UNIVERSIDADE FEDERAL DO ABC
CENTRO DE ENGENHARIA, MODELAGEM E CIÊNCIAS SOCIAIS APLICADAS
ENGENHARIA DE INFORMAÇÃO

BRUNO MARQUES DOS SANTOS PANASIO

EMPREGO DE WEBRTC PARA SISTEMA DE ATENDIMENTO AO CLIENTE

Santo André, SP
2018

BRUNO MARQUES DOS SANTOS PANASIO

EMPREGO DE WEBRTC PARA SISTEMA DE ATENDIMENTO AO CLIENTE

Trabalho de Graduação apresentado ao curso de Engenharia de Informação, do Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas, da Universidade Federal do ABC, como requisito parcial para a Obtenção do grau de Engenheiro de Informação.

Santo André, SP

2018

BRUNO MARQUES DOS SANTOS PANASIO

EMPREGO DE WEBRTC PARA SISTEMA DE ATENDIMENTO AO CLIENTE

Trabalho de Graduação apresentado ao curso de Engenharia de Informação, do Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas, da Universidade Federal do ABC, como requisito parcial para a Obtenção do grau de Engenheiro de Informação.

Santo André, SP, 26 de novembro de 2018.

BANCA EXAMINADORA

Prof. Dr. Amaury Kruel Budri

Universidade Federal do ABC

Prof. Dr. Luiz Henrique Bonani do Nascimento

Universidade Federal do ABC

Prof. Dr. Marco Aurélio Cazarotto Gomes

Universidade Federal do ABC

Dedico este trabalho aos meus familiares, amigos e minha esposa que sempre me incentivaram.

Agradecimentos

Agradeço aos meus professores e colegas pelo aprendizado obtido ao longo de todos os anos de curso da faculdade. Agradeço a minha família e amigos por me ajudarem nos momentos mais difíceis. Agradeço especialmente minha esposa pelo incentivo e força ao longo destes anos.

RESUMO

Este trabalho de graduação apresenta um estudo sobre o emprego da tecnologia WebRTC para sistemas de atendimento ao cliente. O WebRTC é um conjunto de protocolos e APIs de código aberto para comunicação de mídias em tempo real pela internet. Com a utilização das APIs em JavaScript do WebRTC é realizada a programação e execução de suas funções nos principais navegadores de internet. Com o WebRTC é possível desenvolver aplicações para atingir simultaneamente qualquer plataforma que utilize navegadores *web*, como Android, iOS, Windows, MacOS, entre outros, independente do hardware utilizado. Para validar as funcionalidades do WebRTC, um estudo de caso de um serviço de central de atendimento ao cliente foi criado usando a plataforma de *contact center* em nuvem da empresa Velip, que é baseada em WebRTC.

Palavras-chave: WebRTC, Contact Center, Call Center, Central de atendimento, cloud, nuvem, multiplataforma.

ABSTRACT

This undergraduate work presents a study about use of WebRTC technology for contact center systems. WebRTC is a set of protocols and open source APIs for-real time web media communication. Using WebRTC JavaScript APIs, the programing and execution of its functions are performed in the main internet browsers. With WebRTC is possible to develop applications to simultaneously reach any plataform that uses web browsers, such as Android, iOS, Windows, MacOS, among others, regardless of the hardware used. To validate WebRTC functionalities, a case study of a customer care service was created using the cloud contact center of company Velip, wich is based on WebRTC.

Key-words: WebRTC, Contact Center, Call Center, Customer Care, cloud, multiplataform.

SUMÁRIO

1. Introdução

1.1. Motivação

A criação de uma posição de atendimento em um *contact cente*r convencional exige vários recursos de infraestrutura, tais como: computador, hardphone (aparelho físico) ou um softphone (software instalado no computador que simula um aparelho), servidores, redes e recursos humanos para o projeto. Por exemplo, podemos elencar os seguintes custos para uso de um hardphone, entre outros: compra de aparelhos, licenças unitárias, infraestrutura de servidores para o PABX, manutenção do equipamento. O *hardphone* tem apenas a capacidade de tráfego da mídia voz. No caso dos custos de um *softphone,* podem ser elencados: licenças unitárias, trabalho de homologação e testes de instalação, infraestrutura de servidores. Durante o planejamento de um projeto de uma central de atendimento, uma das peças chaves é o custo, dividido em itens como: aquisição da infraestrutura, instalação, manutenção e suporte.

A convergência para aplicativos *Web* e *Cloud Computing* tem mostrado suas vantagens ao longo do tempo em relação a escalabilidade, contingência e custos. A tecnologia WebRTC (Web Real-Time Comunications) é um conjunto de APIs em JavaScript que permite a criação de aplicações web, dispensando a necessidade de compra de aparelhos dedicados e instalação de softwares ou plugins nos computadores (pois é uma tecnologia embutida no navegador). Sua atualização é feita pelo código fonte no servidor, de uma maneira invisível para o usuário. Além disso o WebRTC permite o tráfego de outras mídias além do áudio, tais como: texto, vídeo, troca de arquivos e co-browsing.

O uso das APIs do WebRTC para desenvolvimento em smartphones aumenta a expectativa de abrangência desta tecnologia. Portanto oferece comodidade por ser acessível em qualquer lugar. Esta característica é importante para as novas gerações, como os *Millenials*. Por ser uma tecnologia nascida para o ambiente web, tem a facilidade de ter sua infraestrutura desenvolvida em *cloud computing*. Outro ponto importante é que com o HTML5 e JavaScript todas as plataformas, computadores e

mobile, podem ter acesso independente do sistema operacional (SO). Sem a necessidade de desenvolvimento para cada uma das inúmeras plataformas e seus SOs.

Este trabalho consiste em um estudo da aplicação da tecnologia WebRTC como solução para sistemas de atendimento ao cliente. Para constatação da utilidade prática do WebRTC como solução, foi realizado um estudo de caso com um serviço de *Contact Center* em nuvem da empresa Velip. Este serviço utiliza as APIs do WebRTC pelo JsSIP como solução para chamadas de voz.

1.2. Objetivos

O objetivo deste trabalho de graduação é analisar o emprego da tecnologia WebRTC no ambiente de centrais de atendimento ao cliente.

1.3. Organização do trabalho de graduação

No Capítulo 1 de introdução, é apresentado um resumo geral dos problemas atuais do projeto de uma central de atendimento, e como o WebRTC pode ser empregado em *contact centers*.

O Capítulo 2 descreve a importância econômica das centrais de atendimento e o cenário tecnológico de nuvem em que o WebRTC deve ser aplicado.

Os conceitos apresentados no Capítulo 3 abordam os pilares tecnológicos para o funcionamento do WebRTC. Tecnologias como VoIP e seus protocolos, linguagens web e APIs, e a arquitetura do framework WebRTC fazem parte do conteúdo.

O Capítulo 4 ilustra casos de sucesso no uso do WebRTC no mundo e no ramo de *contact center*.

O Capítulo 5 relata um estudo de caso de uma ferramenta em *cloud* que utiliza a tecnologia WebRTC como solução de central de atendimento.

2. Contextualização do cenário

2.1. Centrais de Atendimento - Importância e modo de operação.

As tecnologias de comunicação possibilitam uma maneira rápida e fácil de contato com as pessoas. O *Call Center* é o ramo de negócios que se utiliza dessas tecnologias para realizar anúncios de produtos e serviços de atendimento ao cliente pelo o telefone. Os termos mais comuns para a área de gestão do relacionamento com o cliente são, entre outros: *call center*, *contact center*, telemarketing, SAC (Serviço de atendimento ao cliente), centrais de atendimento, *custom care*. “Uma central onde as chamadas são processadas ou recebidas, em alto volume, com objetivos ligados às funções de venda, marketing, serviço ao consumidor, telemarketing, suporte técnico e qualquer outra atividade administrativa especializada" (MADRUGA, 2015). Esses objetivos ganharam diversidade de canais de atendimento como: email, chat, SMS e redes sociais.

As centrais de atendimento são ferramentas competitivas financeiramente para as empresas. Segmentos como catálogos, vendas a varejo e principalmente serviços financeiros, movimentam milhões de reais diariamente.

Os setores de convergência (mídia, internet e conteúdo) e o financeiro são os que mais contratam os serviços de centrais de atendimento no Brasil, com 43,4% e 36,2% da demanda respectivamente, de acordo com relatório de Sintelmak (2016).

Os aspectos técnicos da operação de um *call center* são classificadas como *Inbound* e *outbound*, de acordo com a descrição de Bergevin et al. (2010).

- *Inbound*: Operações em que o contato é gerado pelo cliente em direção à central de atendimento. Um atendimento específico para suporte ou resolução de problemas. Como boa prática para um melhor atendimento, a central possui uma URA (unidade de resposta audível) que filtra e direciona o cliente para operadores com melhores habilidades no assunto.

- *Outbound*: É a operação que inicia o contato com o cliente. Oferecer um serviço ou realizar uma cobrança são exemplos de motivadores de ligações

outbound. Para ganhar eficiência, grandes *call centers* utilizam um software que dispara várias chamadas simultâneas utilizando uma lista de contatos. O software identifica automaticamente quando a ligação é atendida pelo cliente, e só então encaminha a chamada para um operador.

Tradicionalmente um *call center* simples com operações do tipo *Inbound* tem o seguinte exemplo de fluxo de operação:

a. A chamada é realizada pelo cliente para um número da empresa. Esse número é gerenciado por uma empresa de telecomunicações, que trabalha com o fornecimento de serviços de acesso à PSTN (*public switched telephone network*).
b. A empresa de telecomunicações fecha o circuito de comunicação da chamada com um equipamento de telefonia da empresa de *call center*.
c. Por sua vez, a empresa direciona a chamada para o operador realizar o atendimento. Este fluxo é ilustrado na Figura 1, nas linhas a, b e c.

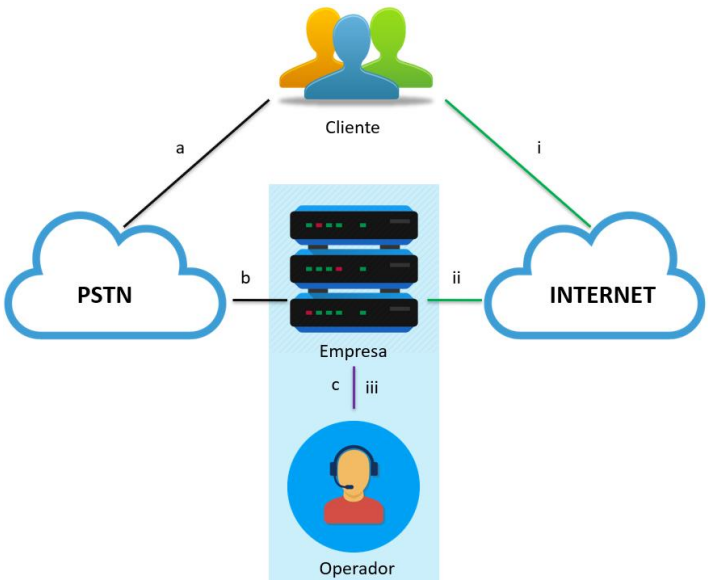


Figura 1. Fluxo simplificado de uma chamada para atendimento em um *call center*.

Outra opção é o contato do cliente por meio de canais digitais, entre outros: chat, email, voz e videoconferência. Para esses canais digitais o fluxo da direita da Figura 1, nas linhas i, ii e iii, ilustra da seguinte forma:

1. Acessando um link de internet entre cliente e operadora de telecomunicações (i), o cliente acessa o servidor por meio de um site que está hospedado o serviço de atendimento por canal digital (ii).
2. A empresa direciona o contato do cliente para um operador (iii) com a habilidade específica para atender o serviço disponibilizado.

Para realizar o atendimento, os operadores das centrais de atendimento necessitam de vários recursos de hardware e software. Por exemplo, os itens da seguinte lista:

1. Computador com acesso à rede.
2. Softwares básicos.
    - Sistema operacional.
    - Sistema CRM.
    - Sistemas de *Call Center.*
        - CTI – Estratégia de roteamento das chamadas.
        - Telefonia – PABX.
        - WFM (*Work force management*) – Software responsável pelo cálculo de rendimento e escala de operadores.
        - Gravação de chamadas.
3. Telefone físico (caso não seja um softphone).
4. Headset.

Todos os sistemas listados, exigem ao menos uma ou mais licenças de software atreladas ao uso das funcionalidades. É comum que para cada sistema seja necessário um software instalado por computador, afim de coletar e disponibilizar informações. A instalação de vários softwares onera o sistema, que ao longo do tempo vai deixando o computador mais lento se não houver manutenções periódicas. As atualizações de software custam esforços de recursos para acontecer. Quanto maior o parque de máquinas, maior o trabalho e chances de erros. Cada sistema e sua infraestrutura de servidores e rede participam dos custos uma central de atendimento.

2.2. Convergência dos meios de comunicações e suas tecnologias para a internet.

A internet revolucionou a comunicação ao redor do mundo e continua com os avanços computacionais e tecnologias de telecomunicação, segundo Gião (2010). Todo esse avanço reflete também na comunicação entre organizações e clientes. A redução da distância entre organizações e as comunidades e pessoas por meio da internet pode reduz significativamente os custos das comunicações conforme análise de Gião (2010). Os *call centers* utilizam dessa redução de custos com as inovações tecnológicas baseadas em VoIP (Voice over IP) e aplicativos móveis. “O custo médio do atendimento ao cliente no balcão pode custar R$12,00, por telefone R$6,00 e apenas R$1,00 para obter o mesmo resultado em um atendimento digital” (GIÃO, 2010). Esses valores mostram o atendimento mais barato realizado pela internet do que qualquer outro meio de comunicação.

Outro ponto importante de destaque sobre a evolução das comunicações são as aplicações *mobile* via internet, como Skype e WhatsApp, por exemplo. Estes são aplicativos que realizam comunicação entre pessoas, e que podem substituir as chamadas e SMS das operadoras de telecomunicações. Os aplicativos diminuem diretamente o lucro das operadoras de telefonia em relação aos serviços de voz, segundo Congo (2013). Porém a convergência dos serviços telefônicos via internet é um caminho bem-visto, pois uma prova são alguns planos de dados oferecidos atualmente. Esses planos não debitam da conta do cliente a utilização do WhatsApp e Facebook. Em ambas as plataformas (Whatsapp e Facebook) é possível se comunicar via texto, voz e até vídeo, além da troca de arquivos entre os contatos. Estas mesmas operadoras de telecomunicações que tem sua receita reduzida nos serviços de voz, e sua receita ampliada nos pacotes de dados comprados pelos usuários.

Eduardo Navarro, presidente da Telefônica Brasil, afirmou em entrevista à Portinari (2017) para Folha, que o foco do investimento de R$ 4,8 bilhões entre 2017 e 2019 será em serviços digitais. Visando melhorar o relacionamento com o cliente, ele afirma também que o objetivo é acabar com o *call center* da maneira tradicional, e trocar esse serviço por interações digitais em aplicativos. Nessa entrevista, ele expõe o ponto de vista de que há uma migração do atendimento dos clientes para os meios

digitais do tipo não voz, como por exemplo os aplicativos de chat e *self-service* para alterações e contratações de planos. Eduardo afirma também que caso o cliente queira utilizar o *call center*, continuará disponibilizando este serviço, mas entende que cada vez menos será requisitado. Uma prova disso é a queda em 8% das chamadas comparando 2016 e 2017. As pessoas estão cada vez mais utilizando aplicativos não só para se comunicar, mas também para resolver seus problemas. Com o *self-service* via aplicativos, é possível obter a liberdade de resolver sozinho sem a necessidade de passar por um intermediário, que normalmente realiza inúmeras perguntas, gastando muito mais tempo do cliente.

Empresas do setor bancário também acompanham a migração dos meios das transações e canais de comunicação há alguns anos. Em matéria do G1, Alvarenga (2017) mostra os números das operações em canais digitais registradas pelos bancos ao longo de três anos. A matéria foca na inversão do volume de operações dos canais tradicionais para os digitais. A análise da ilustração da Figura 2 ilustra que em 2014 a quantidade de operações era muito próxima entre os canais, mas em 2015 os canais digitais superaram os tradicionais, e em 2016 as operações digitais representaram quase 62% do volume total.

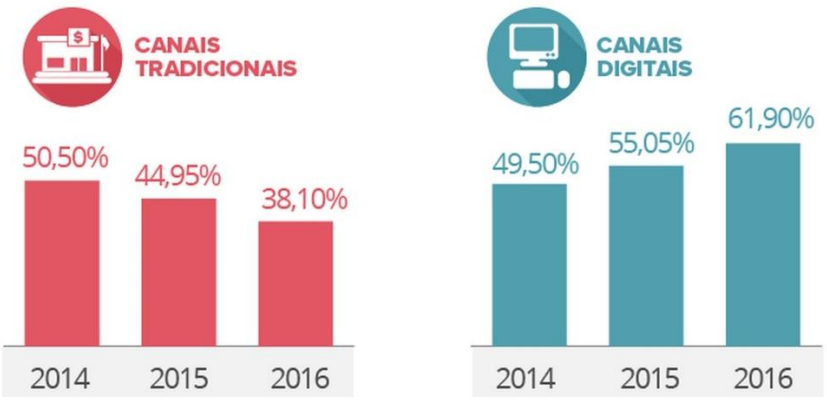


Figura 2. Distribuição das operações feitas pelos clientes de banco por canal de acesso. Fonte: ALVARENGA (2017).

Ambas as notícias citadas apontam para indicadores da mudança para os canais digitais. Esses relatos apontam como as empresas levam em conta na hora investir em suas tecnologias, influenciando seu mapa de tecnologia para os próximos anos. Outro ponto importante é o cultural, nos quais os *Millenials* já são usuários desses serviços. As empresas mudam seus serviços para se ajustar a esse público,

que já nasceu mais conectado ao mundo digital. A geração dos *Millenials* tem o poder real de compra considerável, segundo a ClienteSA (2016, p. 12). Com perfil mais tecnológico, criativo e empreendedor, essas características são traduzidas para o tipo de consumo dos *Millennials*(nascidos entre a década de 90 e 2000). Essa geração é a primeira que usa a internet desde cedo e, quanto mais novos, mais conectados são via celular e às redes sociais, que também podem ser um canal de atendimento digital.

As empresas de tecnologia do setor de atendimento ao cliente também apontam para um cenário digital multicanais. O “Quadrante mágico” da Gartner, conceituada companhia de pesquisa e consultoria mundial em fornecedores de infraestrutura de TI, mostra as maiores empresas do setor. Esse “Quadrante mágico” das empresas de *Contact Center* está ilustrado na Figura 3. A companhia as classifica em duas direções, Visão e habilidade de execução, e em quatro quadrantes que são os fornecedores de nicho, desafiadores, visionários e líderes. Com base nesses dados, grandes empresas no mundo utilizam o Gartner como apoio para tomada de decisão de qual tecnologia e fornecedor adotar.

