UNIVERSIDADE ESTADUAL DO MARANHÃO
CENTRO DE CIÊNCIAS TECNOLÓGICAS
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO

DESENVOLVIMENTO DE UM APLICATIVO NAS PLATAFORMAS ANDROID E IOS UTILIZANDO COMUNICAÇÃO SEGURA PARA A SOLUÇÃO VOIP UEMA

MAGNO CASTRO MORAES

SÃO LUÍS - MA
2021

UNIVERSIDADE ESTADUAL DO MARANHÃO
CENTRO DE CIÊNCIAS TECNOLÓGICAS
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO

MAGNO CASTRO MORAES

DESENVOLVIMENTO DE UM APLICATIVO NAS PLATAFORMAS ANDROID E IOS UTILIZANDO COMUNICAÇÃO SEGURA PARA A SOLUÇÃO VOIP UEMA

Trabalho para o Exame de Qualificação apresentado ao curso de Mestrado Profissional em Engenharia de Computação e Sistemas na Universidade Estadual do Maranhão sob orientação do(a) Prof. Dr. Carlos Henrique Rodrigues de Oliveira.

SÃO LUÍS - MA
2021

Moraes, Magno Castro.
Desenvolvimento de um aplicativo nas plataformas de Android e iOS utilizando comunicação segura para a solução VoIP UEMA/ Magno Castro Moraes. – São Luís, 2021.

54 folhas

Dissertação (Mestrado) – Curso de Engenharia de Computação e Sistemas, Universidade Estadual do Maranhão, 2021.

Orientador: Prof. Dr. Carlos Henrique Rodrigues de Oliveira.

1.VoIP UEMA. 2.Softphone. 3.Android. 4.iOS. 5.SIGUEMA. 6.Segurança. I.Título.

CDU: 004.4:621.395

Elaborado por Giselle Frazão Tavares - CRB 13/665

MAGNO CASTRO MORAES

DESENVOLVIMENTO DE UM APLICATIVO NAS PLATAFORMAS ANDROID E IOS UTILIZANDO COMUNICAÇÃO SEGURA PARA A SOLUÇÃO VOIPUEMA

Dissertação apresentada ao Mestrado Profissional em Engenharia de Computação e Sistemas (PECS) da Universidade Estadual do Maranhão, como registro para obtenção do grau de Mestre em Engenharia de Computação e Sistemas.

Trabalho aprovado. São Luís – MA, 31 de março de 2021.

Prof. Dr. Carlos Henrique Rodrigues de Oliveira

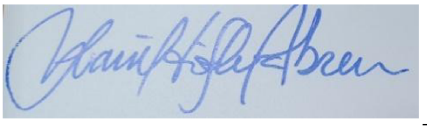
Orientador

Prof. Dr. Luís Carlos Costa Fonseca

Primeiro membro

Prof. Dr. Leonardo Henrique Gonsioroski Furtado da Silva

Segundo membro



Prof. Dr. Ivanildo Silva Abreu

Terceiro membro

AGRADECIMENTOS

Primeiramente aos meus pais Luís Magno Silva Moraes e Delci Mendes Castro, por serem meu alicerce e pelo grande esforço para proporcionar sempre o melhor a mim.

Às minhas irmãs, Maynara Castro Moraes e Hellen Castro Moraes, pelo suporte que sempre me deram.

À minha tia, Linielma Mendes Castro, por sempre me incentivar na minha formação e desenvolvimento.

Aos meus avós Domingas Mendes Castro (In Memória), Leonel Viegas Castro, Luiz Gonzaga Souza Moraes (In Memória) e Margarida Silva Moraes por serem o exemplo de superação e luta que sempre me espelhei.

Ao meu orientador, Professor Dr. Carlos Henrique Rodrigues de Oliveira, pela chance que me deu para participar do projeto VoIP UEMA e por toda motivação que me deu para sempre buscar mais.

Aos meus queridos amigos, do grupo de estudos Gestão C, Luiz Ripardo, Richardson Lima, Rodrigo Frazão e Rodrigo Oliveira, bem como aos meus amigos do curso de Engenharia de Computação, em especial a Flávia Fernandes e Eryck Araújo pelo companheirismo e apoio.

Aos meus amigos e familiares, em especial às minhas primas Luana Nogueira e Daiane Silva, pelos incentivos.

Aos integrantes do projeto VoIP UEMA, pela ajuda e suporte no meu trabalho.

RESUMO

Buscando suprir as deficiências do estado do Maranhão no contexto da tecnologia, bem como solucionar os problemas de comunicação que a Universidade Estadual do Maranhão sofre, foi criado o projeto VoIP UEMA para fornecer um serviço de telefonia com alta qualidade e baixo custo para utilização da comunidade UEMA. Neste trabalho, foi desenvolvido uma ferramenta *freeware* para *smartphones* das plataformas Android e iOS, capaz de realizar chamadas de áudio e vídeo via rede TCP/IP com sucesso, fazendo a autenticação com credenciais do SIGUEMA. Foi analisada a viabilidade do uso de ferramentas disponíveis no mercado e constatado que estes não possibilitam realizar uma autenticação com o sistema SIGUEMA, além de alguns apresentarem certas deficiências e defeitos. A aplicação desenvolvida permite a autenticação junto ao SIGUEMA, através do Web Service VoIP UEMA, a conexão com o servidor de serviços VoIP, Asterisk, com sucesso e a realização de chamadas com qualidade e suprindo os defeitos das aplicações testadas.

Palavras-chave: VoIP UEMA; Aplicativo; Android; iOS; SIGUEMA; VoIP.

ABSTRACT

Seeking to overcome the deficiencies from Maranhão state in the technological area, as well solving the communication problems that State University of Maranhão is currently facing, the project VoIP UEMA was created to provide a telephone service with high quality standards and low cost for the UEMA community to utilize. In this project, a freeware tool for smartphones on Android and iOS platforms, capable of making audio and video calls through TCP/IP network, while authenticating with SIGUEMA credentials. A marketing research has been done looking for similar tools that could perform the same task, but none of the available tools could provide an authentication with SIGUEMA, also, some of those applications showed deficiencies and defects. The developed application allows authentication with SIGUEMA, through the Web Service VoIP UEMA, a connection with VoIP service provider, Asterisk, successfully making high quality calls while filling out the defects of the evaluated applications.

