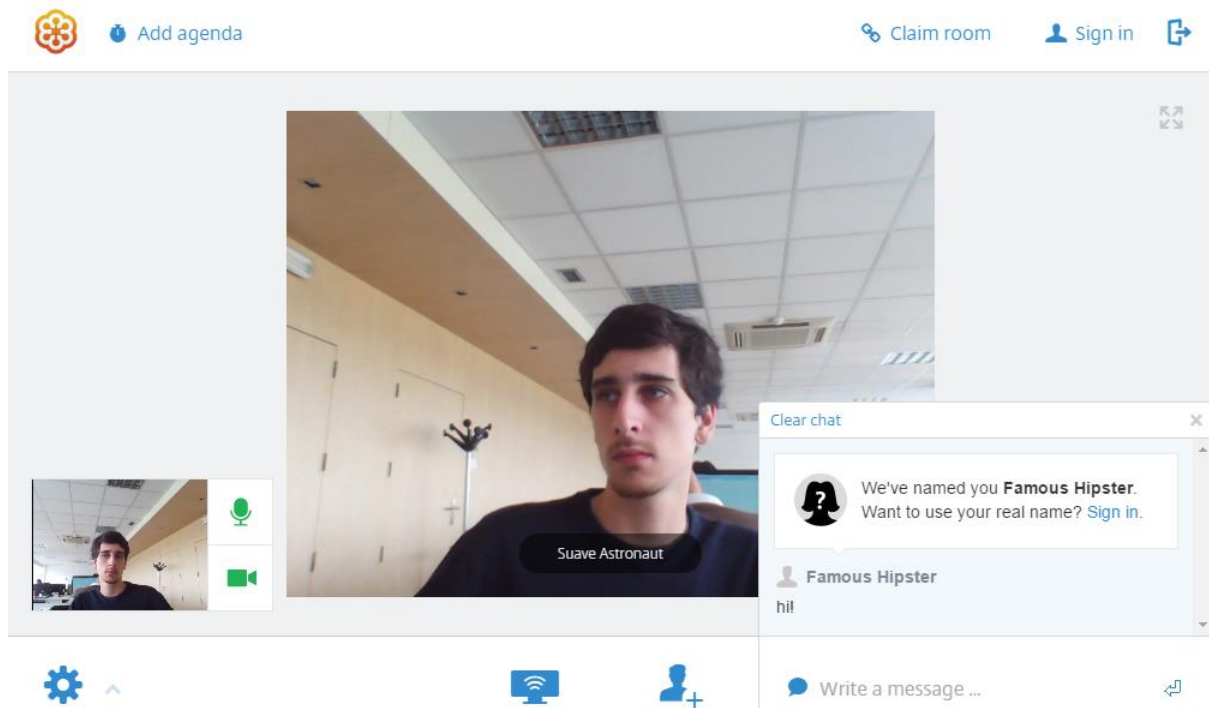
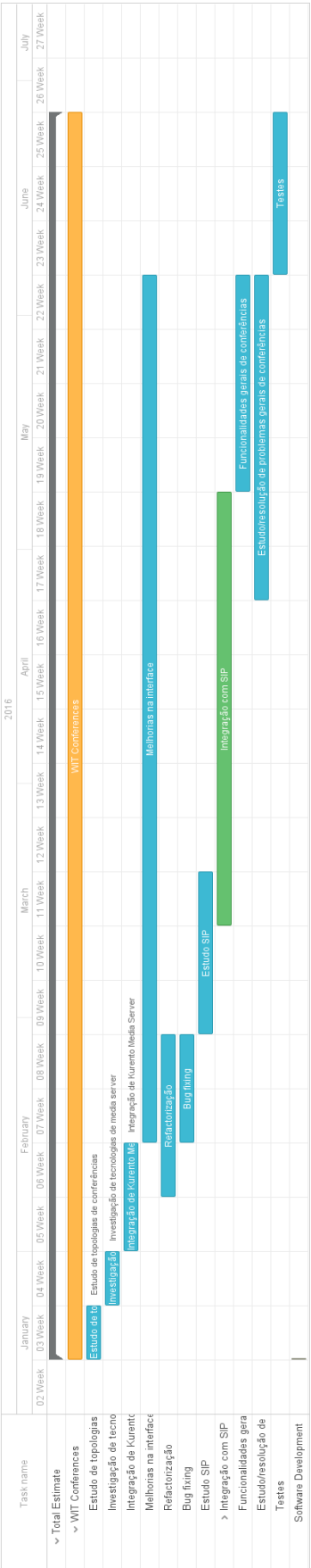


Figura 0-9 - GoToMeeting: sala de conferência

B – Imagens e Diagramas





## **Anexos**

A – Abordagem

B – Arquitetura

C – Qualidade de Software