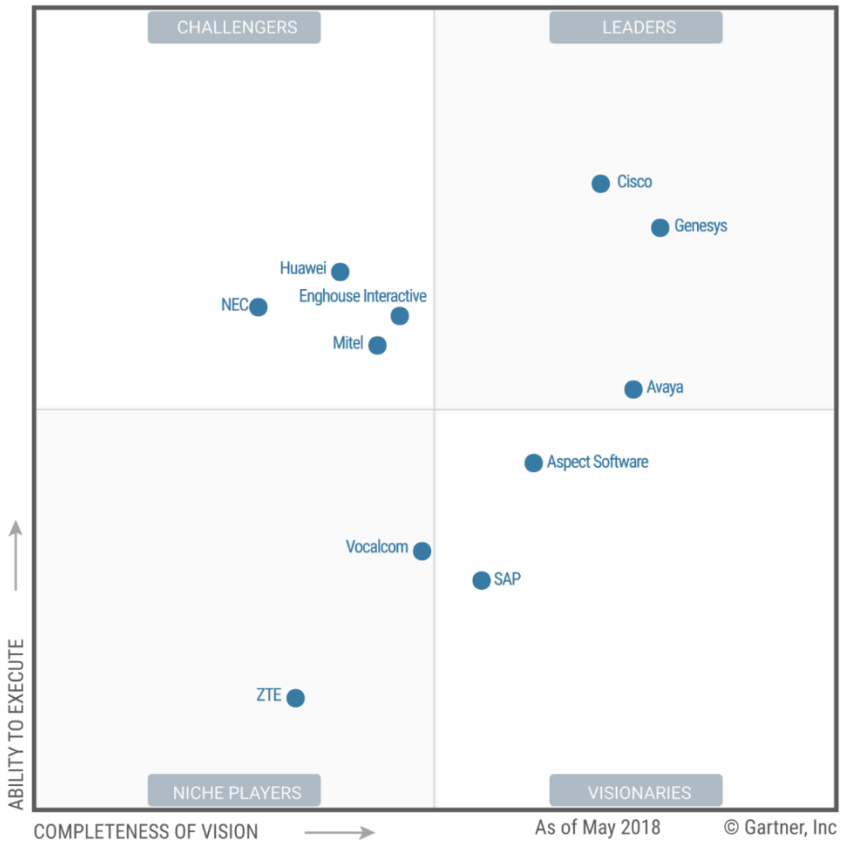


Figura 3. Quadrante Mágico Gartner para fornecedores de infraestrutura de *Contact Center*. Fonte: Gartner, Inc 2018.

Ao analisar dois fornecedores de tecnologia desse quadrante, Genesys e Avaya, bem posicionados nos quesitos de liderança e visão respectivamente, possuem soluções multicanais em seus últimos produtos para os setores de centrais de atendimento. A Avaya provê serviço de atendimento em multicanais, como texto, mídias sociais, email e chat, além da tradicional voz, com seu produto ***Avaya Aura ® Contact Center***. A Genesys também oferece as mesmas funcionalidades, e destaca ainda uma abordagem da melhor experiência do cliente, com um produto que conecta todas as interações realizadas em diversos canais para oferecer uma experiência dinâmica e personalizada ao longo de toda jornada do cliente, pelo ***Customer Journey Management***. Ambos os produtos focam o atendimento em multicanais. Este conceito é de que independentemente do canal que o cliente inicie uma conversa, é possível ter histórico de todo o contexto de interações deste mesmo cliente nos outros canais.

Com a melhora na qualidade do acesso à internet, o trabalho do tipo *home office* vem crescendo. Esse modelo de trabalho propõe que o funcionário ou prestadores de serviço realize suas atividades de casa. Nesse modelo, ambas as partes saem ganhando, pois quem presta o serviço não precisa se deslocar até o trabalho, e a empresa economiza em infraestrutura com menos posições de trabalho.

As tecnologias em nuvem e a automatização são tendências para o ambiente de *call centers,* afirma Antonelli (2015). Compartilhando a visão de diretores de empresas ele afirma que o investimento nessas tecnologias visa redução de custos, facilidades de uso e certeza de sistemas mais atualizados. Existe a demanda de trabalhos do tipo *home office* que necessitam realizar atendimento ao cliente, como nos casos de suporte remoto, indicando o WebRTC como facilitador para tais serviços, de acordo com Antonelli (2015).

A tendência de *home office* vem se confirmando devido a fatores como redução de custos, flexibilidade na jornada e até qualidade de vida, conforme analisa a matéria da Revista Call Center (2016). É possível eliminar as dificuldades de realizar um atendimento via *home office* com o emprego do WebRTC, pois todas as operações, inclusive receber ou fazer chamadas, agora são feitas apenas utilizando um navegador (REVISTA CALL CENTER, 2016).

Outra frente de mudanças tecnológicas que vem se estabelecendo é a computação em nuvem (*cloud computing*). Essa filosofia de armazenamento e troca de informação em um ambiente online e altamente disponível, e isso traz várias vantagens para o desenvolvimento de novas tecnologias baseadas em software. O relatório técnico da Universidade de Berkeley na Califórnia enumera alguns obstáculos e oportunidades que vem se consolidando ao passar dos anos (ARMBRUST, 2009). Estes obstáculos estão na Tabela 1.

	Obstáculo	Oportunidade
1	Disponibilidade	Usar múltiplos provedores de cloud; Uso da elasticidade para prevenir ataques DDOS
2	Confidencialidade e auditoria de dados	Criptografia da implementação; VLANs; Firewalls; Armazenamento de dados espalhado geograficamente
3	Gargalo na transferência de dados	Switches com grande largura de banda; backup e arquivamento de dados;
4	Performance estável	Suporte de máquinas virtuais (VM), memórias flash
5	Armazenamento escalável	Invenção de Loja de armazenamento escalável
6	Agilidade de escalabilidade	Invenção de Auto-Scanner das VMs, conservação de *Snapshots* (criar novas máquinas a partir de uma imagem pronta)
7	Licenciamento de software	Pague por uso de licenças; vendas de uso em massa

Tabela 1 Obstáculos e oportunidades da computação em nuvem enumerados pelo relatório técnico da Engenharia Elétrica e da Ciência da Computação da Universidade de Berkeley na Califórnia (ARMBRUST, 2009).

Em suma, um ambiente em nuvem pode fornecer o acesso e a infraestrutura de servidores para aplicações rodarem sem interrupções, com alto desempenho, rápida escalabilidade (aumentar ou diminuir a quantidade de servidores ou armazenamento de acordo com a demanda) e contingência geográfica, afim de

garantir o perfeito funcionamento de uma aplicação. Isso tudo sem a necessidade de espaço físico do usuário, e pagando apenas pelo uso da quantidade selecionada durante determinado período. Essa forma de cobrança evita o desperdício de recursos, que normalmente acontecem ao estimar uma infraestrutura para um novo *data center*. Com isso é possível ter uma infraestrutura de *Data Center* sem comprar nenhum servidor e nem se preocupar em adquirir infraestrutura de rede.

Com aplicativos na nuvem é possível oferecer custo baseado em utilização, e não mais em instalações e mensalidades. Essa é uma tendência do mercado chamada *SaaS* "*Software as a service*".

Existem empresas que tem serviços de atendimento ao cliente que utilizam tecnologias defasadas para prestação de serviços. Com isso é gasto muito dinheiro com infraestrutura que não é de última geração, como por exemplo, a utilização de cabos coaxias com conectores E1 para troncos telefônicos. Os "E1's" são gerenciados por equipamentos de alto custo e ocupam enormes espaços em *Data Centers*. Existem aparelhos de telecomunicações que evoluíram para modelos de menor ocupação física com maior quantidade de recurso. Por exemplo, os modens ópticos, que utilizam fibras ópticas com links SDH (Synchronous Digital Hierarchy) e MPLS (Multiprotocol Label Switching) para comunicação rápida e de longa distância com grande capacidade. Nesse quesito, o SIP *Trunking* com a operadora é o modelo mais recente disponível para recebimento de chamadas, e o único naturalmente compatível com o modelo de computação em nuvem.

Ao inovar no desenvolvimento das tecnologias computacionais, é possível obter o mesmo serviço sem possuir nenhuma infraestrutura local pesada de telecomunicações, utilizando apenas um bom link de dados. Isso é possível com a utilização de servidores em nuvem de empresas como Amazon, Google e Microsoft. Chamadas de *IaaS* (*infrastructure as a service*), possuem recursos de alta qualidade e disponibilidade, além de menor custo. Este trabalho de graduação enfatiza essas premissas para a infraestrutura dos canais de atendimento, utilizando o WebRTC como base do *SaaS*.

Para motivar o assunto, pode-se realizar a seguinte reflexão analisando o surgimento das últimas empresas consideradas gigantes globais: Uber que possui o maior serviço do mundo de transporte pessoal terrestre sem precisar comprar nenhum

carro. Airbnb que é a maior rede hoteleira do mundo sem possuir nenhum quarto sequer. Amazon, Ebay e Mercado Livre que vendem de tudo sem possuir nenhum produto físico. Todas essas empresas apenas fazem o intermediário das negociações, e tem suas aplicações rodando em um ambiente de computação em nuvem.

Tomando como inspiração as empresas citadas, é possível oferecer serviços de *Contact Center* no mesmo molde de intermediação, com modelo de negócio similar. A ideia é que o usuário se cadastre no site ou aplicativo para oferecer seu tempo para atender um cliente de uma empresa. A empresa por sua vez busca no site uma pessoa com perfil esperado para realizar atendimento de suas necessidades. Ambos não necessitariam adquirir grande infraestrutura, nem instalar softwares e comprar licenças. Como pré-requisito apenas de uma simples conexão de internet e um *browser* compatível. De acordo com suas especificações, o WebRTC tem os alicerces para o desenvolvimento de tal plataforma.

3. Conceitos introdutórios

3.1. VoIP

A palavra VoIP é um acrônimo de Voice over Internet Protocol (Voz sobre protocolo IP), e por meio da rede de pacotes IP transporta informação de voz e sinalização. Para que o VoIP seja viável, protocolos específicos de sinalização e de transporte de mídia foram criados para reger essa comunicação de maneira organizada e funcional.

Os protocolos de sinalização têm como objetivo estabelecer e realizar as chamadas, conhecidas como sessões. Em cada sessão há uma variedade de tarefas como, enviar sinal de toque (tom que avisa a chegada de uma chamada), sinal de ocupado e mensagens de falha por exemplo. Informações tais como a identificação de quem está ligando ou sendo chamado, sinais para iniciar e finalizar a cobrança da chamada também fazem parte desse protocolo, de acordo com Olejniczak (2009). Todas as informações necessárias da chamada estão na sinalização, exceto os pacotes IP de mídia (voz no caso de uma ligação). Como principais protocolos de sinalização para chamadas existem o SIP e o H.323.

Os protocolos de transporte têm o papel de transportar os pacotes de mídia pela rede. Estes pacotes são chamados de carga (*payload*), e são a representação da voz em formato digital, previamente convertida de um sinal analógico por um processo de amostragem da voz coletada por um microfone. Os principais protocolos de transporte são em conjunto o RTP e RTCP, definidos no Capitulo 3.1.3.

A utilização desta tecnologia permite também a mobilidade e menor custo, pois ao utilizar uma rede de dados como a internet, possibilita estar acessível para realizar ligações de qualquer lugar. O preço de uma ligação VoIP costuma ser muito menor do que uma ligação realizada pela rede de voz das operadoras.

A regulamentação do VoIP no Brasil é dada por uma licença de serviços multimídia, que é de responsabilidade da Agência Nacional de Telecomunicações – ANATEL, que por sua vez está sob a Lei Geral de Telecomunicações – LGT. Sendo assim, sua regulamentação não enquadra o VoIP como serviço de voz de Telecomunicações, e sim como serviço de valor agregado regido pelo Artigo 61 da LGT. Outra norma recomendada a ser seguida por empresas do ramo de VoIP é a de segurança da informação, definida pela Tríade CID de confidencialidade, integridade e disponibilidade.

3.1.1. SIP – Session Initiation Protocol (Protocolo de Iniciação de Sessão)

O protocolo SIP é um padrão IETF (Internet Engineering Task Force). Sua norma de origem é a RFC 2543 de 1999, e existem dezenas de outras normas complementares. A norma foi atualizada várias vezes e a mais atual é a RFC 3261. O modelo SIP para iniciar sessões de comunicação é do tipo “requisição-resposta”, este modelo utiliza para cada estimulo uma resposta. O SIP trabalha com requisições enviadas para reunir informações ou executar uma ação em uma chamada. Essas requisições são chamadas de métodos SIP. Os métodos SIP são respondidos por códigos de resposta SIP que são traduzidos por exemplo em mensagens de sucesso (código família 2xx) e falha (código família 4xx) (OLEJNICZAK, 2009). Existem dezenas de outros códigos e suas especificidades que podem ser estudados em sua documentação.

O SIP é descrito de acordo com a RFC 2543 do SIP em sua tradução livre:

> "O protocolo de iniciação de sessão (SIP) é um protocolo de controle na camada de aplicação (sinalização) para criar, modificar e finalizar sessões com um ou mais participantes. Essas sessões incluem conferência multimídia pela internet, chamadas pela internet e distribuição de multimídias", tradução livre de Handley et al. (1999).

Uma das tarefas que o SIP realiza durante a sinalização é a negociação dos parâmetros de mídia. Para isso é utilizado o protocolo de descrição de sessão SDP (*Session Description Protocol*) para descrever e negociar os parâmetros de inicialização dos *streamings* de mídia. Porém o SIP não é responsável pelo transporte dos pacotes de mídia. Uma vez a chamada negociada e estabelecida pelo SIP em conjunto com o SDP, a troca dos pacotes e estatística de mídia são realizadas pelos protocolos RTP e RTCP.

3.1.2. H323

O protocolo H.323, diferente do SIP, é recomendação para Sistemas Audiovisuais e Multimídia de maneira mais geral e não especificamente para protocolo IP. O H.323 faz parte da família de recomendações H da ITU-T (International Telecommunication Union Telecommunication Standardization sector) para especificar sistemas de comunicação multimídia de vários tipos de rede, como IPX (Internet Packet Exchange), redes de longa distância (WAN), conexões discadas PPP, além das redes IP.

A ITU-T descreve o padrão H.323 traduzido de maneira livre da seguinte forma:

> "Esta recomendação cobre os requisitos técnicos para um sistema de comunicação multimídia naquelas situações em que o transporte subjacente é uma rede baseada em pacotes (Packet-Based Network – PBN) que pode não fornecer a garantia de qualidade de serviço (Quality of Service – QoS). Estas redes baseadas em pacotes podem incluir LANs, EANs, MAN, IntraNets e InterNetwoks incluindo a internet. São incluídas as conexões *dial up* ou ponto a ponto sobre a GSTN ou ISDN, que usa um transporte baseado em pacote adjacente como o PPP. Esta rede pode consistir em um simples segmento de rede ou ter

> complexas topologias com vários segmentos robustos interconectados por outros links de comunicação” tradução livre da ITU-T H.323.

O H.323 e o SIP tem as mesmas finalidades e padronizam o controle de mídia e serviços, troca de capacidade dos terminais, sinal e rota da chamada. Um se diferencia do outro em alguns aspectos, como o H.323 na interoperabilidade com a rede pública comutada (PSTN), e o SIP para internet e sua grande escalabilidade e flexibilidade.

3.1.3. RTP – Real-time Transport Protocol

Na sua tradução o RTP é o Protocolo de Transporte em tempo Real, que em sua recomendação RFC 3550, traduzida livremente como:

> “O RTP define adequadamente funções de transporte fim a fim na rede para aplicações transmitirem dados em tempo real, como áudio e vídeo, sobre serviços de rede. O RTP não faz a reserva de endereço e não tem garantia de qualidade de serviço para serviços em tempo real. Para isso o transporte de dados é incrementado por um protocolo de controle, RTCP, que permite monitorar a entrega de dados de maneira escalável na rede, e provê funcionalidades mínimas de controle e identificação. O RTP e o RTCP foram desenhados para serem independentes das camadas de transporte e rede subjacentes”, tradução livre de Schulzrinne et al. (2003).

O Protocolo RTPC envia dados estatísticos em seus pacotes de controle, que são usados especificamente para auxiliar a mensurar a qualidade de serviço (QoS), feedback e sincronização entre os fluxos de dados de mídia. A largura de banda do tráfego RTCP é mínima comparada ao do RTP, em torno de 5%. Com isso a aplicação que utiliza o RTCP tem condições de mostrar e atuar de acordo com as estatísticas enviadas pelo protocolo, como no caso de alterar suas taxas de transmissão.

Os protocolos RTP E RTCP são usados em sistemas de comunicação e entretenimento envolvendo fluxo de dados de mídia, como telefonia, aplicativos de vídeo conferência e serviços de televisão.

3.1.4. SDP – Session Description Protocol

O SDP (Session Description Protocol) em sua tradução literal é protocolo de descrição de sessão, que é definido pela RFC 3264 como “um mecanismo para duas entidades chegar a uma visão comum de sessões multimídia entre as partes” (ROSENBERG et al., 2002). Neste modelo, um participante oferece ao outro uma descrição de sessão desejada a partir de sua perspectiva, e o outro participante responde com a sessão desejada de sua perspectiva. Este modelo de oferecer e responder é o mais utilizado em sessões em que a informação de ambos os participantes é necessária para completar uma vista da sessão. O SDP é o modelo utilizado pelo protocolo SIP para os participantes receberem suas capacidades, por exemplo, Codecs disponíveis.

Para evitar redundância e maximizar a compatibilidade com tecnologias estabelecidas, os métodos e protocolos de sinalização não são especificados pelos padrões WebRTC. Pode-se utilizar o protocolo SIP para sinalização por meio de WebSockets, chamado SipOverWebSocket. Neste modelo o SIP em conjunto com protocolo SDP negocia os parâmetros de mídia e sessão. Outras tecnologias também podem ser utilizadas para sinalização, tais como XMPP e JSEP por exemplo.

3.2. WebSockets

Segundo a RFC 6455 da IETF “a tecnologia WebSocket tem o objetivo de proporcionar um mecanismo para aplicações baseadas em navegadores que necessitam de comunicação de duas vias com servidores, e que não dependa de abertura de várias conexões HTTP” (FETTE et al., 2011). O WebScoket atinge seu objetivo realizando uma comunicação bidirecional *full-duplex* sobre um único soquete TCP.

Na padronização do WebRTC 1.0 adotou-se a Recomendação da API WebSocket W3C de 20 de setembro de 2012. A API permite que páginas de internet usem o protocolo WebSocket para comunicações de duas vias em um único túnel

sobre a internet. Além de prover redução de tráfico desnecessário de rede, ainda reduz a latência ne troca de informações se comparado a soluções que simulam uma conexão de duas vias. A característica de baixa latência é imprescindível para telefonia, como em casos para não causar um *delay* na chamada entre os participantes.

O handshake define entre cliente e servidor o estabelecimento da conexão WebSocket, passando uma lista de parâmetros (endereço, porta, segurança, etc.), protocolos e extensões a serem utilizados. A RFC 6455 descreve a conexão via WebSockets da seguinte maneira:

> "O protocolo WebSocket tem duas partes: o handshake e a transferência de dados.
>
> Uma vez o handshake realizado com sucesso, então a transferência de dados é iniciada. Este é um canal de comunicação de duas vias que cada lado pode, independentemente do outro, enviar dados à vontade", tradução livre de Schulzrinne et al. (2003).

O WebSocket pode realizar o controle de grandes volumes de chamadas devido a sua alta capacidade de conexões simultâneas no servidor, e sua fácil escalabilidade para ampliação da plataforma. Hoje essas características são imprescindíveis para qualquer tecnologia baseada em web e *Cloud Computing*.

3.3. JavaScript

Javascript é uma linguagem de programação direcionada para o desenvolvimento web. Implementada de maneira embutida nos navegadores para execução de códigos carregados na máquina do usuário, sem a necessidade de passar pelo servidor.

O WebRTC oferece sua solução para páginas de internet apenas por APIs em JavaScript, direcionando o desenvolvedor ter habilidades nesta linguagem de programação para uso da tecnologia.