Keywords*: VoIP; Application; Android; iOS; Web Service; VoIP UEMA.*

LISTA DE TABELAS

Tabela 1: Padrões de codificadores de sinais de voz [14] e [31]. 22

Tabela 2: Comparação entre o H.323 e o SIP [10]. 26

Tabela 3: Comparação entre o Flutter e o React Native. 43

LISTA DE ABREVIATURAS E SIGLAS

ACELP	*Algebraic-Code-Excited Linear-Prediction*
ACSE	*American Council on Science and Education*
ADPCM	*Adaptive Differential Pulse Code Modulation*
API	*Application Programming Interface*
ARPANET	*Advanced Research Projects Agency Network*
ATA	*Analog Telephone Adapter*
CS-ACELP	*Conjugate Structure Algebraic-Code-Excited Linear-Prediction*
CSS	*Cascading Style Sheets*
ENIAC	*Electronic Numerical Integrator and Computer*
FPS	*Frames Per Second*
GC	*Garbage Collector*
GSM	*Groupe Special Mobile*
HAL	*Hardware Abstraction Layer*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
ISO	*International Organization for Standardization*
ITU	*International Telecommunication Union*
JSON	*JavaScript Object Notation*
JSX	*JavaScript XML*
JVM	*Java Virtual Machine*
LAN	*Local Area Network*
LDCELP	*Low-Delay Code Excited Linear Prediction*
MeGaCo	*Media Gateway Control*
MOS	*Mean Opinion Score*
MP3	*Moving Picture Experts Group 1 Layer-3*
MP-MLQ	*Multipulse Maximum Likelihood Quantization*
MVP	*Minimum Viable Product*
NDK	*Native Development Kit*
NTI	Núcleo de Tecnologia da Informação
PBX	*Private Branch Exchange*
PCM	*Pulse Code Modulation*
PSTN	*Public Switched Telephone Network*
QoE	*Quality of Experience*
QoS	*Quality of Service*
RFC	*Request for Comments*

RTP	*Real-time Transport Protocol*
RTSP	*Real Time Streaming Protocol*
SDK	*Software Development Kit*
SGBD	Sistema de Gerenciamento de Banco de Dados
SIGUEMA	Sistema Integrado de Gestões Acadêmicas da UEMA
SIP	*Session Initiation Protocol*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
UDP	*User Datagram Protocol*
UEMA	Universidade Estadual do Maranhão
UI	*User Interface*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
UTF-8	*8-bit Unicode Transformation Format*
VoIP	*Voice over Internet Protocol*
W3C	*World Wide Web Consortium*
WAN	*Wide Area Network*
WebRTC	*Web Real-Time Communications*
Wi-Fi	*Wireless Fidelity*
WS	*WebSocket*
WSS	*WebSocket Secure*
XML	*eXtensible Markup Language*

SUMÁRIO

INTRODUÇÃO **12**
Objetivos 14
Objetivo Geral 14
Objetivos Específicos 14
Metodologia 14
Justificativa 14
Aplicabilidade 15
Trabalhos relacionados 15
Estrutura do documento 15
FUNDAMENTAÇÃO TEÓRICA **16**
Voz sobre IP 17
Vantagens 18
Desvantagens 19
Arquitetura e Funcionamento do VoIP 20
Codecs e Qualidade de Áudio 21
Protocolos 23
SIP 24
Arquitetura do SIP 27
Sinalização 28
Cabeçalho SIP 31
RTP 32
Cabeçalho RTP 34
WebRTC 36
SDP 37
Arquitetura Limpa 37
DESENVOLVIMENTO **40**
Pesquisa de ferramenta para desenvolvimento multiplataforma 41
Mudança do PJSIP para o dart-sip-ua 44
Projeto Flutter-WebRTC 44
Projeto dart-sip-ua 45
Desenvolvimento do aplicativo 46
Submissão de paper 49
Certificado de Registro de Programa de Computador 49
CONCLUSÃO E TRABALHOS FUTUROS **50**
REFERÊNCIAS **51**

1. INTRODUÇÃO

Desde os primórdios o homem anseia pela comunicação e, principalmente, de forma fácil e praticamente instantânea. Uma das grandes tecnologias que atendeu a essa demanda foi a criação do telefone. Este está presente em diversos lares, empresas e organizações possibilitando uma comunicação eficiente, deixando as pessoas mais próximas e informadas sobre o mundo.

Em paralelo à telefonia, tem-se também o desenvolvimento das redes de dados que tem crescido de forma exponencial com o auxílio da evolução dos computadores e da aplicação de novas tecnologias, com destaque ao TCP/IP (*Transmission Control Protocol/Internet Protocol*) que é atualmente o modelo utilizado pela rede mundial conhecida como Internet. É inevitável a abrangência da telecomunicação em diversas áreas de serviços, isso permite que cada vez mais tecnologias estejam conectadas à rede de dados e dentre esses serviços temos a telefonia sobre IP (*Internet Protocol*) ou VoIP (*Voice over Internet Protocol*) que apesar da sua fama recente, surgiu em meados dos anos 90 quando a tecnologia da época não permitia um bom desempenho dessa aplicação, mas com o atual cenário das telecomunicações o VoIP voltou a ter destaque e passou a ser alvo de várias empresas de serviços de comunicação.

O VoIP se trata da transmissão da voz, digitalizada e comprimida em forma de dados, através de pacotes por uma rede IP. Além da utilização de codecs e dos protocolos do TCP/IP, faz-se necessário a utilização de protocolos para sinalização, capaz de estabelecer, administrar e finalizar as sessões de comunicações, bem como para transmissão do áudio entre os usuários. Existem vários protocolos com essas finalidades, no entanto, destaca-se o protocolo SIP (*Session Initiation Protocol*) para a sinalização, RTP (*Real-time Transport Protocol*) para transmissão de mídia e o WebRTC (*Web Real-Time Communication*), que possibilita, também, o tráfego de mídia como áudio, vídeo e texto, sendo geralmente usado por navegadores.

Além dessas tecnologias, também vale destacar a dependência do *software* em que vai se estabelecer as chamadas VoIP, pois este é quem vai ser o intermediário entre o usuário e o serviço, além de ser um importante agente na qualidade e disponibilidade da aplicação. Os *smartphones* são dispositivos móveis que tiveram uma grande evolução desde o início deste século. Eles estão presentes na vida de várias pessoas no mundo e são ferramentas poderosas de comunicação, pois possuem capacidade de embarcar sistemas operacionais e possuem uma

boa base para o desenvolvimento de *software* de forma fácil e eficaz. No mercado dos *smartphones*, encontramos vários aplicativos que implementam o serviço VoIP, bastando apenas configurar as credenciais de um servidor VoIP conhecido.