3.4. JsSIP

O JsSIP é uma biblioteca em javascript atualizada com as últimas caraterísticas do protocolo SIP e APIs WebRTC para auxiliar no desenvolvimento de um endpoint de comunicação hospedado em um website. “Com o uso de JsSIP qualquer site da internet pode implementar comunicação em tempo real com poucas linhas de código” (MILLÁN; CASTILLO; CORRETGÉ, 2012).

As principais características do JsSIP são:

- Transporte do SIP sobre WebSocket.
- Chamadas de áudio/vídeo, mensagens instantâneas e notificação de presença.
- Compatibilidade com servidores SIP de código aberto, como Asterisk, Kamailio e OverSIP.

Com a utilização das APIs do JsSIP está embutido o uso do “*The* WebSocket *Protocol as a Transport for the Session Iniciation Protocol* (SIP)”, definido pela RFC 7118 IETF, atuando como um “SIP WebSocket *Client*”. O JsSIP adotou este protocolo pois o WebRTC não tem definido em seu padrão como é realizada a sinalização. Com essa adoção habilita-se a capacidade de abrir conexões via WebSocket com servidores para trocar sinalização usando sub-protocolos WebSocket SIP.

A premissa para uma arquitetura com a solução JsSIP é ter um SIP Server com suporte a WebSocket. Que serve para se registrar e encontrar a rota da chamada desejada. É necessário também utilizar o navegador Google Chrome na versão 24 ou superior para ter embutida a tecnologia WebRTC.

3.4.1. Mensagens e protocolos do *WebSocket as a Transport for SIP* (RFC7117)

Devido a não padronização da sinalização pelo WebRTC, ficou aberto para o desenvolvedor da solução escolher como é realizada a sinalização. A solução de sinalização utilizada pelo JsSIP foi a aplicação do protocolo SIP sobre WebSocket (RFC6455). Esse protocolo permite a troca de mensagens entre cliente e servidor por uma conexão persistente do tipo TCP, opcionalmente sob uma camada TLS

(RFC5246). O protocolo de *handshake* inicial faz uso da semântica do HTTP (RFC2616), permitindo o reuso a infraestrutura existente HTTP, e o *upgrade* de conexão do HTTP para o WebSocket. Os Navegadores modernos incluem um cliente WebSocket compilado em seu código fonte com a API WebSocket conforme especificado pela W3C, permitindo o uso do SIP sob WebSockets, chamado *WebSocket SIP Subprotocol*.

Cada mensagem SIP deve ser carregada em uma única mensagem WebSocket, e nenhuma mensagem WebSocket pode conter mais de uma Mensagem SIP. Por meio dos cabeçalhos nas mensagens SIP é carregado um identificador do protocolo de transporte, neste caso o ***ws*** ou ***wss*** deve ser usado para requisições sobre as conexões WebSocket. Desta maneira toda a sinalização SIP é transportada via WebSocket. Por questão de segurança, desde a versão 47 do Google Chrome é necessário que o servidor tenha um certificado SSL/TLS para o uso da API getUserMedia do WebRTC.

A Figura 4 ilustra o *handshake* entre um cliente (Alice) e servidor (SIP *proxy*). Neste fluxo Alice acessa uma página *web* usando um navegador que executa o código JavaScript com o *WebSocket SIP Subprotocol*, como o JsSIP por exemplo. O código JavaScript (JsSIP) estabelece uma conexão de WebSocket segura com o SIP *proxy* de endereço proxy.example.com. Uma vez a conexão WebScoket estabelecida, Alice envia uma mensagem de requisição do tipo SIP REGISTER. Por fim o servidor confirma que a conexão foi realizada.

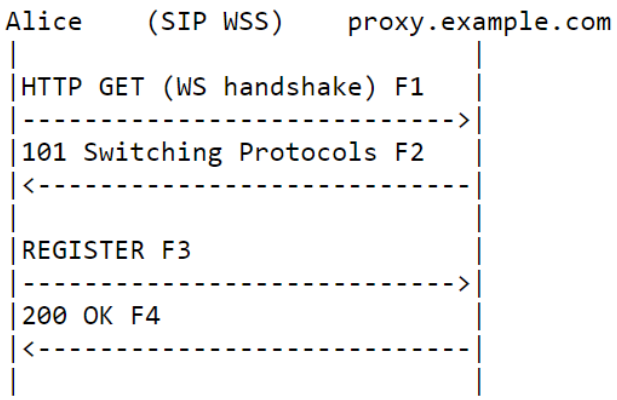


Figura 4. Exemplo de handshake e registro SIP. Fonte: RFC 6455.

Abaixo segue o conteúdo das mensagens trocadas para realização do fluxo da Figura 4:

```
F1 HTTP GET (WS handshake)  Alice -> proxy.example.com (TLS)
   GET / HTTP/1.1
   Host: proxy.example.com
   Upgrade: websocket
   Connection: Upgrade
   Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
   Origin: https://www.example.com
   Sec-WebSocket-Protocol: sip
   Sec-WebSocket-Version: 13
F2 101 Switching Protocols  proxy.example.com -> Alice (TLS)
   HTTP/1.1 101 Switching Protocols
   Upgrade: websocket
   Connection: Upgrade
   Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
   Sec-WebSocket-Protocol: sip
F3 REGISTER  Alice -> proxy.example.com (transport WSS)
   REGISTER sip:proxy.example.com SIP/2.0
   Via: SIP/2.0/WSS df7jal23ls0d.invalid;branch=z9hG4bKasudf
   From: sip:alice@example.com;tag=65bnmj.34asd
   To: sip:alice@example.com
   Call-ID: aiuy7k9njasd
   CSeq: 1 REGISTER
   Max-Forwards: 70
   Supported: path, outbound, gruu
   Contact: <sip:alice@df7jal23ls0d.invalid;transport=ws>
     ;reg-id=1
     ;+sip.instance="<urn:uuid:f81-7dec-14a06cf1>"
F4 200 OK  proxy.example.com -> Alice (transport WSS)
   SIP/2.0 200 OK
   Via: SIP/2.0/WSS df7jal23ls0d.invalid;branch=z9hG4bKasudf
   From: sip:alice@example.com;tag=65bnmj.34asd
```

```
To: sip:alice@example.com;tag=12isjljn8
Call-ID: aiuy7k9njasd
CSeq: 1 REGISTER
Supported: outbound, gruu
Contact: <sip:alice@df7jal23ls0d.invalid;transport=ws>
  ;reg-id=1
  ;+sip.instance="<urn:uuid:f81-7dec-14a06cf1>"
  ;pub-gruu="sip:alice@example.com;gr=urn:uuid:f81-7dec-
14a06cf1"
  ;temp-gruu="sip:87ash54=3dd.98a@example.com;gr"
  ;expires=3600
```

Com o registro de Alice efetuado no servidor, Alice pode enviar uma mensagem SIP do tipo INVITE (convite para uma chamada) para outro usuário. Dado que exista outro usuário Bob registrado e acessível para o servidor, o fluxo da Figura 5 ilustra uma ligação entre Alice e Bob.

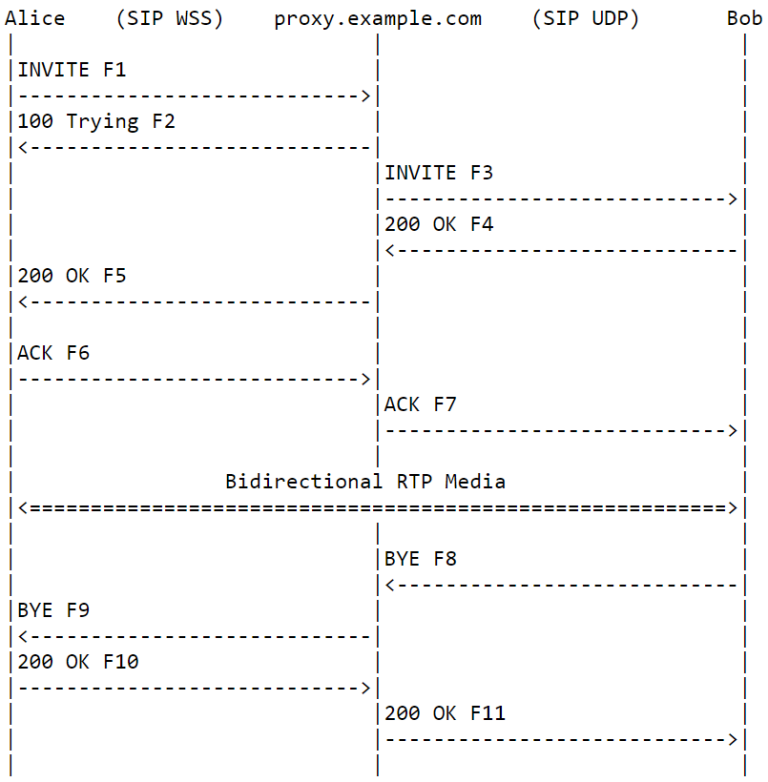


Figura 5. Fluxo de mensagens SIP entre Alice e Bob. Fonte: RFC 6455.

Na ordem das mensagens da Figura 5:

- F1: Alice envia INVITE para Bob pelo servidor proxy.example.com. Nesta etapa, no cabeçalho SIP existe o campo SDP que contém os parâmetros de mídia para negociar o acordo. Por exemplo, quais Codecs de áudio são aceitos.
- F2: Servidor retorna o status 100 Trying que significa que recebeu o INVITE e está prosseguindo com a solicitação.
- F3: O servidor encaminha o INVTE para bob.
- F4: Bob responde 200 OK aceitando o INVITE. Neste mesmo pacote existe os parâmetros de mídia de Bob, que será utilizado para negociar com Alice.
- F5: Alice recebe o repasse do 200 OK de Bob para o servidor.
- F6 e F7: Nestes passos é enviado o ACK de Alice confirmando o recebimento das informações d Bob. A partir daí, entra o protocolo RTP e RTCP trocando os pacotes de voz entre Alice e Bob com o áudio propriamente dito, e a estatística de controle destes pacotes.
- F8: Bob encerra a chamada com a mensagem BYE
- F9: Servidor repassa para Alice a sinalização de fim de chamada vinda de Bob.
- F10 e F1 respectivamente são referentes a confirmação do recebimento da mensagem SIP (BYE) por Alice. Finalizando a chamada.

Os fluxos, mensagens e ilustrações foram retiradas da própria documentação da RFC 7118 descrita neste capítulo.

3.4.2. Código JsSIP – *Client* WebSocket SIP

O *JsSIP User Agent* é o elemento principal do JsSIP. Ele representa o cliente SIP associado a uma conta SIP. O *JsSIP User Agent* é definido em uma classe JsSIP.UA (MILLÁN; CASTILLO; CORRETGÉ, 2012).

Para criar um JsSIP User Agent o código abaixo cria um objeto de configuração para sua inicialização. Neste objeto existem parâmetros de configuração mandatórios e existem alguns opcionais, que para simplificar não serão necessários.

```
var socket = new JsSIP.WebSocketInterface('wss://sip.myhost.com');
var configuration = {
  sockets  : [ socket ],
  uri      : 'sip:alice@example.com', //usuário para se registrar no SIP server
  password : 'superpassword' // senha do usuário };
```

O código JavaScript abaixo descreve como utilizar o JsSIP para criar uma instância de um comunicador no navegador Google Chrome e realizar uma chamada SIP:

```
// Cria a instância JsSIP e a executa:
var coolPhone = new JsSIP.UA(configuration);
coolPhone.start();
// Realizar uma chamada de áudio/video:
var useAudio = true;
var useVideo = true;
var views = {
  'selfView':   document.getElementById('my-video'),
  'remoteView': document.getElementById('peer-video')
};
var eventHandlers = {
  'connecting': function(e){ // Seu código aqui },
  'progress':   function(e){ // Seu código aqui },
  'failed':     function(e){ // Seu código aqui },
  'started':    function(e){ // Seu código aqui },
  'ended':      function(e){ // Seu código aqui }
};
coolPhone.call('sip:bob@ ExemploServidorSIP.com, useAudio,
useVideo, eventHandlers, views);
```

A Velip é o serviço de contact center em nuvem utilizado como ferramenta no estudo de caso deste trabalho. O JsSIP é a solução SIP *Client* utilizada para realizar chamadas VoIP pela Velip. Todos os eventos e métodos de instanciação do JsSIP podem ser estudados em sua documentação na internet no site http://jssip.net/documentation/.

3.5. Qualidade – Codecs e outros aspectos

A qualidade de uma chamada VoIP é diretamente relacionada ao Codec utilizado e a qualidade de conexão entre os participantes da chamada. O Codec tem o papel de codificar e decodificar a voz que será enviada por meio de pacotes RTP pela rede. E com a ajuda do RTCP é possível analisar estatisticamente o comportamento dos pacotes RTP e prover informações suficientes para gerar análise de qualidade de serviço (QoS). Com isso é possível que a aplicação (*software*) atue, por exemplo alterando a taxa de bits do Codec de áudio de acordo com a largura de banda disponível durante a conversação.

3.6. Codecs

Um codec é um algoritmo ou dispositivo de codificação de sinais. Nessa codificação o sinal pode ser comprimido, criptografado, armazenado e posteriormente realizar as operações reversas para reproduzir o sinal. Codecs são usados em *streaming* de dados de videoconferências e ligações telefônicas por exemplo.

Pode-se dividir os codecs em dois tipos, com perda e sem perda. Os codecs sem perdas codificam um sinal para atingir uma determinada compressão de dados, e garantem que ao reconstruir o sinal ele seja igual ao original. Esse tipo de codec normalmente gera arquivos com baixa taxas de compressão em relação aos codecs com perdas. Alguns exemplos de codecs sem perdas são MPEG e H.264 para vídeo e PNG e TIFF para imagens.

Já os codecs com perdas comprimem muito mais o tamanho dos arquivos, porém existe uma perda de qualidade de informação na decodificação do arquivo. Dependendo da taxa de compressão dos dados, alguns codecs chegam a proporcionar diferenças imperceptíveis para a análise de um humano, ao comparar a diferença entre o sinal original e o codificado. Alguns exemplos de codecs com perdas são OGG, MP3, AC3 e WMA para áudio, Xvid, DivX, WMV, RMVB para vídeo, além de JPEG, JPEG 200 e GIF em imagens.

Todos esses conceitos refletem diretamente na taxa de transferência de dados, assim deve-se escolher a que melhor se ajusta de acordo com o projeto desenhado. É abordada de maneira mais profunda neste trabalho o VoIP, e por isso são trazidos alguns Codecs de áudio para familiarização.

- O G.711 é um Codec padrão de áudio ITU-T (1988a) usado para telefonia que utiliza uma modulação PCM (Pulse code modulation) para frequências de voz. O G.711 é gerado a uma taxa de 64 kbit/s por sinais de áudio captados na faixa dos 300Hz a 3400 Hz de banda.
- O codec G.722 WideBand é um codec que suporta áudios com amostragens de 48, 56 e 64kbits/s definidos pelo padrão ITU-T (1988b). Em comparação com codecs de banda estreita como o G.711, o G.722 oferece superior qualidade de áudio. Esse padrão provê qualidade de fala nas bandas entre 50Hz e 7000Hz.
- O Opus é um codec de código-fonte aberto e livre para uso. O diferencial deste codec é sua alta versatilidade, pois é possível alterar seu *bitrate* dinamicamente durante uma chamada. No VoIP o *bitrate* pode variar devido a degradação do sinal da rede. Padronizado também pela Internet Engineering Task Force sob a RFC 6716 proposta por Valin et al. (2012), a taxa de *bitrate* do Opus varia de 6kb/s à 510kb/s com amostrar de 8kHz até 48Khz. É a primeira opção da lista de codecs compatíveis do Google Chrome.
- iSac é um codec desenvolvido pela Global IP Solutions (agora parte do Google) utilizado para aplicações VoIP. É um codec adaptativo que opera com *delay* curto e que é altamente recomendado para comunicação em tempo real. Foi desenvolvido especialmente para aplicações de baixas taxas de *bitrate*. A compressão é feita em quadros de 16Khz com amostras em 16 bits com cada

quadro contendo 30 ou 60ms de fala. Sua definição é dada pelo esboço na IETF escrita por Le Grand et al. (2013).

3.7. WebRTC

O WebRTC permite realizar a comunicação multimídia em tempo real pelo navegador de internet. Mídias como áudio, vídeo, chat e quaisquer outros dados podem ser trocados entre computadores ou outros dispositivos com *browser* compatível. Isso tudo por meio de um *gateway* de interconexão para estabelecer a comunicação da conversa ponto a ponto. Com o WebRTC todas essas funcionalidades são possíveis sem a necessidade de instalar um software ou plugin adicional. Um resumo de Bergkvist (2018) descreve que o ambiente WebRTC inclui os blocos fundamentais para permitir o envio e recebimento de mídia por navegadores ou dispositivos, implementando o conjunto apropriado de protocolos em tempo real.

Para o desenvolvedor o WebRTC é uma coleção de protocolos e APIs de código aberto para comunicação de mídias em tempo real pela *Web*. Essas APIs em JavaScript possibilitam programação e execução do WebRTC nos principais navegadores de internet. O WebRTC é um padrão aberto e descrito pelos consórcios World Wide Web Consortium (W3C), que descreve a parte das *APIs*, e o Internet EngineeringTask Force (IETF) que implementa seus protocolos. Os resultados são as APIs para desenvolvimento web de soluções utilizando linguagens como JavaScript e HTML5. O WebRTC é candidato a recomendação na W3C para comunicações em tempo real entre navegadores desde setembro de 2018. Sua RFC 7478 de categoria informativa descreve casos de uso e requisitos (HOLMBERG, 2015).

O WebRTC é de uso livre, e tem uma comunidade de desenvolvedores que sustentam sua constante evolução. A facilidade e o poder da API para obtenção dos recursos de mídia dos dispositivos também são aliados desta tecnologia. O WebRTC está sendo estudado e utilizado por grandes empreses de *softwares* e telecomunicações.

Nos seguintes subcapítulos são descritas algumas vantagens e desvantagens da escolha do WebRTC, como ele funciona, compatibilidade com navegadores,

descrições de alguns detalhes da sinalização e conexão, e protocolos que o WebRTC utiliza como recomendação.

3.7.1. Vantagens do WebRTC

O WebRTC tornou mais simples o desenvolvimento de aplicações web de áudio e/ou vídeo. A principal diferença entre o WebRTC e tecnologias anteriores é o acesso aos dispositivos de hardware, como microfones e câmeras, sem necessidade de instalação de plug-ins ou instalação prévia de algum software.

Os principais benefícios do WebRTC que a GSM Association exalta em seu *white-paper* (LAJTOS; O'BYRNE; ERSOZ, 2016):

> 1) Redução de custos: os custos de licença e integração para soluções baseadas em WebRTC são muito mais baratas do que qualquer solução disponível no mercado.
>
> 2) Facilidade de uso: o WebRTC pode ser usado para entregar chamadas de áudio e vídeo em uma grande gama de navegadores ou dispositivos sem a necessidade de download e instalação de nenhum *client*.
>
> 3) Segurança: toda comunicação utilizando o WebRTC é totalmente criptografada entre os usuários finais, disponibilizando segurança de informação fim a fim nas chamadas para as empresas e seus clientes.

Em seu white-paper, a Twilio (2016) relata cases de sucesso do uso do WebRTC para chamadas online de qualidade. Uma das empresas, a Zendesk, trocou o *plugin* do Flash que utilizavam para realização de chamadas por uma combinação de WebRTC com Twilio. O resultado foi uma grande melhoria na qualidade das chamadas. Comparado aos outros canais como chat e email, o atendimento via chamadas de voz por WebRTC é o que tem melhor índice de satisfação do cliente, próximo de 93%.

O WebRTC é utilizado por meio da linguagem JavaScript numa página HTML5. Essa página pode ser hospedada em plataformas em nuvem como AWS da Amazon, Google Cloud Computing e Microsoft Azure.

3.7.2. Framework WebRTC

O desenvolvimento de uma aplicação que emprega o WebRTC é baseado em APIs JavaScript. As APIs têm particularidades de funcionamento, regras de utilização e respostas em suas execuções. O conjunto de APIs são embutidas dentro do código fonte dos navegadores, que por sua vez são instalados nos dispositivos. Para descrever os detalhes de como tudo isso tudo funciona, neste capítulo, foi utilizado como referência o artigo "*Getting Started with WebRTC*" (DUTTON, 2014).

Na ilustração da Figura 6, é mostrado o diagrama geral da arquitetura do WebRTC. Nessa arquitetura temos duas camadas distintas, a camada do navegador que contém as suas capacidades e as APIs do WebRTC embutidas, e a camada de desenvolvedor *web* que consome essas APIs e recursos computacionais por códigos em JavaScript. A camada do navegador contempla algumas de suas capacidades como, os motores de Voz e Vídeo (Codecs, canceladores de eco e ruído, *buffer*, etc.), e os mecanismos de transporte de pacotes (P2P, SRTP, etc.).

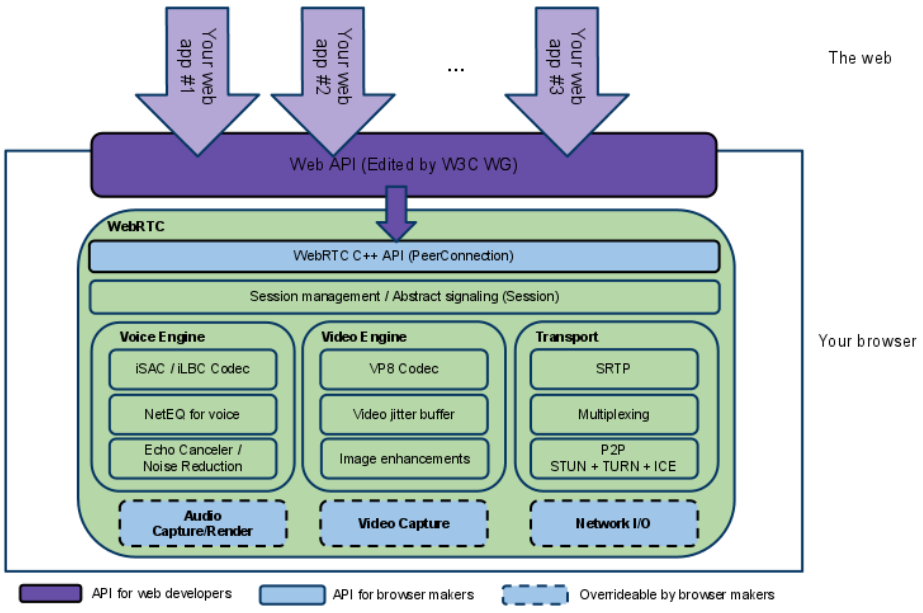


Figura 6. Visão geral da arquitetura WebRTC (DUTTON, 2014).

Dentro das camadas ilustradas na Figura 6, tem-se o detalhamento do funcionamento de cada habilidade do *framework*.

Your Web App - Aplicação de terceiros baseada em desenvolvimento web (JavaScript), como chats de vídeo e áudio capacitado pela API Web para comunicação em tempo real.

Web API - Uma API para ser usada por desenvolvedores terceiros para desenvolvimento web baseado em aplicativos.

WebRTC Native C++ API - Uma camada API que permite navegadores fazerem facilmente a proposta de implementação da API Web.

Transporte / Sessões - Os componentes de sessão são construídos pela reutilização de componentes do libjingle, sem usar ou requerer o protocolo xmpp / jingle.

- **RTP *Stack***: Uma rede de pilhas para RTP
- **STUN/ICE**: Um componente que permite que as chamadas usem os mecanismos de *STUN* e *ICE* para estabelecer conexões por vários tipos de rede.
- **NAT**: O Network Address Translation serve para realizar a tradução dos IPs expostos à internet, tendo em vista os roteadores e firewalls que intermediam os dispositivos alvo.
- **Session Management**: Uma camada de sessão abstrata que permite a configuração de chamada e da camada de gerenciamento. Esta deixa a implementação do protocolo para o desenvolvedor do aplicativo.

VoiceEngine - *Framework* para a fluxo de mídia de áudio, da placa de som para a rede.

A maioria das aplicações desenvolvida pelos programadores em WebRTC precisam realizar as seguintes operações:

- Obter informações de áudio, vídeo ou qualquer outro tipo de dado a ser trocado.
- Obter informações sobre a rede, como endereços IPs e portas. E trocar essas informações com outro cliente WebRTC para habilitar a conexão.
- Coordenar a comunicação da sinalização para reportar erros, e iniciar e fechar sessões entre as pontas.
- Trocar informação de mídia e capacidades entre aplicações, como resolução e codecs.
- Comunicar a transmissão de arquivos de mídia ou dados.

E para adquirir e transmitir esses dados, o WebRTC implementa as **APIs**:

- **MediaStream**: Obtém acesso a streams de dados, por exemplo a câmera e microfone do usuário
- **RTCPeerConnection**: Chamada de áudio e/ou vídeo, com facilidades para criptografia e gerenciamento de banda
- **RTCDataChannel**: Comunicação genérica de dados de ponta a ponta.

Esses três métodos são os pilares para o funcionamento de uma aplicação baseada em WebRTC. Essas APIs são as principais funções que realizam todo o trabalho mais complexo como o acesso aos recursos de mídia dos dispositivos, conexão entre os participantes e mantém um canal de comunicação aberto para troca de dados entre cliente e agente.

3.7.3. APIs WebRTC – MediaStream, RTCPeerConnection e RTCDataChannel

3.7.3.1. MediaStream

Essa API representa a sincronização dos streams de mídia. Para utilizar essa API é utilizado como chamada a função navigator.getUserMedia(). E pode ser obtido

como saída um elemento de vídeo ou um RTCPeerConnection por exemplo. Essa saída pode obter três parâmetros, respectivamente, um objeto com restrições, uma chamada de volta com sucesso ou uma chamada com falha.

Cada MidiaStream tem uma etiqueta com uma sequência de códigos do tipo “Xk7EuLhsuHKbnjLWkW4yYGNJJ8ONsgwHBvLQ”, que são caracteres que formam combinações difíceis de serem repetidas, com a função de identificador o *stream*.

3.7.3.2. RTCPeerConnection *e* Sinalização: Controle de sessão, rede e informação de mídia

No WebRTC a sinalização não é realizada automaticamente pela API RTCPeerConnection. Porém o processo de sinalização é necessário para identificação das duas pontas da chamada, trocando descrições de sessão para configurar as portas de mídia, e ajuda no compartilhamento de tudo que é utilizado no *handshake* (protocolo inicial de conexão). O WebRTC não especifica em seu escopo um ou qual o protocolo de sinalização deve ser utilizado. Com isso, parte do desenvolvedor escolher o protocolo para sinalização.

Por não definir a sinalização, o WebRTC aceita qualquer mecanismo de sinalização, tais como SIP, XMPP ou qualquer outro que faça um canal de comunicação de duas vias (*duplex* / *two way*). E para maximizar a interoperabilidade com sistemas existentes ele não especifica por padrão nenhuma autenticação, caso o seja necessário o WebRTC pode ser conectado em um mecanismo existente via API.

Por meio de protocolo SIP via WebSockets por exemplo é possível integrar o WebRTC com plataformas de telefonia como Asterisk, FreeSWITCH, Kamailio e OverSIP. Com essa integração é possível realizar o gerenciamento de chamadas com tecnologias legado de telefonia. Inclusive algumas que já operam atualmente de maneira adaptada com o protocolo SIP. Essa possibilidade aumenta o leque de oportunidade do WebRTC.

O WebRTC usa o **RTCPeerConnection** para realizar o streaming de dados de comunicação entre os navegadores (entre as pontas), mas também é necessário um

mecanismo para coordenar a comunicação e enviar mensagens de controle, pelo processo conhecido como sinalização.

A sinalização é usada para trocar três tipos de informação:

- **Mensagens de controle de sessão**: Para inicializar ou fechar a comunicação e reportar erros
- **Configuração de Rede**: Para se comunicar com o mundo pelo seu endereço IP e porta.
- **Capacidades de mídia**: Quais codecs e resoluções podem ser suportadas pelo meu navegador e qual queremos utilizar.

Para haver troca de informação de mídia é necessário primeiramente estabelecer com sucesso a troca de sinalização. O método para criar a sinalização é o **RTCPeerConnection**.*createSignalChannel*().

Os participantes da conversa podem precisar verificar e trocar remotamente informações de mídia de áudio e vídeo, como resolução e capacidades de codecs. A sinalização para negociar a configuração da informação da mídia é feita usando o SDP (Session Description Protocol), e seu método é o **RTCPeerConnection**.*createOffer*(). Para definir as informações de mídia local é chamado o **RTCPeerConnection**.*setLocalDescription()* e a informação remota como **RTCPeerConnection**.*setRemoteDescriptionI()*. As descrições são enviadas pelo método **RTCPeerConnection**.*createAnser().*

Uma vez que o processo de sinalização está completo, os dados de streaming de mídia podem ser trocados diretamente entre os participantes.

Para ilustrar melhor o uso das APIs do WebRTC, foi disponibilizado no **Anexo A** deste trabalho um fluxo fictício entre Alice e Bob. Nesta ilustração é possível verificar o instante de cada evolução da sinalização entre os participantes. No estudo de caso

deste trabalho é utilizado SIP via WebSocket para realizar a sinalização. Essa sinalização é detalhada no Capítulo 3.4.1.

3.7.3.3. RTCDataChannel

A API RTCDataChannel permite troca de dados arbitrários entre as pontas com baixa latência e alta taxa de transferência. Esta característica pode ser utilizada em jogos, acesso remoto ao computador, chats e transferência de arquivos. A troca de informação pode ocorrer diretamente entre navegadores, tornando esta aplicação mais rápida do que qualquer outra que seja necessário passar por um servidor, firewall, NATs, etc. Esta API é interessante pois pode deixar a aplicação livre para qualquer tipo de implementação sem restrições, dando ao programador a liberdade de criar de acordo com sua imaginação.

3.7.3.4. ICE, TURN / STUN

No mundo real da internet, o WebRTC precisa de servidores intermediários, mesmo que simples, para que as seguintes atividades aconteçam:

- Descoberta de usuários e comunicação: Os usuários descubram uns aos outros e troquem detalhes reais, como seus nomes por exemplo.
- Sinalização: As aplicações *clients* WebRTC troquem informação de rede, como endereços IP e portas, formatos de mídia e resolução
- Aplicações *clients* WebRTC atravessem gateways NAT (Network address Translatioin) e firewalls.
- Servidores de retransmissão em caso de falha na comunicação ponto a ponto.

O protocolo STUN e sua extensão TURN são usados pelo *framework* ICE para habilitar que o RTCPeerConnection lide com NAT e outros caprichos da rede.

ICE (Interactive Connectivity Estabilishmente) é um *framework* para conectar pares, como em conversas de áudio e vídeo entre duas pessoas. Inicialmente o ICE

tenta encontrar o melhor caminho para conectar os pares diretamente, com a menor latência possível via protocolo UDP KERANEN; HOLMBERG; ROSENBERG (2018). Alternativamente tenta também via TCP. No processo de STUN, os servidores têm uma única tarefa de habilitar que os pares atrás de um NAT encontrem seus endereços e portas públicos. O TURN é responsável pela retransmissão do tráfego se a conexão direta entre pares falhar.

Com estas introduções breves do funcionamento da sinalização, conexão, arquitetura e desenvolvimento do WebRTC, é possível ilustrar um possível exemplo de fluxo de operação em um ambiente de contact center via ilustração da Figura 7.

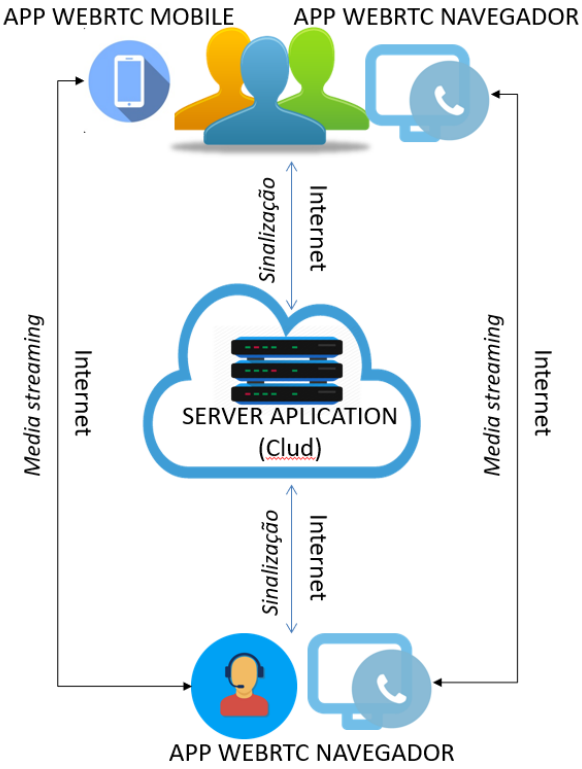


Figura 7. Fluxo de operação de um *Contact Center* simplificado com WebRTC.

3.7.4. Segurança

O WebRTC nasceu com a premissa de ter **criptografia** por padrão em sua tecnologia, pois atualmente não é mais um item opcional em uma chamada privada.

Além da criptografia, existem outros mecanismos de segurança que foram projetados para funcionar entre WebRTC e navegadores de acordo com o tipo de mídia transmitido.

A criptografia de mídia e comunicação é padrão no WebRTC, e essa questão é forçada em todos os componentes da tecnologia, incluindo os protocolos definidos. O Protocolo de criptografia utilizado depende do tipo de canal, como por exemplo o *stream* de dados usa *Datagram Transport Layer Security* (**DTLS**) e o *stream* de mídia é criptografado pelo protocolo *Secure Real-time Transport Protocol* (**SRTP**).

Outra premissa de segurança no WebRTC é que o site com conteúdo das APIs WebRTC seja acessado via HTTPS, ou seja, tenha um **certificado de segurança/autenticidade**. Caso contrário não é possível realizar nenhum tipo de transferência de dados pela API getUserMedia. Além disso ao acessar um site sem certificado, o navegador não carrega a página e alerta que o site não é confiável. Para continuar em uma página não segura, o usuário deve assumir o risco configurando o navegador para incluir o site em uma lista manual de sites confiáveis. Em versões mais recentes de alguns navegadores, caso o site não tenha certificado de segurança, então o navegador não habilita os dispositivos de hardware multimídia como câmera e microfone. A boa notícia é que em 2018 a compra de um certificado de segurança/autenticidade já é de fácil acesso. Algumas plataformas de *cloud* já disponibilizam certificados sem custo adicional ao utilizar seus serviços, como o Google Cloud Computing por exemplo.

Para ter acesso aos recursos de mídia como Microfone e Webcam, ao executar uma função de MidiaStream do WebRTC, necessariamente o navegador **pede ao usuário que permita habilitar** esse dispositivo de entrada por meio de um *popup*. Assim alertando ao usuário de que esse tipo de informação está sendo transmitida via navegador, e dando a oportunidade para que tome uma decisão.

3.8. Premissas

O WebRTC é suportado nativamente por grande parte dos principais navegadores de internet existentes como o Google Chrome, Mozilla Firefox, Opera,

Canary, Nightly, Browser (Samsung), Microsoft Edge e tem suporte para algumas funcionalidades no Safari (Apple). A Figura 8 gerada pelo site *iswebrtcreadyyet.com* ilustra o atual cenário de suporte dos navegadores ao WebRTC. Na figura o quadrado vermelho indica que não há suporte ao WebRTC, o quadrado amarelo indica algum suporte e o quadrado verde o suporte completo a todas as funcionalidades do WebRTC.

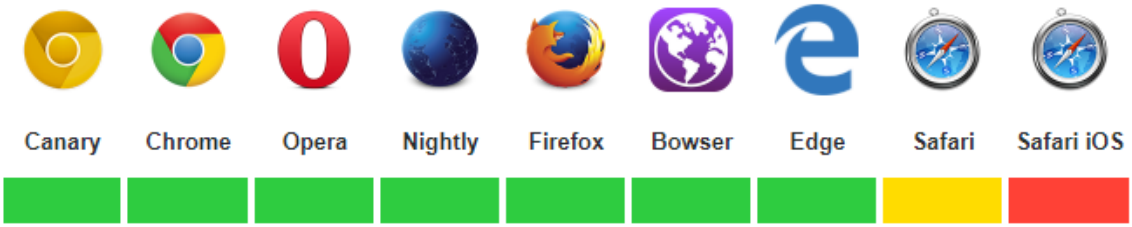


Figura 8. Suporte detalhado dos navegadores ao WebRTC. Fonte: iswebrtcreadyyet.com (2018).

Ou seja, tanto operador quanto cliente devem utilizar algum dos navegadores citados previamente para obter toda funcionalidade de comunicação multimídia oferecido pelo WebRTC usando um *browser*. Esses navegadores têm nativamente em seu código fonte as instruções necessárias para interpretar as requisições das APIs do WebRTC, além de alguns padrões de *Codecs* de áudio e vídeo.

Juntando os líderes no mercado de navegadores como Chrome, Safari e Firefox já se engloba cerca de 80% dos usuários de navegadores do mundo segundo o relatório da StatCounter de setembro de 2018 disponibilizado em seu site http://gs.statcounter.com/browser-market-share/desktop/worldwide/#monthly-200901-201807.

4. Soluções que empregam o WebRTC

O WebRTC já é utilizado em vários aplicativos de comunicação em tempo real. Por exemplo o Google Hangouts e Facebook Messenger utilizam a tecnologia em suas chamadas de áudio e videoconferência TSAHI (2017). O maior objetivo deste

capítulo é mostrar ideias “fora da caixa” e soluções de *contact center*. A tecnologia WebRTC já vem sendo estudada pelas grandes empresas de *contact center*, porém apenas para soluções de áudio e vídeo. O WebRTC tem recursos para fazer muito mais de só isso.

4.1. Oracle WSC – WebRTC Session Controller

A Oracle com seu produto **WSC** disponibiliza um SDK para desenvolvimento de aplicativos mobile Android, iOS e web (API JavaScript). Nesse SDK está embutido as APIs do WebRTC. Com isso é possível trazer para dentro de uma aplicação de celular todas as possibilidades multimídias que o WebRTC provê.

Um caso de uso em que a Oracle mostra em suas apresentações presenciais é a solução de uma seguradora. O cliente segurado faz a abertura de um sinistro pelo aplicativo. As informações são enviadas por meio da tecnologia WebRTC, como imagens do acidente a informação da localização por GPS que o celular provê. O cliente também pode aproveitar o sinistro aberto para ligar para a central de atendimento via aplicativo, pulando a URA ao enviar as informações de identificação e do sinistro via rede de dados. A chamada também ocorre pela rede de dados (Wifi ou 4g).