Este tipo de serviço é bem útil para organizações, principalmente àquelas que possuem uma estrutura de redes estabelecida. A implementação deste serviço na Universidade Estadual do Maranhão (UEMA) seria de grande utilidade, pois traria uma grande economia para a universidade, substituindo a telefonia convencional, além de atender aos setores onde nem mesmo possuem acesso a telefonia convencional, comprometendo a eficiência de seus trabalhos. Mas a configuração e distribuição de ramais desta aplicação para autenticação no servidor VoIP da UEMA se tornaria uma tarefa trabalhosa e, provavelmente, pouco aceitável.

Almejando essa aceitação e uso pela universidade, foi idealizada a integração deste serviço com o sistema acadêmico da própria universidade, o SIGUEMA, através de um *web service,* tornando a autenticação única.

Todavia, os aplicativos disponíveis no mercado se demonstraram insuficientes em diversos aspectos. Primeiramente, nenhum deles é capaz de realizar essa integração, pois só realizam a comunicação com um servidor VoIP, não permitindo a interação com o *web service*. Além do mais, temos que os aplicativos pagos representam um custo que nem todos os usuários estariam dispostos a pagar e os aplicativos *freewares* testados também apresentam algumas características negativas particulares, como uma interface pouco amigável, complexidade de configuração das credenciais e a pouca personalização.

Com isso, nasceu o projeto VoIP UEMA com intuito de suprir essas carências da universidade. Além do mais, visa reduzir ao máximo os custos econômicos da UEMA utilizando diversas ferramentas *freeware*. Este projeto conta com a seguinte equipe:

- Magno Castro Moraes – Mestrando em Engenharia de Computação, responsável pelo desenvolvimento da aplicação móvel;
- Luiz Ricardo Souza Ripardo – Graduado em Engenharia de Computação, responsável pelo desenvolvimento do Web Service VoIP UEMA;
- Prof. Dr. Carlos Henrique Rodrigues de Oliveira – Professor do curso de Engenharia de Computação e o idealizador do projeto;

O projeto já se encontra operante na UEMA desde 2018, possuindo um aplicativo para Android publicado na *Play Store* e um discador *Web*, ambos são capazes de autenticar com o sistema acadêmico SIGUEMA com sucesso e de forma segura.

1.1 Objetivos

1.1.1 Objetivo Geral

Contribuir no desenvolvimento do aplicativo VoIP UEMA, utilizando o *framework* Flutter para desenvolvimento em Android e iOS capaz de realizar chamadas de áudio e vídeo com segurança, aplicando um teste de qualidade de experiência de usuário QoE (*Quality of Experience*) com base na qualidade da chamada de voz e vídeo.

1.1.2 Objetivos Específicos

- Desenvolver o aplicativo utilizando o *framework* multiplataforma Flutter;
- Realizar chamadas de áudio e vídeo;
- Realizar comunicação criptografada com os servidores;
- Teste de qualidade QoE com base na qualidade de voz e vídeo.

1.2 Metodologia

Inicialmente, foi realizada uma pesquisa sobre os *frameworks* de desenvolvimento *mobile* multiplataforma (Android e iOS) disponíveis no mercado que viessem a atender as necessidades específicas do projeto, realizando o estudo sobre as ferramentas encontradas para avaliar a melhor opção. O candidato escolhido foi o Flutter.

Em seguida, passou-se a desenvolver a aplicação utilizando o Flutter para realizar chamadas VoIP através de dispositivos Android e iOS. Primeiramente, foi realizada uma série de testes e validações das ferramentas utilizadas para determinar a estruturação do projeto.

Após isso, foi desenvolvido uma interface amigável para realização da autenticação junto ao serviço SIGUEMA, através do Web Service VoIP UEMA, e a realização de chamadas de áudio e vídeo.

1.3 Justificativa

Sanar a necessidade da comunidade da Universidade Estadual do Maranhão quanto aos problemas de comunicação entre seus membros, promovendo a inclusão dos usuários na solução, além de trazer economia para a universidade.

1.4 Aplicabilidade

O projeto pode ser aplicado na UEMA, no projeto VoIP UEMA já vigente contribuindo com a versão do aplicativo *softphone* para iOS e, também, possibilitar a realização de videochamadas para os dispositivos móveis. Pode ser também aplicado a outras comunidades acadêmicas, bem como instituições públicas ou privadas.

1.5 Trabalhos relacionados

Existem diversas soluções VoIP para empresas e universidades. Os trabalhos encontrados realizam a implementação de projetos VoIP com a instalação de PBXs, geralmente o Asterisk, com conexão a telefones IPs e a utilização de *softphones* disponíveis no mercado como X-Lite, como mencionado em [1] e [2]. Porém, não apresentam autenticação junto ao sistema acadêmico da universidade e não possuem *softphones* próprios para dispositivos móveis e navegadores, sendo assim o projeto VoIP UEMA traz inovação para a Universidade Estadual do Maranhão.

1.6 Estrutura do documento

O trabalho está estruturado com o primeiro capítulo tratando da introdução do projeto. Em sequência, no segundo capítulo é apresentado toda a fundamentação teórica utilizada como base para o desenvolvimento deste projeto, bem como definições sobre a rede de dados. Já o terceiro capítulo traz todo o processo de desenvolvimento e atendimento aos objetivos propostos. Por fim, o capítulo quatro traz a conclusão do projeto e os trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Apesar deste projeto tratar de um assunto recente, do início deste século, a história envolvida nas tecnologias utilizadas é bem mais antiga e não há como falar de telefonia sem citar o grande nome desta área: Alexander Graham Bell que, em meados do ano de 1876, concluiu um trabalho de anos e criou o primeiro telefone [3].

Também temos o importante surgimento do primeiro computador digital eletrônico, o ENIAC que começou a ser desenvolvido em 1943 ainda durante a II Guerra Mundial, mas que só teve seu trabalho concluído em 1946. Esta foi uma criação dos pesquisadores da empresa Eletronic Control Company [4].

A partir do surgimento do ENIAC (*Electronic Numerical Integrator and Computer*), os avanços em torno dessa recente tecnologia tiveram um aumento exponencial e logo fez-se necessária a interconexão entre essas máquinas em grandes áreas. E este foi o objetivo do departamento de defesa americano no fim dos anos 60, onde se iniciou uma pesquisa buscando interligar todas as redes de computadores militares espalhados pelos Estados Unidos, mas tinha um grande desafio pela frente: as diferentes tecnologias proprietárias utilizadas nessas redes [4].

As pesquisas levaram ao surgimento do ARPANET (*Advanced Research Projects Agency Network*) com protocolos muito semelhantes ao do atual modelo TCP/IP. Em seguida o TCP/IP se tornou o padrão da Internet, pois permite que as informações sejam entregues a qualquer custo mesmo que chegue com atraso [5].

Quanto a transmissão de dados em tempo real, como áudio e vídeo, é necessária rapidez, porém o TCP (*Transmission Control Protocol*) é um protocolo orientado a conexão e isso acaba diminuindo o tempo que cada pacote chega no receptor, então como atender a esse requisito? A resposta é utilizar o UDP (*User Datagram Protocol*). O UDP é um protocolo de comunicação integrante do sistema TCP/IP que, ao contrário do TCP, não é orientado à conexão (protocolo sem conexão) e, portanto, não garante a entrega ou integridade dos dados transmitidos. No entanto, o UDP permite que as informações possam ser transmitidas com maior rapidez, um pacote atrás do outro, num efeito chamado “rajada de dados” e que, dentro de uma rede com pouca perda de pacotes, alta velocidade de transmissão e pequeno atraso na entrega dos pacotes, permitirá a transmissão de voz (ou vídeo) em tempo real [6].

2.1 Voz sobre IP

O VoIP apareceu no ano de 1995, em Israel, quando a empresa VocalTec Communications desenvolveu um sistema que permitisse utilizar os recursos multimídia de um PC doméstico para iniciar conversas de voz através da Internet [7]. O *softphone*, um *software* capaz de realizar chamadas VoIP, era o *Internet Phone Software* que era executado em um computador 486/33 MHz possuindo uma placa de som, alto-falantes, microfone e modem que devido às tecnologias da época possuía uma péssima qualidade, com atrasos e os famosos "cortes" na voz, no entanto representou o primeiro telefone por IP, precursor dos atuais *softphones*, como Skype e Zoiper, e foi o primeiro passo para que pesquisadores se interessassem neste tipo de aplicação [5].

Em 1998, o VoIP deu um grande passo com o desenvolvimento dos *gateways* que são equipamentos capazes de conectar aparelhos telefônicos convencionais ou centrais telefônicas à rede de dados possibilitando a comunicação destes com os sistemas VoIP. Inicialmente as conexões eram feitas entre computador e telefone, mas logo pôde se conectar dois telefones convencionais. Mais tarde foi desenvolvido *gateways* especializados e dispositivos chamados de ATA (*Analog Terminal Adapter*) que possibilitaram a interligação entre dois sistemas convencionais, ou PABX (*Private Branch Exchange*) [5].

Por volta do ano 2000 algumas empresas, como a Nortel e a Cisco, desenvolveram *hardwares* para a telefonia VoIP. Isso tirou a carga dessa aplicação dos processadores dos usuários e passou a ser realizadas por estes *hardwares* e, desta forma, o VoIP ficou menos dependente das capacidades do computador usuário permitindo uma melhor acessibilidade a esta tecnologia fazendo com que diversas empresas começassem a implementá-la em suas redes internas [8].

O VoIP é uma tecnologia que digitaliza, comprime e converte sinais de voz (analógico) em pacotes de dados e os transmite utilizando qualquer rede TCP/IP. Assim que estes pacotes são recebidos no destino, são convertidos em sinal analógico novamente, utilizando qualquer meio no qual seja possível reproduzir o som [9].

Quanto a rede, o VoIP trafega em qualquer rede de dados, desde que use os protocolos do modelo TCP/IP. Desta forma, um sistema VoIP pode ser implementado em quaisquer redes que usem o protocolo IP: Internet, intranets e redes locais [10].

A rede não tem conhecimento do tráfego da voz, ela foi projetada apenas para permitir o fluxo de pacotes de dados de uma ponta a outra através dela, no entanto os

conteúdos destes pacotes são conhecidos pelas aplicações que as enviaram e/ou receberam. Assim, as interfaces do VoIP tratam de receber a voz, digitalizá-la, comprimi-la e acomodá-la em pacotes de dados idênticos aos que trafegam pela rede. Desta forma, os elementos da rede como *switches*, *hubs*, interface de VoIP, roteadores e etc são responsáveis apenas por conduzir esses pacotes até o destino onde serão convertidos novamente para a voz pela aplicação que a recebe, sendo assim um processo totalmente transparente ao usuário [7].

2.1.1 Vantagens

O VoIP é uma tecnologia muito útil e com uma grande capacidade de difusão, de modo que pode servir tanto ao usuário comum que está tentando se comunicar com um parente distante, por exemplo, quanto à uma grande empresa em que seus funcionários utilizam dessa tecnologia para passar ou receber informações, realizar solicitações diversas etc. Essa é uma das grandes características desta aplicação que permitiu seu impulsionamento no mundo.

Além dessa, tem-se também uma das mais importantes e agradáveis vantagens: os fatores financeiros. É de conhecimento geral o alto custo econômico da telefonia tradicional, ligações de longa duração e ligações internacionais são muito custosas aos usuários que têm que controlar o consumo ao máximo. Segundo [11] “Nos EUA uma chamada de longa distância custa em média U$0,0171 o minuto. Com esse valor a Bell e suas subsidiárias faturam em média 8 bilhões de dólares por ano”. Desta forma, para uma empresa que já possui uma infraestrutura de rede IP consolidada e conectada aos diversos pontos para onde realiza as ligações, o uso do VoIP no lugar da telefonia convencional reduz o custo em 100% em relação a telefonia, visto que as chamadas passam a ser dados na rede IP [12].

Há redução de custo também em relação aos equipamentos, pois utiliza-se computadores ou *smartphones* para efetuar as ligações e, pela concorrência intensa desses dispositivos que causam uma queda nos preços e pelo fato da aplicação VoIP exigir pouco das características do *hardware*, esses são facilmente acessíveis. Em contrapartida, os equipamentos da telefonia convencional são caros devido à baixa demanda e geralmente possuem tecnologia proprietária que causa uma dependência ao fornecedor. Esta característica do VoIP favorece principalmente usuários comuns e empresas de pequeno porte [12].

O VoIP também apresenta várias vantagens devido às suas características frente a rede. Primeiramente ela é executada sobre a mesma infraestrutura utilizada para acesso à

Internet. Desse modo, devido a digitalização, a voz compartilha o mesmo meio com dados convencionais sem a necessidade de fazer alterações na rede, apenas implementar e utilizar. Essa integração de voz e dados permite que um usuário faça chamadas de voz, enquanto acessa à rede para enviar um *e-mail*, o que não é possível através da telefonia convencional, pois utiliza comutação de circuitos e assim só pode realizar uma de cada dessas ações por vez. Além disso, apresenta melhores aspectos de segurança e confiabilidade já que são obtidos da própria rede IP [11].