4.2. IoT – Drones e sistemas de segurança

Com os recursos do WebRTC um drone preparado pode fazer o *streaming online* de suas imagens por meio de conexões 3g ou 4g, bem como receber comandos à distância. O mesmo conceito pode ser aproveitado para circuitos de monitoração em que não há rede cabeada, como fibra óptica e coaxial. Os próprios dispositivos podem fazer o envio das informações e imagens diretamente para um servidor online, ou o dispositivo centralizador de informações realizar este processo. Atualmente em prática no mercado não existem muitas soluções IoT que utilizam os benefícios do WebRTC.

4.3. Discord

Discord é uma rede social para gamers. Nesta plataforma os usuários de jogos como Minecraft, Fortnite e Pugb conversam durante os jogos por chats e salas de áudio utilizando a tecnologia WebRTC como base.

O Discord compartilhou alguns números atingidos em 2.5 anos de vida, como 14 milhões de usuários ativos diariamente. Esses usuários trocaram 9.5 bilhões de mensagens, e chegaram a um pico de 2.5 milhões de usuários de voz simultaneamente.

O Discord está disponível para todos os navegadores compatíveis com WebRTC, em softwares instaláveis para Windows, MacOS e Linux, e aplicativos móveis para iOS e Android VASS (2018).

4.4. Amazon's Mayday

A Amazon desenvolveu uma solução de suporte para seus Tablets Kindle fire utilizando a tecnologia WebRTC. A solução é um botão de ajuda que conecta o cliente diretamente com o suporte da Amazon para solucionar duvidas e problemas. A solução além de realizar uma conferência em tempo real, ela também compartilha a tela do usuário com o atendente. O atendente ainda consegue mostrar informações na tela para o usuário, como piscar qual botão o usuário de tocar ou quaisquer outras instruções de maneira visual. Esta solução é uma das soluções que mais contempla o uso bidirecional da troca de informação entre os pares.

4.5. Soluções WebRTC de *Contact Center* em nuvem

Tendo em vista a aplicação dos conceitos abordados, neste capítulo são mostradas soluções WebRTC no cenário de centrais atendimento com infraestrutura em nuvem. O intuito é mostrar aplicações comerciais que utilizam o WebRTC para vender serviços de *Contact Center* na prática.

Para embasamento de uso da tecnologia, foram pesquisadas três empresas que oferecem tais serviços. As empresas escolhidas foram a nacional Velip, a Genesys que é a empresa líder no “Quadrante Mágico” do Gartner, e a Zendesk que faz tem uma plataforma para gerenciamento de *tickets* para suporte.

4.5.1. Velip

A VELIP é uma empresa de tecnologia que oferece soluções de comunicação em nuvem. A empresa oferece serviços desde envio de mensagens de voz, passando por automação do processo de atendimento, *Call Center* em nuvem, Inteligência artificial, entre outros.

Com o produto de Contact Center na nuvem, a Velip disponibiliza uma Plataforma de *Call Center* em nuvem com *frontend* Web. Esta página web utiliza tecnologia WebRTC. Com isso permite montar um *Call Center* sem investir na compra de PABX, linhas telefônicas e aparelhos. Toda programação e configuração da plataforma é feita via Web. Nesta plataforma estão inclusos:

- Discador inteligente: de acordo com mailing enviado, realiza chamadas e as entrega para os operadores quando a ligação é completada.
- URAs: As Unidades de Resposta Audível fazem as boas vindas da central de atendimento, e podem realizar o direcionamento para o serviço que o cliente deseja ser atendido.
- Distribuidor automático de chamada: Pode distribuir as chamadas recebidas de forma igualitária entre os operadores, ou dando preferência para um que tenha maior habilidade em determinado serviço.
- Gravação disponibilizada pela Web: Possibilidade de escuta e download das chamadas realizadas entre clientes e operadores.
- Torpedos de Voz: Tem a facilidade de enviar torpedos de voz para enviar recados aos clientes cadastrados.
- Supervisão online: Pelo navegador ou aplicativo, é possível realizar a supervisão do atendimento da central e suas estatísticas. Tendo maior gestão sobre o andamento da operação.

- Segurança: Sistema redundante, relatórios online e acompanhamento minuto a minuto. Além de ser integrado aos PROCONS.

Em seu site, a Velip descreve de maneira simples o funcionamento básico do processo:

> “O administrador do *call center* tem acesso a um portal web onde faz as programações das campanhas, filas e atendentes, e acompanha em relatórios online o andamento das campanhas e rendimento dos atendentes. Do outro lado, para o operador basta apenas entrar em um browser onde receberá as ligações e todas as informações do cliente. Isto lhe permite até trabalhar remotamente em um *home office*.”

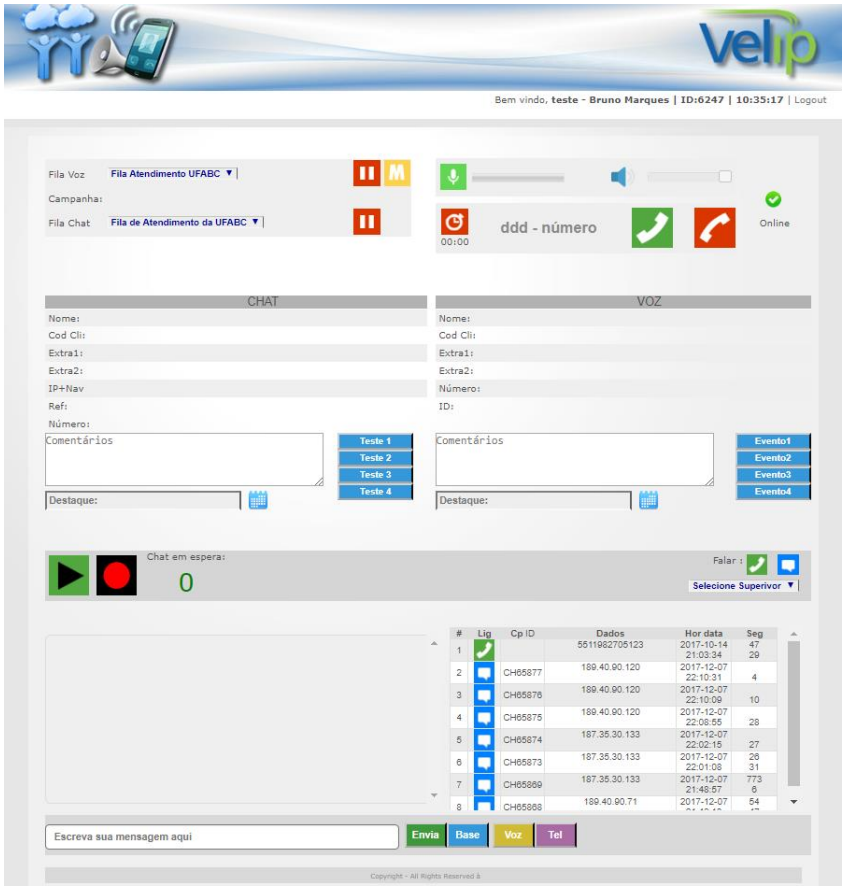


Figura 9. Posição de atendimento virtual Velip - Tela do operador.

A Figura 9 ilustra a tela de trabalho do operador. É possível visualizar as filas de voz e chat que o operador está disponível, bem como as informações de seus últimos atendimentos.

4.5.2. Genesys

Fundada no ano de 1990 e com sede em Daly City na Califórnia, a Genesys é uma empresa que desenvolve tecnologia para *Contact Center* e gestão de experiencia do cliente para empresas de médio e grande porte. Os softwares vendidos podem ser baseados com computação na nuvem ou infraestrutura local. Para se sobressair no mercado a empresa realizou a compra de várias empresas do ramo para adquirir suas tecnologias. Como a Utopy provedora de análise de voz, a Angel desenvolvedora de URA baseada em *cloud*, e a SoundBite empresa de cobrança e pagamentos, marketing e software de atendimento ao cliente baseado em computação na nuvem. A Genesys pertence ao fundo de investimentos Permira e Technology Crossover Ventures, que a comprou da Alcatel-Lucent por 1,5 bilhões de dólares em fevereiro de 2012.

Sua solução de entrada para *Contact Center* em nuvem é a **PureCloud**. Que é uma plataforma multicanal de comunicação unificada, que pode ser implementada rapidamente e é escalável. A infraestrutura *cloud* é provida pela Amazon Web Services, com capacidade “ilimitada” e com segurança integrada sem custo adicional. Oferece serviços do tipo:

- Omnichannel Engagement: Acompanha a jornada do cliente em todos os canais utilizados para atendimento, e com essas informações são geradas estatísticas e estratégias de venda pelo meio de aprendizado de máquina.
- Roteamento Avançado
- URA
- Discador inteligente
- Relatório e análises
- Otimização da força de trabalho
- Funções de supervisor
- Integração com CRM
- Chat
- Vídeo
- Segurança: Amazon AWS security, SSAE16 e conformidade PCI

Nas páginas de venda do serviço, não é encontrado nenhum relato do uso da tecnologia WebRTC, provavelmente para que não fique evidente para os concorrentes o uso da tecnologia como solução, ou porque não é uma informação imprescindível para o cliente. Porém, em algumas páginas de suporte como a https://help.mypurecloud.com/articles/get-purecloud-webrtc-phones-running/ é encontrada a evidência de sua utilização, tanto para voz quanto para vídeo. A Figura 10 ilustra a arquitetura da solução de *Contact Center* em nuvem da Genesys. Nela é possível ver as diversas maneiras que o cliente pode entrar em contato com os serviços da empresa. Bem como todas as suas ferramentas e soluções para formar o ambiente da solução.

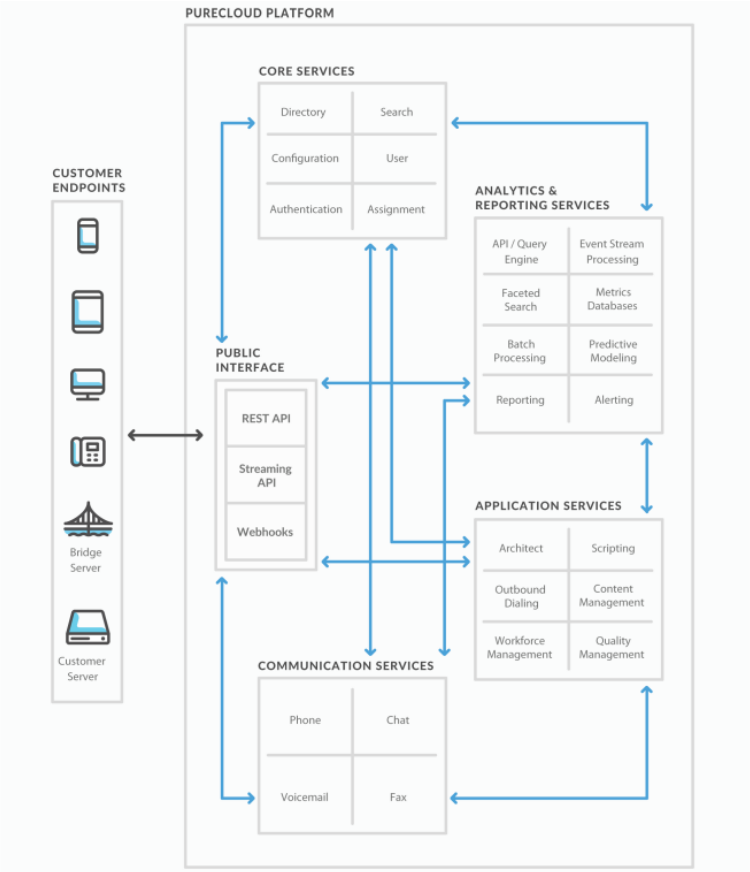


Figura 10. Arquitetura das aplicações da PureCloud Genesys.

O *frontend* web do agente tem sua ilustração na Figura 11, em que a agente está disponível em uma fila de atendimento e recebe a chamada de um cliente.

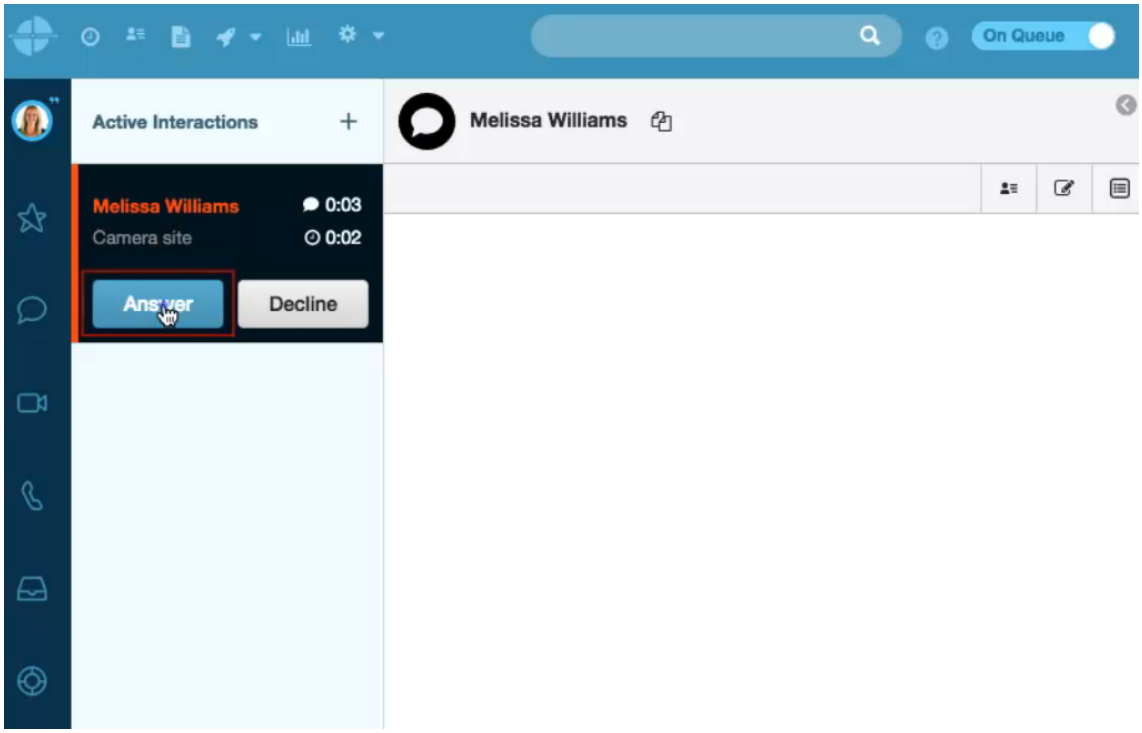


Figura 11. Tela de atendimento do operador - Frontend Web Genesys.

4.5.3. Zendesk

A ZenDesk oferece em seu portfólio soluções do tipo SaaS (software as a service) para atendimento ao cliente, disponível para qualquer tamanho de empresa. A empresa oferece produtos para atender as necessidades de equipes de suporte e help center integrando várias mídias, como chat, voz, email, chatbot, etc.

No ano de 2017 a Zendesk tornou o WebRTC a tecnologia incorporada em todos as suas versões de clientes de telefone. E com isso recebeu de seus clientes o feedback da melhora na experiência do atendimento, e com a mesma qualidade e eficiência e melhor performance em relação a tecnologia (flash) utilizada anteriormente.

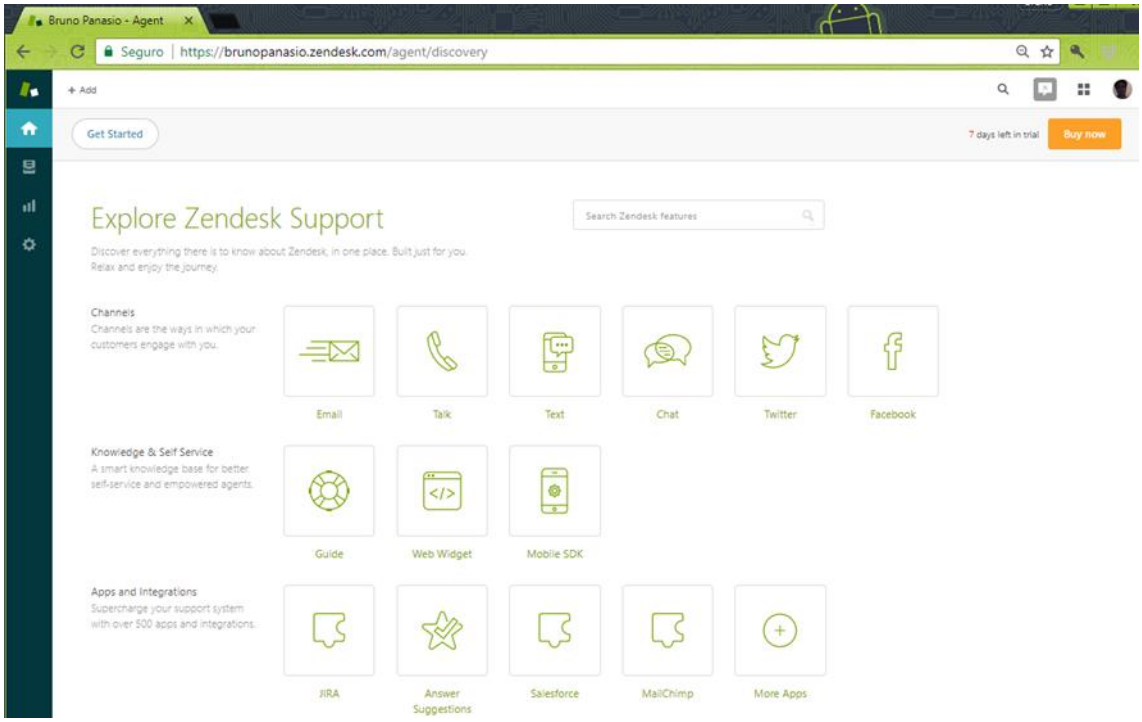


Figura 12. Tela de opções de mídias de atendimento - Frontend Web Zendesk.

A Figura 12 ilustra as opções de mídias que a plataforma provê como padrão. São conectores que puxam as informações diretamente das plataformas existentes como Facbook, Twitter, servidores de emails, etc. A Zendesk também oferece um *marketplace* que vende integrações fora desses padrões, como Instragram por exemplo.

5. Estudo de caso

A ideia deste estudo de caso é analisar o uso do WebRTC como tecnologia para centrais de atendimento em *cloud computing*. Neste estudo de caso é definido um escopo limitado e as ferramentas de validação. No final foram realizados os testes e validações do uso do WebRTC para a finalidade.

5.1. Especificação

O estudo de caso tem como objetivo desenvolver uma central de atendimento de teste simulando um ambiente real. O ambiente idealizado foi o atendimento de

possíveis dúvidas de alunos da Universidade Federal do ABC. Essas dúvidas podem ser sanadas via chat ou voz por um site utilizando o WebRTC como tecnologia de comunicação. Outra premissa é o oferecer esse serviço como *SaaS* (*software asa service*) em uma plataforma em nuvem, que faz a cobrança pela quantidade de serviço utilizada por período.

Para definição do escopo do estudo de caso, o fluxo de atendimento deve acontecer semelhante à Figura 13. Em uma das pontas o aluno acessa o site que contém o link da aplicação e aguarda sua vez caso haja fila. Na outra ponta da interação, um operador disponível no sistema atende o aluno. Iniciado o atendimento por chat, o atendente pode oferecer o atendimento por voz. Caso o atendimento por voz utilizando a tecnologia WebRTC seja realizado com sucesso, então o estudo de caso é concluído.