Também se trata de uma aplicação que independe da camada de Acesso à Rede, pois faz uso do protocolo IP e este realiza as devidas medidas para os diversos protocolos da camada subjacente. Assim, essa tecnologia pode ser implementada tanto para LAN (*Local Area Network*) como para WAN (*Wide Area Network*), fazendo uso da Ethernet ou uma rede *wireless* qualquer que utilize o protocolo IP tornando, assim, possível o uso por parte dos *smartphones*.

A largura de banda utilizada pela telefonia IP é bem reduzida, isso porque as diversas tecnologias de compressão de voz possibilitam a transmissão a 32 kbps, 16 kbps, 6,3 kbps ou 5,4 kbps. Em razão disso, pode ser realizada múltiplas ligações através de uma mesma banda simultaneamente sem ocorrer interferência entre elas. Já na telefonia convencional é realizada uma conexão ponto-a-ponto exclusiva para a ligação a uma taxa fixa de 64 kbps sem permitir que outras ligações sejam feitas através dessa banda [12].

Além disso, de acordo com [13] temos que “hoje, a combinação de VoIP com as redes sem fio, que permitem acesso à Internet em qualquer lugar e a qualquer momento, tem se tornado uma forte rival para a telefonia tradicional”.

Devido a todas essas vantagens, a telefonia sobre IP também é um meio de grande lucro e, consequentemente, de geração de empregos para profissionais especializados na área [11].

2.1.2 Desvantagens

Apesar de todas as vantagens apresentadas, a telefonia sobre IP ainda tem muitos desafios a serem vencidos para vir a substituir a telefonia convencional. Dentre elas, temos a dificuldade de integração entre os protocolos de sinalização que existem para controle da sessão multimídia. Existem muitos desses protocolos, porém os mais famosos e utilizados são o H.323, criado pela ITU-T (*Telecommunication Standardization Sector of International*

Telecommunication Union), e o SIP, desenvolvido pela IETF (*Internet Engineering Task Force*) que, embora realizem a mesma tarefa, diferem quanto ao modo de operação e isto dificulta a interoperabilidade [12].

Além disso, temos também um problema ainda pertinente nas redes IP que é a confiabilidade e disponibilidade. Este problema tem um grande impacto sobre serviços que dependem da rede IP, como o VoIP, pois um sistema de comunicação que não tem um bom percentual de disponibilidade e confiabilidade não se torna aceitável e quanto a isso, a rede PSTN (*Public Switched Telephone Network*) ainda se encontra muito à frente da rede IP, isto porque a disponibilidade da rede de telefonia convencional é de 99,999%. São cerca de 5,26 minutos de indisponibilidade da rede por ano [14], o que está longe ainda para as redes IP e, consequentemente, para o sistema de telefonia sobre IP. Embora a rede de dados ainda apresente diversas redundâncias e outros mecanismos para manter a confiabilidade e disponibilidade da rede, ela continua sujeita a problemas como falhas na rede elétrica e ciberataques, pois está sujeita às mesmas vulnerabilidades de qualquer serviço comum da rede IP.

Por fim, tem-se os desafios quanto à qualidade da voz. Ao realizar a conversão, a voz passa a sofrer dos mesmos problemas que os dados, porém não é permitido que esses problemas venham a interferir de tal forma que o sistema se torne ineficiente, afinal ninguém irá adotar um sistema de comunicação a distância onde as vozes que chegam até os usuários são distorcidas ou cortadas. Com isso, há um limite aceitável para que a qualidade da voz não prejudique o sistema VoIP e para isso são procurados meios de melhorar este fator através de desenvolvimento de codecs, protocolos e equipamentos [11].

2.1.3 Arquitetura e Funcionamento do VoIP

Para que o sistema de telefonia IP funcione, é essencial obedecer à arquitetura padrão do VoIP. Porém, antes de ocorrer a transmissão da voz para o receptor, é preciso primeiro converter a voz, que é um sinal analógico, em dados, que é um sinal digital. Isso é possível através de codificadores de voz (codecs) que participam do processo ilustrado na Figura 1.

Figura 1: Transmissão da voz. Fonte: Autor.

Após o processo de transformação da voz em dados, estes dados ficam armazenados em um *buffer* até que possam ser enviados. Antes de serem enviados, é preciso acrescentar cabeçalhos fundamentais para que a conexão entre os clientes ocorra normalmente e, após isto, é permitida a transmissão através da rede comutada de dados.

Ainda, segundo [15], as etapas de uma chamada básica, feita através de um sistema de telefonia IP, são as seguintes:

1. O originador da chamada retira o monofone e ouve um tom de discagem;
2. O originador da chamada digita o número ou ramal do telefone, o qual será mapeado para o endereço de IP do receptor;
3. Protocolos de sinalização de chamada são utilizados para localizar o receptor e enviar um sinal para produzir um toque;
4. O telefone de destino toca, o que indica ao receptor uma chamada entrante;
5. O destinatário da chamada retira o monofone do gancho e inicia uma conversa de duas direções. A mensagem de áudio é codificada usando um codec e é transmitida por meio da rede IP usando um protocolo de transporte de voz (RTP, por exemplo);
6. A conversa termina, ocorre o encerramento da chamada, a faturação é realizada.

Nota-se que há uma abstração de diversas tecnologias por trás de todo o sistema, como a topologia da rede, o servidor fornecedor do serviço VoIP e etc.

2.2 Codecs e Qualidade de Áudio

Os processos de conversão do som em *bits* e vice-versa é feito, em ambos os casos, com utilização de amostras do sinal. A amostragem do sinal analógico determina a qualidade do áudio. Se a taxa de amostragem for baixa, não vai se obter um som tão fiel ao original devido aos poucos valores assumidos em cada amostra, porém se a taxa for elevada

irá consumir um espaço significativo de armazenamento. Caso seja transmitido, também custará mais largura de banda, além de requisitar um maior tempo de transmissão [16].

Para solucionar esses problemas, é utilizado o codec que, podendo ser tanto um *hardware* quanto um *software*, é capaz de realizar a compressão e descompressão das amostras de voz coletadas. A utilização desta ferramenta é indispensável em aplicações para transmissão de áudio em tempo real, tal como o VoIP [16].