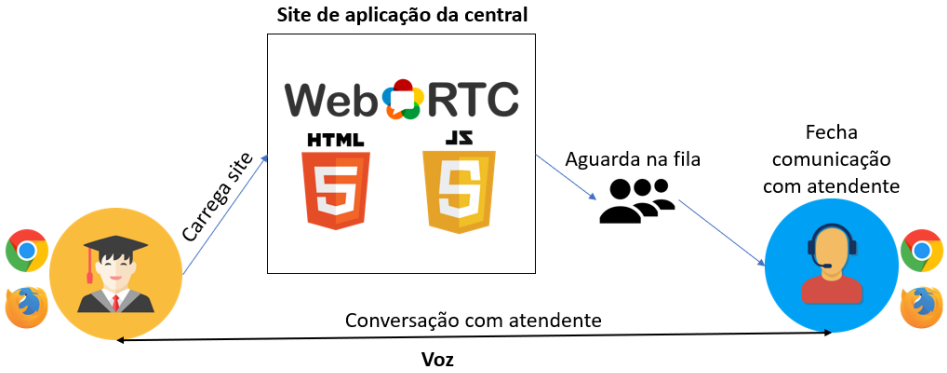


Figura 13. Arquitetura proposta.

Desta forma os recursos necessários para estabelecer um atendimento são um dispositivo com microfone e um navegador instalado compatível com WebRTC (Google Chrome ou Mozilla Firefox), e um servidor de aplicação. É recomendado que apenas o estudante possa utilizar um smartphone, e que o atendente utilize um computador com headset para garantir um melhor desempenho. Neste estudo de caso é abordado apenas o funcionamento da tecnologia WebRTC, não o dimensionamento de uma central de atendimento. No próximo capítulo, contém as ferramentas e topologia do estudo de caso estudado.

5.2. Metodologia

Neste capítulo são detalhados os passos para realização do estudo de caso e as ferramentas utilizadas. A solução utilizada é o serviço de Velip, e seus procedimentos de testes foram descritos. Como ferramentas de validação são apresentadas a *console*, *inspector* e o webrtc-internals do Google Chrome.

Como o foco deste trabalho não é a ferramenta Velip, o desenvolvimento e configuração da plataforma Velip está no **Anexo B**. Neste anexo, contém informações sobre o desenvolvimento do site da aplicação.

5.2.1. Ferramenta de solução do estudo de caso

A solução utilizada no estudo de caso é a plataforma de Contact Center em nuvem da empresa Velip. A aplicação oferece infraestrutura necessária para todos os serviços essenciais no atendimento ao cliente. Mais de um canal de atendimento ao foi crucial para escolha, e utilizar a tecnologia WebRTC em uma das mídias foi um pré-requisito. A facilidade de análise dos scripts de suas páginas HTML e JavaScript foi levada em consideração. As outras empresas se mostraram menos acessíveis, principalmente em para análise de código e indisponibilidade de versão para testes.

O desenho do fluxo e arquitetura da central de atendimento é ilustrada na Figura 14. Essa figura mostra que embutido no site da central de atendimento existe o a aplicação da Velip, que realiza o processo de comunicação entre os participantes.

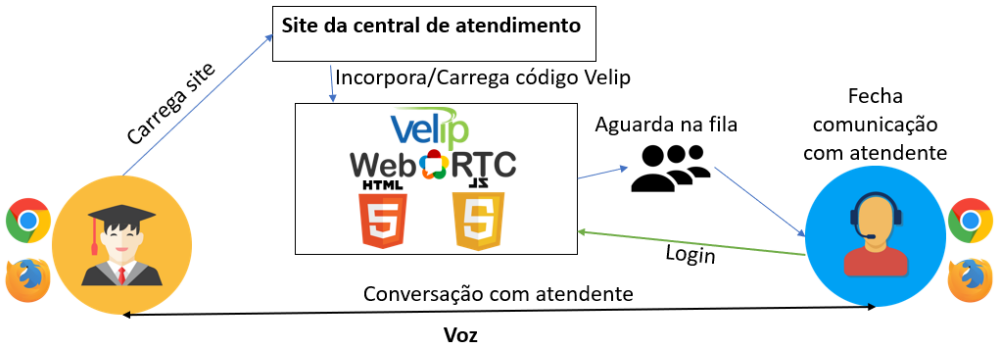


Figura 14. Arquitetura do estudo de caso da Central utilizando a ferramenta Velip.

Para a criação dessa central de atendimento e validação do estudo de caso na plataforma Velip é necessário configurar, conforme **ANEXO B**, os seguintes pontos no sistema:

- Criar uma Campanha Web para atendimento online.
- Programar as filas necessárias e agentes.
- Configurar especificações da campanha.
- Implantar o código HTML/Javascript em um site na internet.
- Simular um atendimento de voz com o cliente.

5.2.2. Ferramentas de validação do estudo de caso

Em função de evidenciar e validar o estudo de caso, são utilizadas duas ferramentas distintas do navegador. A primeira delas é o **webrtc-internals**, e a outra é a **Ferramenta do desenvolvedor**. As nomenclaturas e atalhos podem variar de navegador para navegador, mas no final exibem as mesmas informações. Essas ferramentas permitem olhar em que ponto são chamadas e quais as funções que utilizam a API do WebRTC. Permitem também analisar de maneira fácil quais os protocolos, codecs e detalhes de informação trocadas. Outra característica importante são os gráficos que auxiliam na detecção de perda de pacotes, banda utilizada, cancelamento de eco, pacotes por segundo, entre outros. Finalmente, por meio do console é possível acompanhar os logs programados da aplicação.

Disponível em todos os navegadores, a **ferramenta do desenvolvedor** mostra informações da página web aberta na tela, e tudo que a página importa e incorpora em seu código. Mostra também os logs dos sites e aplicações carregadas nas páginas. A ferramenta de desenvolvedor pode ser acessada pelo menu do navegador, e utilziando atalhos do teclado como CRTL+SHIFT+J no Google Chrome. A Figura 15 ilustra respectivamente a ferramenta de desenvolvedor do Mozilla Firefox e Google Chrome.

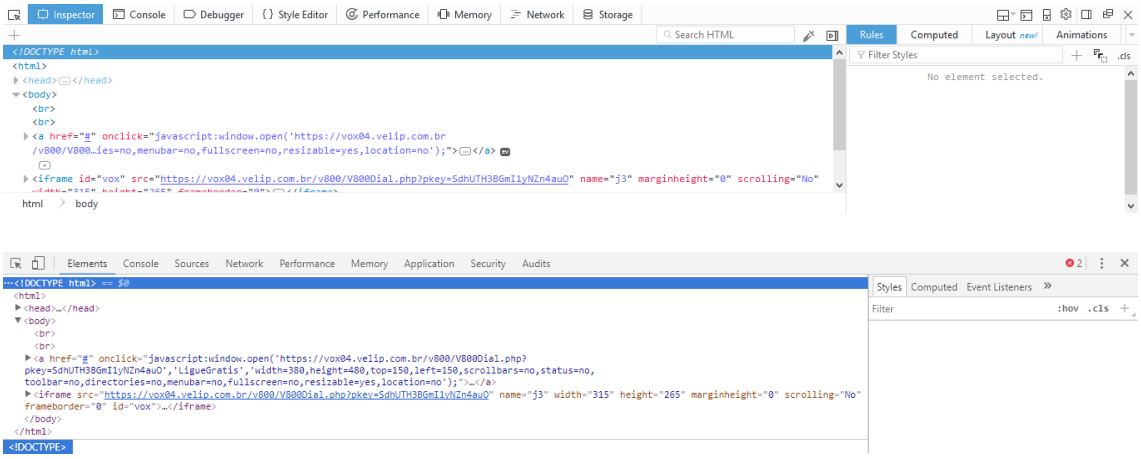


Figura 15. Ferramenta do Desenvolvedor, respectivamente no Mozilla Firefox e Google Chrome.

Para validação deste estudo de caso são utilizadas apenas as abas ***Inspector/Elements*** e ***Console***. Que respectivamente mostram o código fonte da página e os logs de erro e dos aplicativos.

Em **Inspector/Elements**, para analisar as funções que chamam as APIs do WebRTC, são identificados trechos de código do tipo da Tabela 2.

Código	Função
navigator.getUserMedia(constraints, gotStream, logError);	Requisita fluxo de áudio e/ou vídeo para o navegador.
RTCPeerConnection({})	Responsável pelo gerenciamento das conexões

Tabela 2 Exemplo de código a ser localizado no inspector.

Em **Console** é exibido qualquer erro, alertas e logs programados em funções e também APIs do WebRTC. O console ajuda em descoberta e solução de problemas.

Webrtc-internals

Com o **webrtc-internals** é possível verificar e analisar uma conexão entre os pares, detalhando quais os protocolos e codecs trocados por exemplo. Outra característica importante são os gráficos que auxiliam na detecção de problemas e acompanhamento de estatísticas. Por exemplo a perda de pacotes, largura de banda utilizada, cancelamento de eco, pacotes por segundo, entre outros são atualizados constantemente nos gráficos.

A Tabela 3 informa o endereço de acesso à ferramenta para alguns navegadores.

Navegador	Endereço
Google Chrome	chrome://webrtc-internals
Opera	opera://webrtc-internals
Mozilla Firefox	about:webrtc

Tabela 3 Links para acesso ao Webrtc-internals por navegador.

Para cada conexão estabelecida, é criado um novo par de abas dentro do Webrtc-internals que exibem as informações da conexão entre cada ponta. Na Figura 16 é ilustrado um exemplo de conexão de áudio estabelecida.

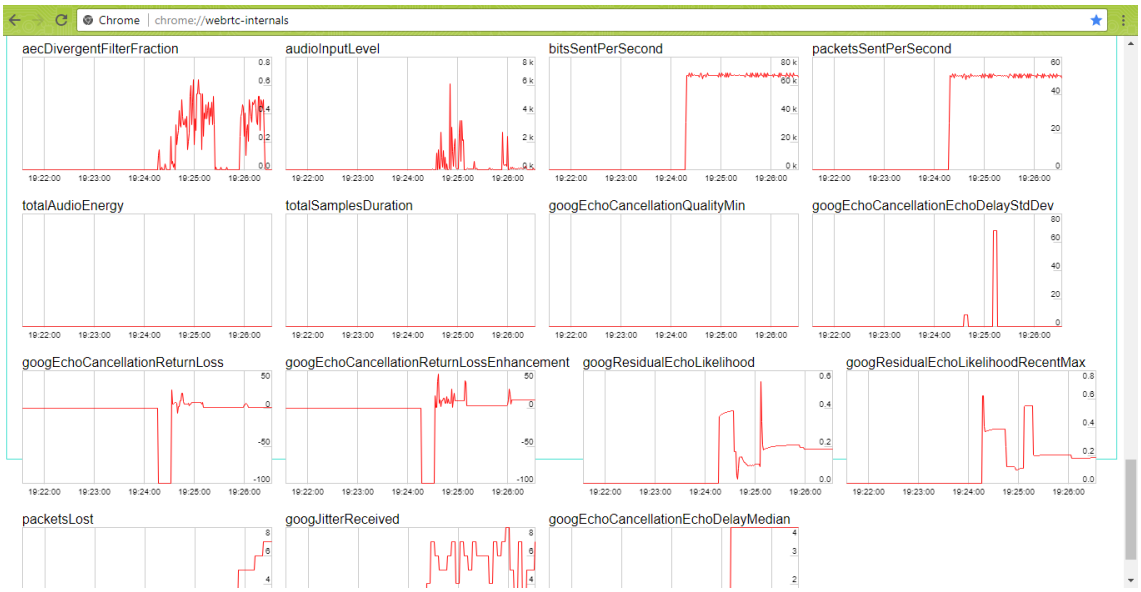


Figura 16. Tela da ferramenta WebRTC-Internals.

5.3. Teste e validação da simulação do atendimento

Neste passo, o site com a aplicação WebRTC é acessado pelo aluno, e o operador é logado na plataforma da Velip. Para facilitar o entendimento, o processo é separado em subcapítulos. Em cada subcapitulo são descritos os procedimentos de login do operador, acesso ao site e aguardo na fila, atendimento via chat e finalmente atendimento por voz via navegador. Em cada etapa são validadas a aplicação do WebRTC como ferramenta.

O navegador utilizado para o estudo de caso é o Google Chrome, tanto no lado do aluno (smartphone), quanto do lado do operador (computador).

5.3.1. Login do operador

O operador acessa o site de login da Velip (https://vox04.velip.com.br/pop/vox/vox_login_cli.php) conforme ilustrado na Figura 17. Se autentica preenchendo o “Usuário” e a “Senha”.

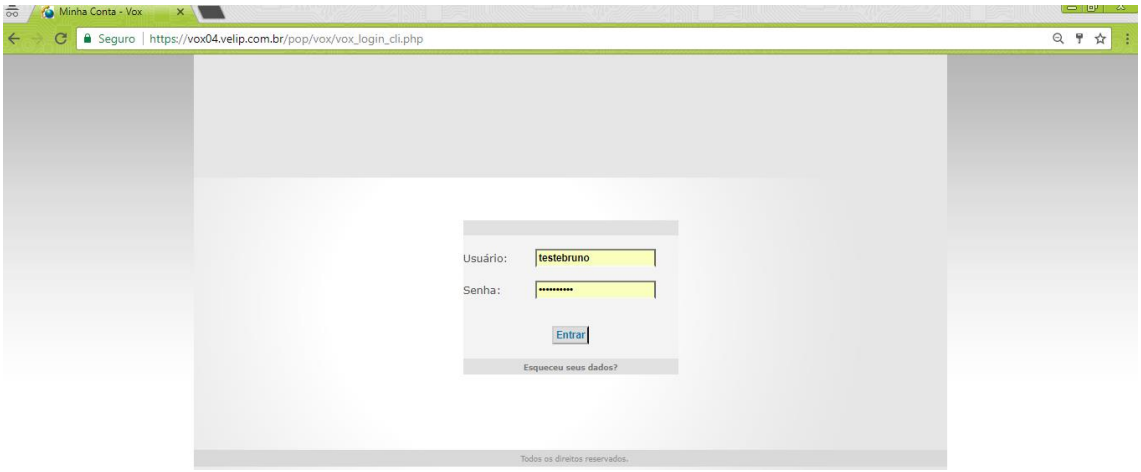


Figura 17. Login do Operador.

Após o login, o operador é automaticamente redirecionado para a página de **Posição de Atendimento Virtual** (**PA Virtual**), ilustrada pela Figura 18. O primeiro passo é permitir o uso de microfone pela página por meio do balão de alerta gerado no canto superior esquerdo da página. Esse pedido de permissão se deve ao trecho do código da função **navigator.getUserMedia** da API do WebRTC.

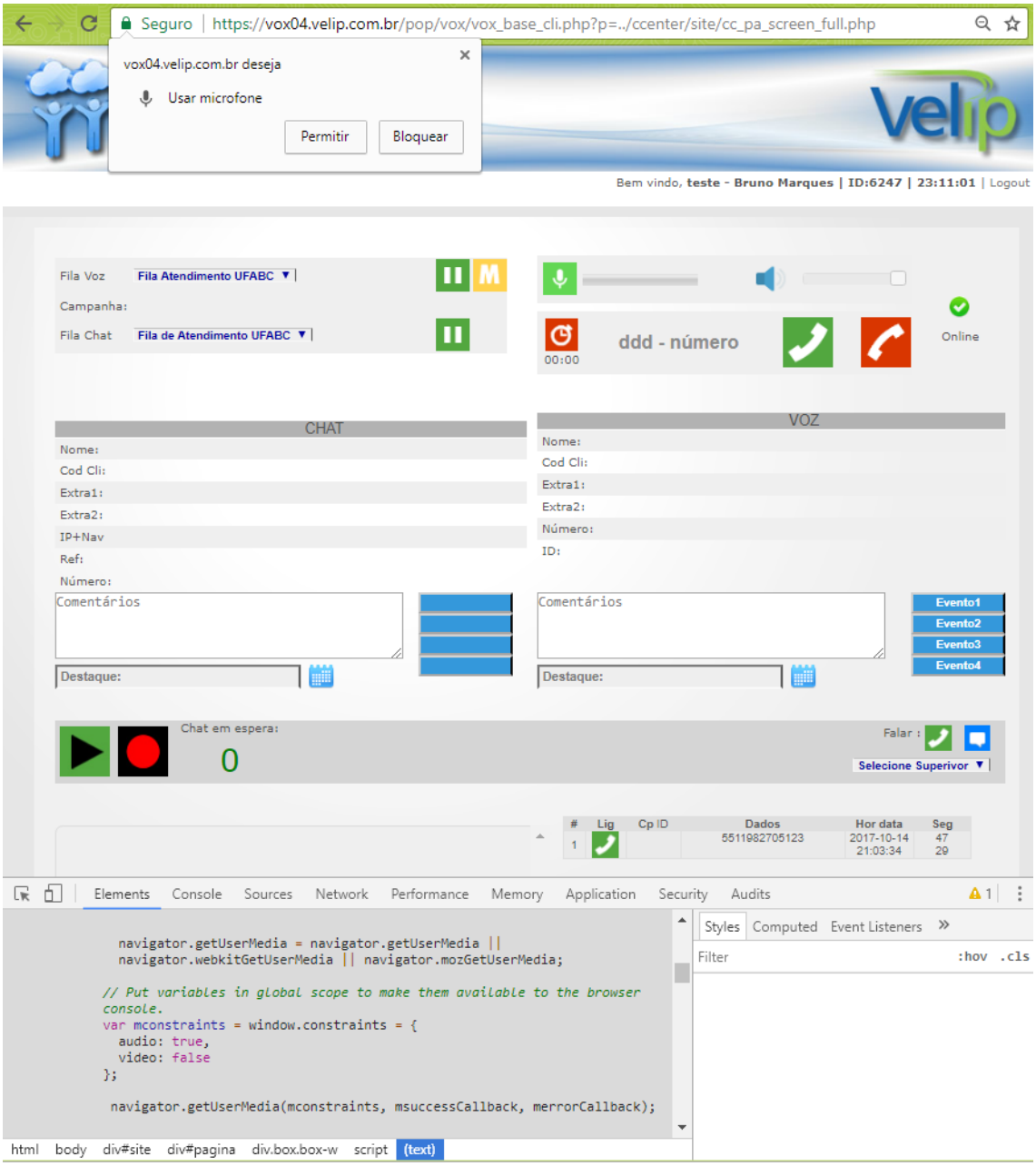


Figura 18. Posição de Atendimento Virtual Experimeto Velip.

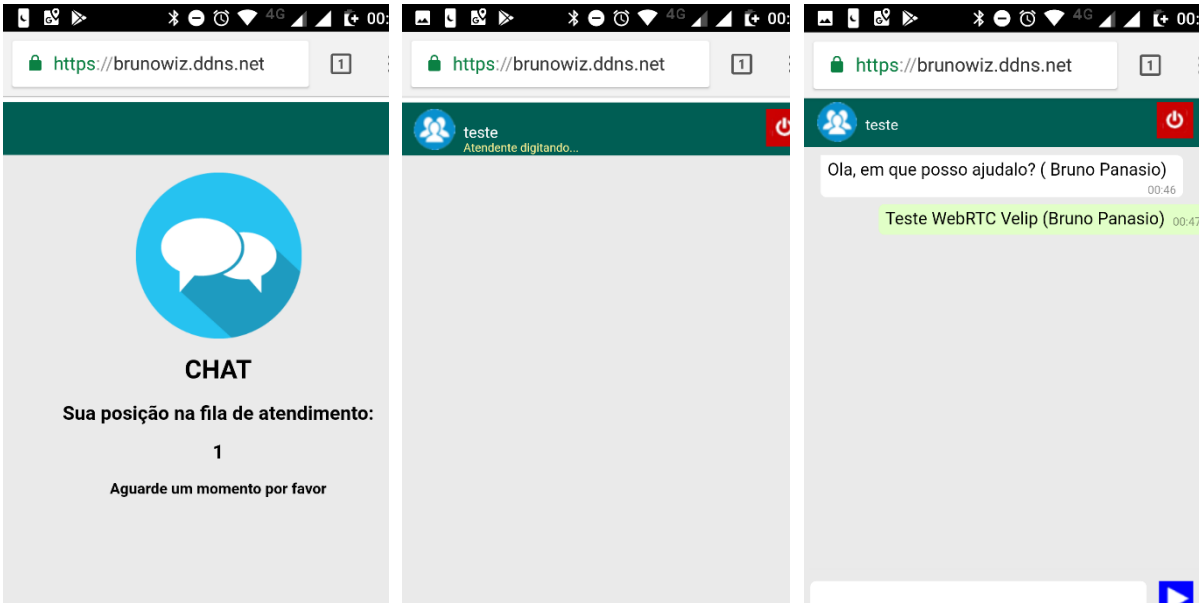
A utilização da API **getUserMedia** do WebRTC é evidenciada pelo trecho de código visualizado no ***Elements*** da **Ferramenta do Desenvolvedor**. A Figura 18 ilustra a captura da evidencia. No código é possível notar que solicitado apenas o dispositivo de áudio do usuário, que está de acordo com a permissão de acesso ao microfone da *popup*.

Em seguida é selecionada a Campanha “**Fila Atendimento UFABC**” e a fila “**Fila de Atendimento UFABC**”. Com isso o operador se torna disponível para iniciar

um atendimento. A fila de clientes está zerada por não haver clientes em espera para ser atendido no momento.

5.3.2. Acesso do aluno na central de atendimento

O estudante acessa o endereço http://brunowiz.ddns.net, e aguarda na fila seu atendimento. Este passo é ilustrado na Figura 19 a. Na Figura 19 b, é ilustrado o momento após o início do atendimento pelo operador.



(a) Fila de espera (b) Aluno em atendimento (c) Atendimento via chat

Figura 19. Tela do cliente no atendimento via chat.

O estudante e operador podem realizar um atendimento por chat conforme ilustra a Figura 19 c. É preciso ressaltar que a Velip não utilizou o WebRTC para sua solução de chat, apenas para solução de voz.

5.3.3. Atendimento por voz usando a ferramenta

Para o atendimento ser realizado por voz, o agente oferece essa opção durante o chat para o aluno. E ao agente clicar no botão “Voz”, ilustrado na Figura 20 da **PA Virtual**, a conversa por áudio é iniciada.

Figura 20. Barra de comunicação do operador Velip - Botão Voz.

Após o convite para a mídia voz, uma permissão para acesso ao microfone do aparelho do lado do cliente é exibida no navegador do smartphone, conforme ilustração da Figura 21. O endereço que solicita a permissão é o da plataforma Velip, vx04.velip.com.br, que é o endereço onde ficam hospedados os servidores que fazem a intermediação inicial (handshake) entre as pontas, antes que elas fechem conexão direta.

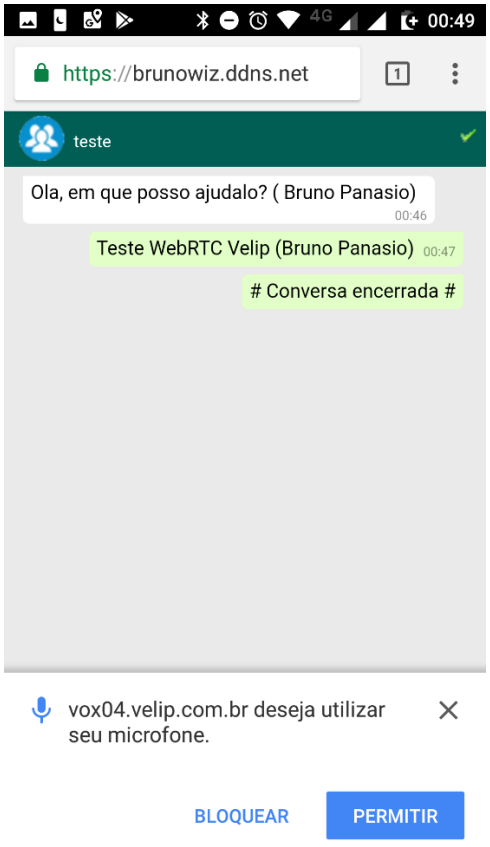


Figura 21. Solicitação do navegador para permissão de acesso ao microfone do smartphone do aluno.

Após o aceite, a troca de informações de protocolos e codecs é negociada, e a comunicação entre as pontas é concretizada.

Neste estudo de caso, a validação da troca de protocolos e o áudio em si é observado via ferramenta **webrtc-internals**, acessado pelo Google Chrome do lado do operador. De acordo com a Figura 22, o Codec PCMA de áudio foi o acordado entre os pares para conversação em voz.

Figura 22. Codec utilizado na conversa do estudo de caso - Webrtc-internals.

O *streaming* de mídia de voz é evidenciado na Figura 23, que ilustra os bits por segundo recebidos pelo operador. A taxa média é próxima de 64Kb/s, que é o bitrate comum do codec PCMA.

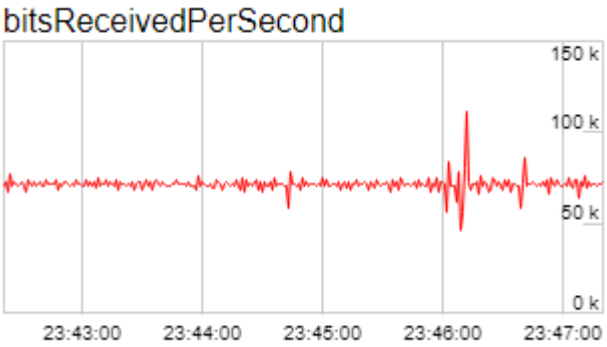
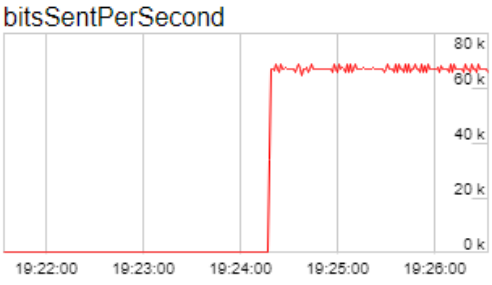
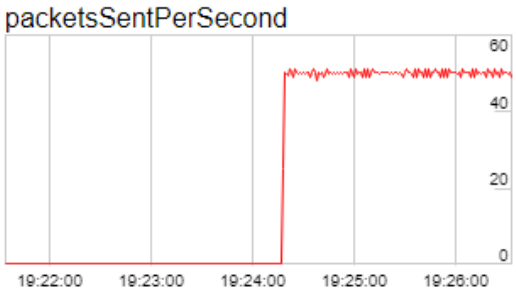


Figura 23. Taxa de dados recebidos pelo operador - Webrtc-internals.

Na Figura 24 é ilustrado um gráfico com a taxa dos pacotes enviados pelo operador. Além do *payload* de voz, é enviado também alguns bytes de controle como RTCP. enviados pela média de 50 pacotes enviados por segundo.



(a) Bits enviados por segundo



(b) Pacotes enviados por segundo

Figura 24. Gráficos dos bits e pacotes enviados por segundo.

Esses dados afirmam tecnicamente a funcionalidade da tecnologia WebRTC para a comunicação de voz entre navegadores. De acordo com a percepção do usuário de testes, a impressão foi de uma chamada de voz convencional. O usuário não constatou delay perceptível.

O atendimento do caso especificado e simulado foi finalizado com sucesso. O resultado foi um atendimento da mídia voz utilizando a tecnologia WebRTC. O aluno conseguiu ser atendido utilizando o navegador de seu smartphone. Do outro lado da conversa, o agente realizou com êxito o atendimento por meio de seu computador.

A ferramenta Velip foi validada conforme cenário proposto. Validando a sua capacidade de oferecer tais serviços totalmente em um ambiente online e com

qualidade. Sem a necessidade de compra ou instalação de nenhum software ou equipamento, modelo este conhecido como *Software as a Service*.

5.4. Engenharia reversa – HTML, JavaScript, JsSIP e WebRTC

Neste capítulo são estudados os códigos das páginas utilizadas pela ferramenta Velip. Esta engenharia reversa tem o intuito de validar as chamadas das funções WebRTC e suas dependências. Com isso, é possível entender o funcionamento da ferramenta. Foram inseridas apenas as linhas de código mais importantes para o funcionamento do atendimento por voz via WebRTC. Foram inseridos alguns comentários com intuito de facilitar o entendimento. Os comentários podem ser identificados pelo seguinte formato **"//comentário", BP**.

No cabeçalho da página foi encontrada a referência da importação da biblioteca do JsSIP. A versão importada é a 3.0.4, e seu funcionamento é abordado no Capítulo 3 deste trabalho.

```
<script src="/pop/comum/jssip3/jssip.js" type="text/javascript"></script>
/*
 * JsSIP v3.0.4
 * the Javascript SIP library
 * Copyright: 2012-2017 José Luis Millan <jmillan@aliax.net>
*(https://github.com/jmillan)
 * Homepage: http://jssip.net
 * License: MIT
 */
```

Também no cabeçalho HTML, o código JavaScript em seguida está embutido. Neste script é utilizada a API getUserMedia do WebRTC para capturar as informações de áudio (Microfone). Quando esse script é executado então surge a *popup* de permissão do microfone no navegador.

```
navigator.getUserMedia = (navigator.getUserMedia ||
```

```
navigator.webkitGetUserMedia || navigator.mozGetUserMedia ||
        navigator.msGetUserMedia);
var mconstraints = window.constraints = {
audio: true,
video: false
};
```

O cabeçalho HTML da página ainda traz o código abaixo, com as funções em JavaScript do WebRTC e JsSIP. O código também realiza a instanciação e registro do *client* JsSIP que se comunica com o SIP Server Velip.

```
// WEBRTC
function conexao () { // conexao
var socket = new
JsSIP.WebSocketInterface('wss://vox10.velip.com.br:8089/ws');
//”o código acima abre um WebSocket com o servidor Velip”, BP
  var configuration = {
  sockets  : [ socket ],
  'uri': 'sip:CIE65882@vox10.velip.com.br',
  'password': getsenha,
  'register': true,
  'register_expires': '300',
  'display_name': "CIE65882",
  'session_timers':false
};
//”o código passa os parâmetros para o JsSIP conectar ao SipServer”, BP
coolPhone = new JsSIP.UA(configuration);
//”o código acima instância de fato um cliente SIP do JsSIP”, BP
//”ocódigo abaixo realiza as ações necessárias de acordo com o status que a...
// ...instância do JsSIP recebe de status”, BP
coolPhone.on('connected', function(e){ });
coolPhone.on('disconnected', function(e){  });
coolPhone.on('registered', function(e){  CallPA(VoxDestID) ; });
```

```
coolPhone.on('unregistered', function(e){  });
coolPhone.on('registrationFailed', function(e){ });
coolPhone.on('newRTCSession', function(e){
rtcSession = e.session;
        displayname=rtcSession.remote_identity.display_name;
rtcSession.on('accepted',function(e){if (At_sel==1) { }
});
rtcSession.on('ended',function(e){
resetOff = setTimeout("CHendChat('timeout');",300000);
coolPhone.stop();
$("#imVozEnd").fadeOut(300);
});
rtcSession.on('failed',function(e){  }); // falhou ao conectar
rtcSession.on('confirmed',function(e){
ph_addStream();
document.getElementById('imVozEnd').style.display="";
clearTimeout(resetOff);
  });
  });
// fim NewRTCSession
//" O código abaixo inicializa a instancia do JsSIP declarada acima", BP
  coolPhone.start();
```

As informações trazidas nos códigos da página validam a utilização do WebRTC e JsSIP para solução de Voz da ferramenta. É possível encontrar informações específicas dos servidores Velip, como por exemplo o SIP server em que o JsSIP se conecta para trocar sinalização via WebSockets.

6. Conclusão

Com o trabalho realizado, chegou-se à conclusão de que o emprego do WebRTC vai de encontro com a evolução do *contact center*, pois a tecnologia resolve e facilita os problemas para o desenvolvimento de um atendimento multicanais. Dentre os canais já conhecidos como, voz, vídeo, chat, transferência de dados (ex. localização) e arquivos, já é possível encontrar de maneira isolada tais soluções no mercado. Devido ao WebRTC ter sido desenvolvido para o mundo *web*, isso amplia a variedade de plataformas compatíveis com a tecnologia.

O WebRTC está de acordo com a convergência das novas tecnologias de infraestrutura em nuvem e comercialização de *SaaS*. Assim deixando a tecnologia como opção de *roadmap* tecnológico e de mercado viável.

O estudo de caso realizado no Capítulo 5 foi considerado concluído com sucesso, pois o resultado da simulação de um atendimento empregando o WebRTC como tecnologia funcionou conforme o escopo proposto. A ferramenta Velip de *contact center* em nuvem proporcionou o ambiente necessário para os testes. Não foi necessária aquisição de nenhum hardware ou software, apenas os serviços em nuvem de hospedagem para o site e o serviço da Velip.

Uma ideia de trabalho futuro que este projeto despertou é desenvolver uma plataforma intermediadora de *contact center*, empregando o WebRTC como tecnologia de solução multimídia. A ideia é intermediar as empresas que necessitem realizar um atendimento ao cliente, com pessoas que seriam prestadoras de serviços de atendimento ao cliente. Oferecendo todas os principais canais de mídias por meio de uma plataforma *online* acessada via navegador ou aplicativo de celular. A infraestrutura seria baseada em nuvem para comportar escalabilidade e alta disponibilidade do serviço.

7. Referências

ALVARENGA, D. **Operações em canais digitais são 62% do total e motivam fechamento de agências**. G1 Globo, março 2017. Disponível em: <http://g1.globo.com/economia/noticia/operacoes-em-canais-digitais-sao-62-do-total-e-motivam-fechamento-de-agencias.ghtml>. Acessado em: 19 out. 2018.

ANTONELLI, V. **Call Center Virtual e Multicanal**. Revista Call Center, abril 2015. Disponível em <http://revistadocallcenter.com.br/tecnologia/132-call-center-virtual-e-multicanal.html>. Acessado em: 19 out. 2018.

ARMBRUST, M. et al. **Above the Clouds:** A Berkeley View of Cloud Computing. Technical Report No. UCB/EECS-2009-28, Electrical Engineering and Computer Sciences University of California at Berkeley, devereiro 2009. Disponível em: <https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf>. Acessado em: 19 out. 2018.

BERGEVIN, R.; AFSHAN, K.; W. S. B. S., [et al.]. **Call Centers For Dummies.** 2ª Edição. Canada: John Wiley & Sons, 2010. ISBN 978-0-470-67743-8.

BERGKVIST, A.; et al. **WebRTC 1.0:** Real-time Communication Between Browsers. W3C Candidate September 2018. W3C, setembro 2018. Disponível em: <https://www.w3.org/TR/webrtc/>. Acessado em 18 out. 2018.

CONGO, M. **WhatsApp e Skype ameaçam lucros das operadoras**. Estadão, 25 fevereiro 2013. Disponível em: <http://economia.estadao.com.br/blogs/radar-tecnologico/whatsapp-e-skype-ameacam-lucros-das-operadoras/>. Acessado em: 15 out. 2018.

Cresceram...e viaram clientes! ClienteSA, ed. 157, junho 2016. Disponível em: <http://revista.clientesa.com.br/scribd/?251/Edicao-157-Junho-2016>. Acessado em: 19 out. 2018.

DUTTON, S. **Getting Started with WebRTC**. HTML5 Rocks, fevereiro 2014. Disponível em: <https://www.html5rocks.com/en/tutorials/webrtc/basics/>. Acessado em 19 out. 2018.

FETTE, I.; MELNIKOV, A., **The WebSocket Protocol** Request for Comments 6455. IETF, dezembro 2011. Disponível em: <https://tools.ietf.org/html/rfc6455>. Acessado em: 15 out. 2018.

GIÃO, P.; BORINI, F.; OLIVEIRA, M. **The influence of technology on the performance of Brazilian call centers**. São Paulo: JISTEM J.Inf.Syst. Technol. Manag. Vol 7 (Online), 2010. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1807-17752010000200005&lng=pt&nrm=iso>. Acessado em: 15 out. 2018.

GRUBE & ASSOCIADOS. **Sintelmark traça previsões para 2016**. 2016. Disponível em: <http://callcenter.inf.br/estatisticas/61176/sintelmark-traca-previsoes-para-2016/ler.aspx>. Acessado em: 15 out. 2018.

G.711: Pulse code modulation (PCM) of voice frequencies. ITU-T, 1988a. Disponível em: <https://www.itu.int/rec/T-REC-G.711/e>. Acessado em 19 out. 2018.

G.722: 7 kHz audio-coding within 64 kbit/s. ITU-T, 1988b. Disponível e:m <https://www.itu.int/rec/T-REC-G.722/e>. Acessado em 19 out. 2018.

HANDLEY, M.; SCHULZRINNE, H.; SCHOONER, E.; ROSENBERG, J. **SIP: Session Initiation Protocol** Request for Comments 2543. IETF, março 1999. Disponível em: <https://www.ietf.org/rfc/rfc2543.txt>. Acessado em: 15 out. 2018.

HICKSON, I. **The WebSocket API** W3C Candidate Recommendation. W3C, setembro 2012. Disponível em: <https://www.w3.org/TR/websockets/>. Acessado em: 15 out. 2018.

HOLMBERG, C.; et al. **Web Real Time Communication Uses Cases and Requirements**. IETF, março 2015. Disponível em: <https://tools.ietf.org/html/rfc7478>. Acessado em: 19 out. 2018

Home Office: a tecnologia para o melhor atendimento. Revista Call Center, março 2016. Disponível em: <http://revistadocallcenter.com.br/tecnologia/2406-home-office-a-tecnologia-para-o-melhor-atendimento.html>. Acessado em: 19 out. 2018.

KERANEN, A.; HOLMBERG, C.; ROSENBERG, J. **Interactive Connectivity Establishment (ICE):** A Protocol for Network Address Translator (NAT) Traversal. RFC 8445. IETF, juho 2018. Disponível em: <https://tools.ietf.org/html/rfc8445>. Acessado em: 23 out. 2018.

LAJTOS, I.; O'BYRNE, D.; ERSOZ, E. **WebRTC to complement IP Communication Services**. White paper. GSM Association, fevereiro 2016. Disponível em: <http://www.gsma.com/network2020/wp-content/uploads/2016/02/WebRTC_to_complement_IP_Communication_Services_v1.0.pdf>. Acessado em 19 out. 2018.

LE GRAND, T.; et al. **RTP Payload Format for the iSAC Codec**. Internet-Draft draft-ietf-avt-rtp-isac-04. IETF, fevereiro 2013. Disponível em: <https://tools.ietf.org/html/draft-ietf-avt-rtp-isac-04>. Acessado em 18 out. 2018.

MADRUGA, R. **A História do Telemarketing e Call Center:** da fase artesanal ao canal de relacionamento, Conquist Educacional e Consultoria. 2015. Disponível em:

<http://www.conquist.com.br/blog-e-artigos/a-historia-do-telemarketing-e-call-center-da-fase-artesanal-ao-canal-de-relacionamento/>. Acessado em: 15 out. 2018.

MILLÁN, J.; CASTILLO, I.; CORRETGÉ, S. **JsSIP** The JavaScript SIP Library. Setembro 2012. Disponível em: <http://jssip.net/documentation/3.2.x/getting_started/>. Acessado em: 15 out. 2018.

OLEJNICZAK**,** S. **VoIP Deployment For Dummies**. Indianapolis: Wiley Publishing, 2009.

Packet-based multimedia communications systems. Recommendation ITU-T H.323, (1999).

PORTINARI, N. **Atendimento por call center tem horas contadas, diz presidente da Telefônica**. Folha, 08 de agosto de 2017. Disponível em: <http://www1.folha.uol.com.br/mercado/2017/08/1907038-atendimento-por-call-center-tem-horas-contadas-diz-presidente-da-telefonica.shtml>**.** Acessado em: 15 out. 2018.

ROSENBERG, J.; SCHULZRINNE, H. **An Offer/Answer Model with the Session Description Protocol (SDP)** Request for Comments 3264. IETF, junho 2002. Disponível em: <https://tools.ietf.org/html/rfc3264>. Acessado em: 15 out. 2018.

SCHULZRINNE, H.; CASNER, S.; FREDERICK, R.; JACOBSON, V. **RTP: A Transport Protocol for Real-Time Applications** Request for Comments 2543. IETF, julho 2003. Disponível em: <https://tools.ietf.org/html/rfc1889>. Acessado em: 15 out. 2018.

TSAHI, L. **10 Massive Applications Using WebRTC**. BlogGeek.Me, dezembro 2017. Disponível em: <https://bloggeek.me/massive-applications-using-webrtc/>. Acessado em: 10 out. 2018.

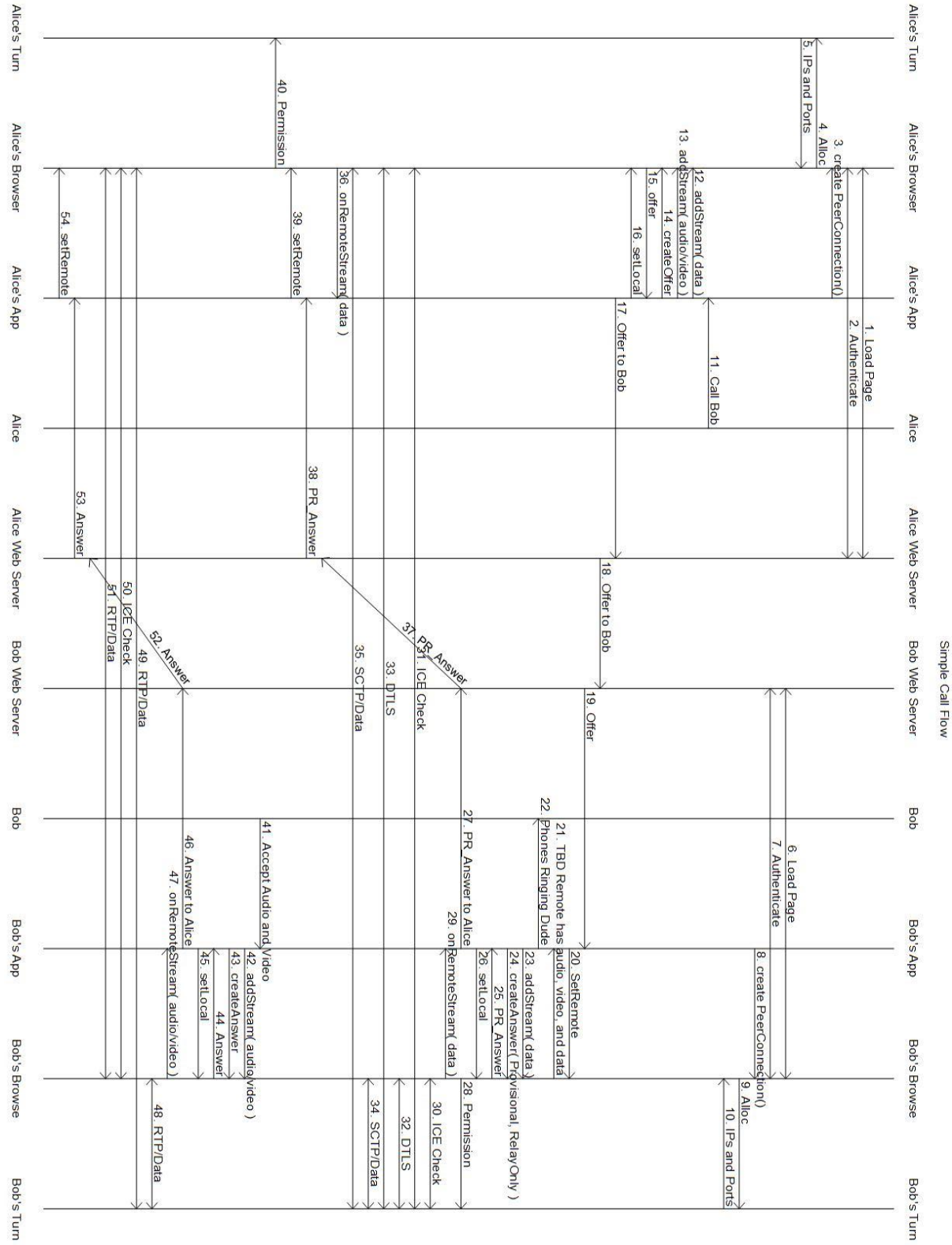
TWILIO. **WebRTC** ROLLS OUT IN BUSINESS APPS. White-paper Twilio, 2016. Disponível em: <https://www.twilio.com/white-papers/read-webrtc-rolls-out-in-business-apps>. Acessado em: 2 fev. 2018.

VALIN, JM.; et al. **Definition of the Opus Audio Codec**. Request for Comments: 6716. IETF, setembro 2012. Disponível em: <https://tools.ietf.org/html/rfc6716>. Acessado em: 19 out. 2018

VASS, J. **How Discord Handles Two and Half Million Concurrent Voice Users using WebRTC**. Discord, setembro 2018. Disponível em: <https://blog.discordapp.com/how-discord-handles-two-and-half-million-concurrent-voice-users-using-webrtc-ce01c3187429>. Acessado em: 10 out. 2018.

8. Anexos

8.1. ANEXO A – Fluxo dos métodos WebRTC para uma chamada



Fluxo de chamada entre Alice e Bob via WebRTC (BERGKVIST, 2018).

8.2. ANEXO B – Desenvolvimento e configuração da central de atendimento na plataforma Velip

Criar conta

Para utilizar a plataforma Velip é necessário criar uma conta. A conta é solicitada pelo site http://www.velip.com.br/. Clique em "**Login"**, e depois em "**Abrir nova conta**". Preencha o formulário de cadastro.

Criação da fila

Para alocar os atendentes e os alunos na fila de espera, foi criada a fila "Fila de atendimento UFABC". O caminho para criação é dado pelo menu *Call Center>Filas* e então no botão **Cadastrar**. O cadastro da nova fila foi preenchido conforme a figura. Foi escolhido o Chat no campo "**Utilizado em**" e preenchido "Fila de atendimento UFABC" no campo "**Nome da Fila**". Pelo botão "**Cadastrar**" a fila foi criada com as informações inseridas.

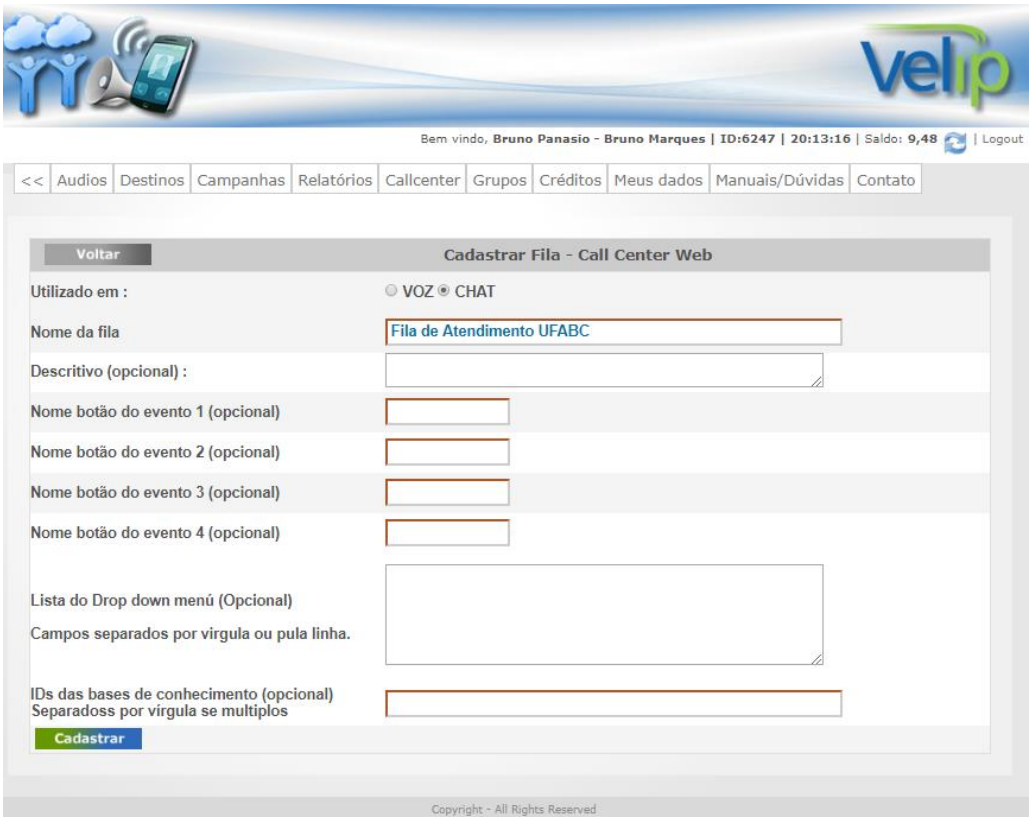


Figura Cadastro de fila - estudo de caso Velip.

A opção de **voz** não foi selecionada pois este é um modelo diferente do utilizado no estudo de caso, em que o cliente não entra em contato pela internet.

Criação da Campanha

Para disponibilizar o código HTML ao site, é necessário o cadastro da Campanha. Sua única dependência é ter uma fila, cadastrada no passo anterior.

O cadastro da campanha é feito pelo menu em *Call Center>Campanhas>Web* no botão **Cadastrar nova campanha Web**.

O preenchimento do cadastro foi realizado conforme figura.

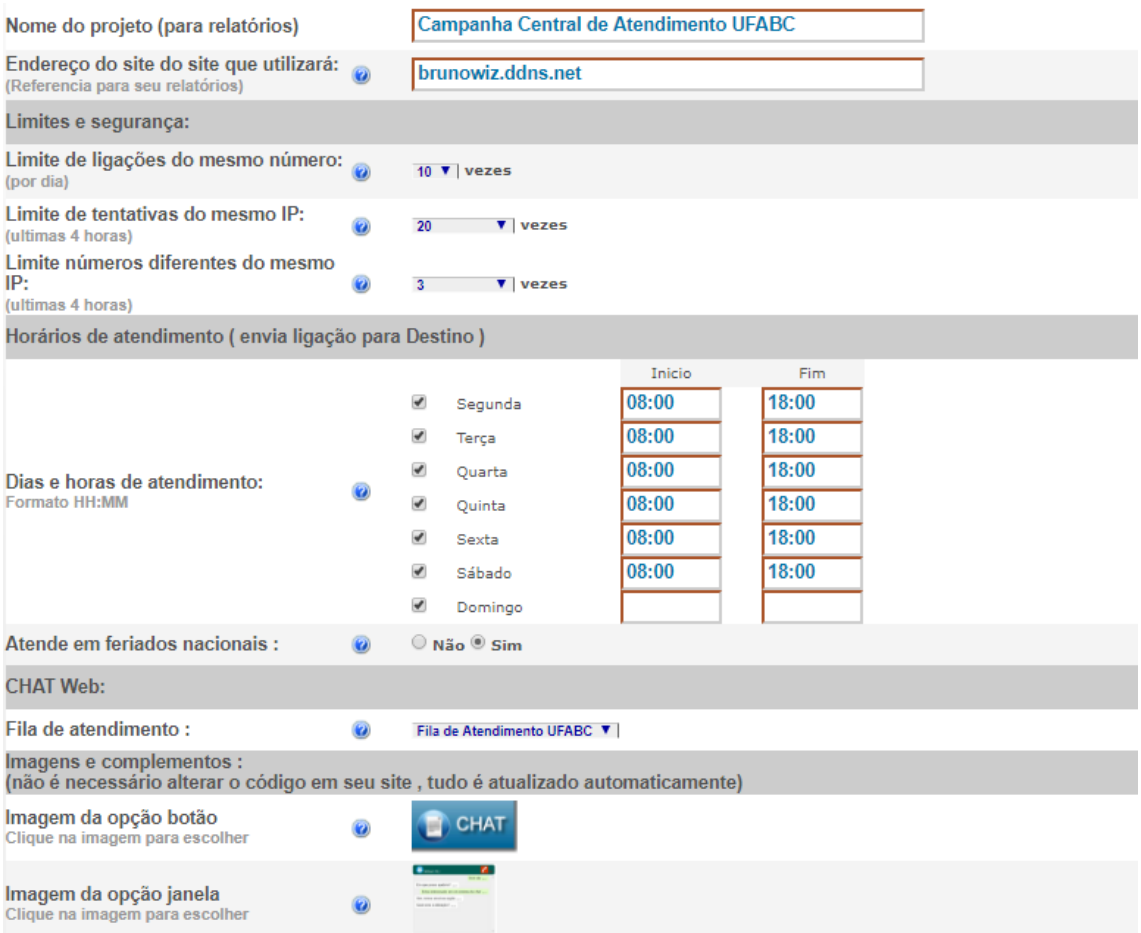


Figura Tela de cadastro da Campanha de Atendimento do estudo de caso Velip.

A informação mais importante deste cadastro é a "**Fila de Atendimento**" e foi selecionada a fila "**Fila de atendimento UFABC**", cadastrada no passo anterior. Os

campos restantes foram deixados como padrão. Após preenchido, o cadastro foi finalizado no botão **Cadastrar**.

Em seguida a página é atualizada automaticamente, e mostra dois botões com as opções “Opção Botão + Janela” e “Opção formulário na Tela (click to Call)”. Respectivamente a diferente é que na primeira opção, é gerado um código HTML de um botão para o cliente clicar e abrir uma nova janela para o atendimento. A segunda opção incorpora o atendimento dentro da própria página por meio do código HTML inserido. No estudo de caso, o código HTML inserido no site de atendimento ao estudante é o que incorpora o atendimento na página, ou seja, a opção do “formulário na Tela (click to Call)”, conforme ilustra a figura seguinte.

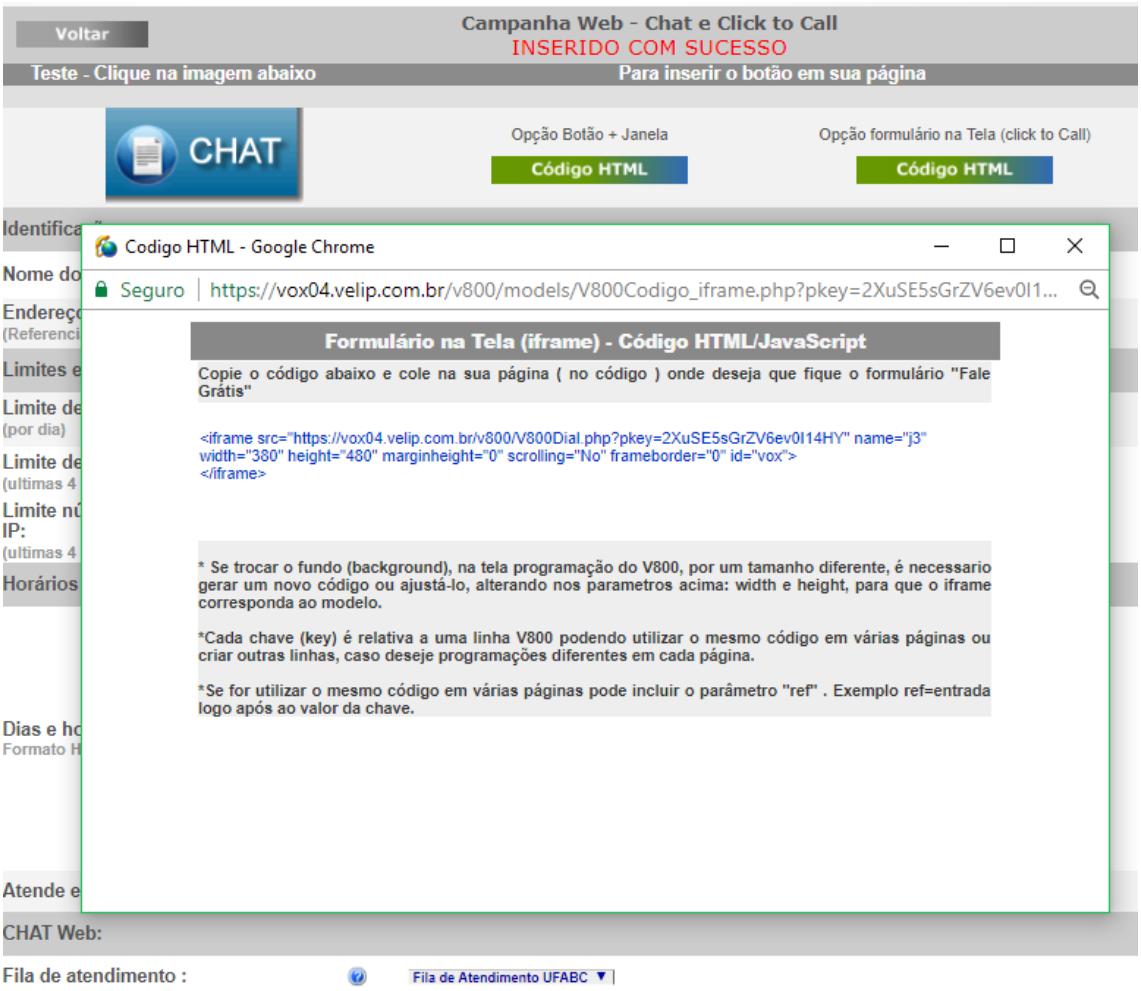


Figura Código HTML inserido no site para atendimento do estudo de caso Velip.

A Figura anterior também mostra a janela aberta após o clique no botão "**Código HTML**", com o código HTML (em azul) do ***iframe*** copiado para a página do site de atendimento ao estudante.

Implantação do código HTML/Javascript no site

O código HTML gerado no passo anterior foi inserido em um site na internet para incorporar a solução da Velip. Esse site é o canal inicial de atendimento ao cliente.

O endereço utilizado para este estudo de caso é o http://brunowiz.ddns.net. Este site foi criado em um servidor na AWS, infraestrutura de *cloud* pública da Amazon. Neste servidor Windows Server 2016 foi habilitado o IIS (*Internet Information Services*), que é o servidor *web* padrão do Windows.

Na página inicial do site foi inserido seguinte código:

```
<iframe
src="https://vox04.velip.com.br/v800/V800Dial.php?pkey=2XuSE5sGrZV6ev0I14HY" name="j3" width="380" height="480" marginheight="0" scrolling="No" frameborder="0" id="vox">
</iframe>
```

Note que o trecho em negrito dentro do "***iframe***" é o código fornecido no passo anterior. Essa página foi salva no diretório C:/inetpub/wwwroot/index.html do servidor, que é o caminho que o IIS aponta como diretório raiz do site.