Existem atualmente diversos codecs disponíveis, cada um com suas vantagens e desvantagens que determinam em qual ambiente é melhor adotado. O ITU-T fez a padronização de diversos destes codecs e alguns dos principais são mostrados na Tabela 1.

Tabela 1: Padrões de codificadores de sinais de voz [14] e [31].

Recomendação/Nomenclatura	Codificação	Taxa de *bits* (kbps)	Ano de publicação
G. 711	PCM	64	1972
G. 726	ADPCM	40, 32, 24, 16	1990
G. 728	LD-CELP	16	1992
G. 729	CS-ACELP	8	1996
G. 723.1	MP-MLQ	6,3	1996
G. 723.1	ACELP	5,3	1996
Opus	OPUS	6 ~ 510	2012

Dos codecs apresentados na Tabela 1, temos:

- G.711: É um codec básico de código livre que utiliza a modulação de pulso (PCM) das frequências de voz. Este realiza a amostragem de 8.000 amostras por segundo, onde cada uma dessas amostras é armazenada em 8 *bits*, exigindo uma largura de banda para transmissão de 64 kbps [14]. Porém, trata-se de um codec de fácil implementação, com atraso de 0,125 ms (tempo de codificação de 1 *bit*) e uma qualidade de áudio de 4,5 a 5 no MOS (*Mean Opinion Score*) [17];
- G.723.1: Este já é um codec que exige licença para o uso. O G.723.1 exige uma largura de banda de 6,3 kbps e 5,3 kbps, com atraso de 37,5 ms oferecendo uma alta compressão, com alta qualidade (MOS 3,9). Entretanto, tem um alto custo de processamento [17];

- G.726: Outro codec que é *freeware*. É capaz de transmitir a 32 kbps, exigindo a metade do G.711 e apresenta cerca de 4,1 no MOS, porém não é aplicável para a transmissão de músicas e/ou outros efeitos sonoros [14];
- G.729: Exige licença para o uso. Possui uma taxa de transmissão de 8 kbps. Com quadro de 10 ms e um MOS pouco acima de 4,0 [18].
- Opus: Padronizada pela RFC 6716 (*Request For Comments*), trata-se de um codec interativo de áudio *open-source royalty-free* cuja aplicação pode ir do VoIP até a transmissão de música ao vivo com qualidade. Isso é possível devido a sua natureza interativa que se adapta às condições do meio de transmissão (como a rede de Internet), podendo adotar uma taxa de amostragem de 8 kHz, para estabelecer conversas de voz (*mono*), até 48 kHz, provendo transmissão de música de alta qualidade (*stereo*). Além disso, pode variar, também, na taxa de transmissão de *bits* (6 kbps a 510 kbps) e tamanho de quadro (2,5 ms a 60 ms) [31].

Além dos codecs, outros fatores determinam a qualidade da voz transmitida em uma rede. Segundo [11], temos os principais:

- Atraso: Determina o intervalo de tempo entre a transmissão e o recebimento do pacote de áudio;
- *Jitter*: Se trata da variação do atraso. Isso é causado quando os pacotes passam por elementos de rede diferentes, causando a desordenação dos pacotes. Estes pacotes precisam ser armazenados para, em seguida, fazerem a reordenação;
- Perda de pacotes: A comunicação de voz aceita uma perda de até 5%. Superior a isso, a qualidade da voz é comprometida com cortes na transmissão.

2.3 Protocolos

Os protocolos sempre estão presentes em qualquer tipo de comunicação realizada pela rede. De acordo com [19] temos que “um protocolo define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de uma mensagem ou outro evento”.

Os diversos protocolos de redes são organizados em pilhas ou camadas que permitem a comunicação correta entre uma camada “x” de uma máquina com a camada “x” de outra máquina. No entanto, essa comunicação não é realizada de forma direta, na verdade cada camada realiza seus processos e passa a informação para a camada imediatamente abaixo

dela até chegar ao meio físico por onde trafegam os dados. Ao chegar no destino, o pacote desloca-se da primeira à última camada, sendo descapsulado em cada uma permitindo assim a comunicação entre os protocolos, como podemos ver na Figura 2, onde se tem duas máquinas que se comunicam utilizando um modelo com quatro camadas [16]:

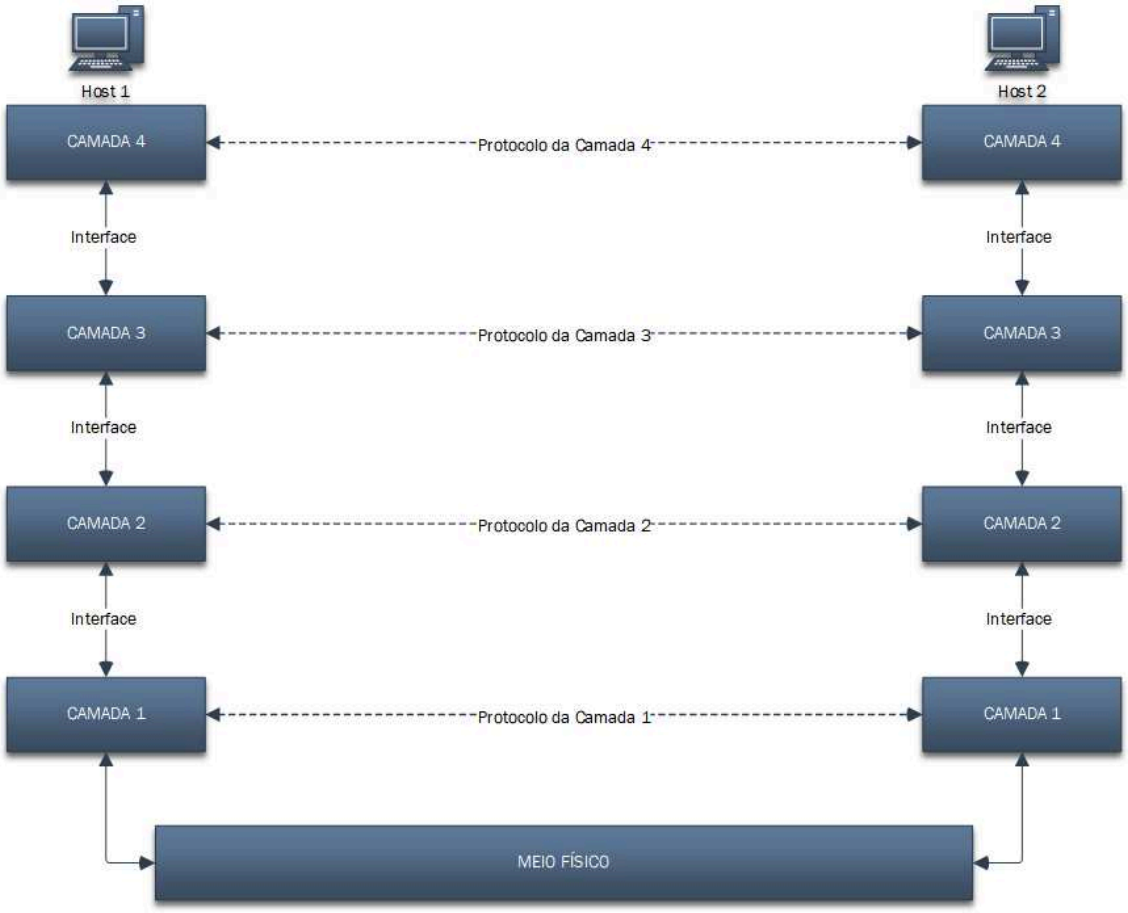


Figura 2: Comunicação entre protocolos. Fonte: Autor.

Com o aumento da demanda das aplicações multimídia em tempo real, fica clara a necessidade de se desenvolver protocolos que padronizem essas aplicações. Devido a isso, grandes organizações criadoras de padrões, tais como o IETF e o ITU-T, se ocupam há anos com a padronização destas aplicações de modo que desenvolveram protocolos que atualmente estão difundidos nos mais variados sistemas interativos em tempo real. São eles o H.323 (ITU-T), SIP, WebRTC, RTP e SDP (IETF) [19].

2.3.1 SIP

O SIP (*Session Initiation Protocol*) se trata de um protocolo de sinalização de código aberto para criação, modificação e finalização de sessões multimídias, como chamadas

de áudio ou vídeo, composto por um ou mais participantes. Foi definido na RFC 2543 [20] em 1999 e revisado na RFC 3261 [21] em 2002 pelo grupo de trabalho MMUSIC (*Multiparty Multimedia Session Protocol*) do IETF [6].

O objetivo do SIP é criar, modificar e terminar sessões entre os usuários, onde estas podem ser *unicast* (ponto a ponto) e *multicast* (conferência) contendo qualquer tipo de tráfego multimídia. Para fazer o controle das sessões, o SIP é capaz de iniciar e encerrar uma chamada, incluir ou excluir participantes de uma sessão e ainda oferecer transferência/manutenção de ligações e transição entre conexões ponto a ponto e conferência [6].

Este protocolo pode ser facilmente integrado em aplicações da Internet por ser um protocolo de requisição-resposta, sendo bastante semelhante ao protocolo HTTP (*Hypertext Transfer Protocol*), implementando uma arquitetura cliente-servidor, fazendo uso de URIs (*Uniform Resource Identifier*). Também tem semelhanças com o SMTP, como por exemplo, a utilização e reuso dos cabeçalhos "*From*", "*To*" e "*Date*".

Segundo [22], este protocolo deve possuir algumas capacidades, tais como:

- Localização de usuários: Determina a sua localização na rede e a viabilidade para se comunicar;
- Capacidades do usuário: Determinar a mídia e os parâmetros utilizados por um ou mais usuários como, por exemplo, identificar os codecs disponíveis, se transmite áudio ou só recebe e etc;
- Disponibilidade do usuário: Avalia a disponibilidade do usuário para participar de uma sessão, se está disponível ou "*online*", se já está em uma sessão com outro usuário e etc;
- Configuração de chamada: Estabelecer a chamada em ambos os lados da comunicação;
- Manipulação da chamada: Manipular a chamada podendo transferir ou terminar a mesma.

Há algumas observações em relação ao protocolo SIP, segundo [21] são elas:

- O SIP não é um sistema de comunicações totalmente integrado, podendo ser utilizado com outros protocolos do IETF para construir uma arquitetura multimídia mais completa, tais como a utilização do SDP (*Session Description Protocol*) para fazer uma descrição das propriedades das sessões multimídias; RTP (*Real Time Protocol*)

para transmissão dos dados em tempo real fornecendo um *feedback* de QoS (*Quality of Service*); WebRTC (*Web Real-Time Communication*) para a transmissão em tempo real de mídias em *browsers*; e o MeGaCo (*Media Gateway Control*), ou H.248, para controlar as portas de entrada para a PSTN. Assim, é preciso a implementação destes protocolos em conjunto ao SIP para o fornecimento de aplicações como o VoIP, no entanto o funcionamento do protocolo SIP não depende destes;

- O SIP fornece primitivas que podem ser utilizadas em diversos serviços;
- O SIP não possui a capacidade de reserva dos recursos da rede, desta forma as mensagens e sessões estabelecidas pelo SIP podem ser transmitidas através de diferentes redes;
- O SIP trabalha sobre primitivas de segurança realizando a criptografia, requisitando autenticação (seja de cliente para *proxy* ou usuário) e fornece serviços de privacidade;
- O SIP é capaz de trabalhar tanto com o IPv4 quanto o IPv6.

Em uma breve comparação do SIP com o H.323, percebemos a vantagem do SIP e a motivação para adotá-lo neste projeto:

Tabela 2: Comparação entre o H.323 e o SIP [10].

Característica	H.323	SIP
Projetado por	ITU	IETF
Compatibilidade com a telefonia tradicional	Grande	Maior
Compatível com a Internet	Não	Sim
Arquitetura	Monolítica	Modular
Negociação de parâmetros	Sim	Sim
Transporte de Informação	RTP/RTCP	RTP/RTCP
Endereçamento	Número de host ou telefone	URL
Segurança com criptografia	Sim	Sim
Implementação	Grande e complexa	Moderada
Tamanho do documento de padrões	1.400 páginas	269 páginas
Estado atual	Tornando-se menos utilizado em VoIP, mas ainda disponível	Tornando-se mais utilizado em VoIP, em expansão

2.3.1.1 Arquitetura do SIP

Uma rede SIP é composta de quatro entidades lógicas do SIP. Cada uma dessas entidades possui uma tarefa específica, participando da comunicação SIP fazendo solicitações de pedido (cliente) ou respondendo aos pedidos (servidor), até mesmo os dois ao mesmo tempo. A seguir, temos a descrição dos componentes que fazem parte da arquitetura SIP:

- O UA (*User Agent*) é a ponta das comunicações multimídias, responsável por requisitar a iniciação e término das conexões através de trocas de pedidos e respostas. É composto pelo UAC (*User Agent Client*) e pelo UAS (*User Agent Server*). O UAC fica responsável por iniciar as chamadas enviando as requisições e o UAS é responsável por responder às chamadas requisitadas enviando as respostas. Ou seja, se uma aplicação inicia um pedido, ele é um UAC durante toda aquela transação, se o mesmo recebe e responde a um pedido, ele é um UAS durante toda aquela transação [6];
- O *Proxy Server* é um servidor intermediário do SIP, que pode atuar também como cliente e servidor. É responsável por redirecionar as requisições e respostas SIP, sinalizando como se fosse o originador da chamada e assim que recebe a resposta redireciona para o originador real. Pode também traduzir endereços, reescrever mensagens, aplicar regras de segurança e armazenar as informações das transações que passam por ele. Um pedido pode ser roteado por diversos *proxys*, mas é importante que o retorno da mensagem siga o mesmo caminho, isso não é um problema quando usa o TCP, porém para o UDP o SIP possui um cabeçalho chamado "*Via*" com este objetivo [6]. Ele também pode replicar a requisição, enviando cópias para diversos *Proxys Servers* de tal forma que procure em diversas localizações ao mesmo tempo e o primeiro que responder é conectado com o cliente que fez a requisição;
- O *Redirect Server* é um servidor responsável por fornecer a um UA, como resposta a uma requisição, a lista de endereços possíveis para alcançar o cliente identificado no pedido, porém ele não reencaminha o pedido. Assim, a função principal desse servidor é realizar o roteamento de chamadas determinando o caminho para completá-la, fazendo uso de programas ou consultando banco de dados [8];

- O *Registrar Server* é um servidor que recebe e aceita as mensagens do tipo *REGISTER* e armazena a localização desse usuário em um banco de dados [6], além de compartilhar essas informações com outros servidores.

2.3.1.2 Sinalização

O SIP utiliza a sintaxe do HTTP/1.1, descrita na RFC 2068 [23] e o conjunto de caracteres utilizados é o ISO 10646 (*International Organization for Standardization*) com codificação UTF-8 (*8-bit Unicode Transformation Format*), presente na RFC 2279 [24]. Como se trata de um protocolo que utiliza a arquitetura cliente/servidor, ele define dois tipos de mensagens: *REQUEST* e *RESPONSE* [6].

As mensagens do tipo *REQUEST* são mensagens enviadas do cliente ao servidor. De acordo com [8], alguns desses métodos são:

- *REGISTER*: Utilizado quando um cliente quer registrar no servidor SIP o "alias" (vulgo) do seu endereço e assim o servidor guarda informações da identificação e localização daquele cliente;
- *OPTIONS*: Faz uma pergunta sobre as capacidades e disponibilidade das funcionalidades do receptor (cliente e servidor) que é descrito no cabeçalho "*To*". A *RESPONSE* contém uma listagem dos métodos, extensões, e codecs suportados;
- *INVITE*: É um convite. O usuário envia uma solicitação a outro usuário para que inicie uma sessão ou participe de uma já existente. O corpo da mensagem pode conter uma descrição da sessão, utilizando-se o protocolo de descrição de sessão SDP. Quando esse método é utilizado em uma sessão já em progresso, é chamado de *Re-INVITE* e geralmente são usados para mudar parâmetros da sessão;
- *ACK*: É uma confirmação de recebimento do *RESPONSE* de um *INVITE*, ou seja, quando o usuário manda um *INVITE* a um destinatário, este retorna uma *RESPONSE* a quem fez o pedido e assim o usuário envia uma mensagem *ACK* confirmando que recebeu o *RESPONSE*. Caso o *INVITE* não possua o SDP, o *ACK* deve possuir;
- *BYE*: Método utilizado para encerrar a participação de uma sessão e forçar a desconexão do mesmo;
- *CANCEL*: Este método cancela uma requisição ainda pendente, ou seja, requisições que não foram respondidas com um *RESPONSE*.

Segundo [8], temos também os seguintes métodos:

- *SUBSCRIBE*: Método utilizado quando o usuário pede para ser avisado de um determinado evento;
- *NOTIFY*: Notifica a ocorrência de um dado evento para quem enviou a requisição *SUBSCRIBE*;
- *PUBLISH*: É uma publicação da presença de um usuário, ou seja, é uma mensagem de "estou *online*";
- *INFO*: Fornece informações adicionais à sessão sem modificá-la;
- *REFER*: Utilizado quando se quer transferir uma chamada;
- *MESSAGE*: Para o envio de mensagens instantâneas;
- *UPDATE*: Modifica o estado da sessão.

Na Figura 3, temos um exemplo de uma requisição do tipo *INVITE*:

```
INVITE sip:9998@voip.uema.br:5080;transport=udp SIP/2.0
Via: SIP/2.0/UDP 10.0.0.111:48498;rport;branch=z9hG4bK54476
Max-Forwards: 70
To: <sip:9998@voip.uema.br:5080;transport=udp>
From: <sip:9997@voip.uema.br:5080>;tag=z9hG4bK34099020
Call-ID: 746869291954@10.0.0.111
CSeq: 358588902 INVITE
Contact: <sip:9997@10.0.0.111:48498>
Expires: 3600
User-Agent: VoIP UEMA
Content-Length: 149
Content-Type: application/sdp

v=0
o=9997@voip.uema.br:5080 0 0 IN IP4 10.0.0.111
s=Session SIP/SDP
c=IN IP4 10.0.0.111
t=0 0
m=audio 21000/8 RTP/AVP 8
a=rtpmap:8 PCMA/8000
```

Figura 3: Exemplo de mensagem *INVITE*. Fonte: Autor.

Já a *RESPONSE* é gerada por um UAS, ou servidor SIP, para responder a uma *REQUEST* feita por um UAC apresentando o formato da Figura 4.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.0.0.111:48498;branch=z9hG4bK31765;received=177.54.143.202;rport=48497
From: <sip:9997@voip.uema.br:5080>;tag=z9hG4bK09052949
To: <sip:9997@voip.uema.br:5080;transport=udp>;tag=as0a2cd500
Call-ID: 883576770678@10.0.0.111
CSeq: 1084011188 REGISTER
Server: FPBX-2.11.0(11.20.0)
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO, PUBLISH, MESSAGE
Supported: replaces, timer
Expires: 900
Contact: <sip:9997@10.0.0.111:48498>;expires=900
Date: Mon, 05 Jun 2017 01:39:53 GMT
Content-Length: 0
```

Figura 4: Exemplo de resposta do servidor.. Fonte: Autor.

Na Figura 4, a primeira linha da *RESPONSE* possui o *SIP version*, que garante a correta interpretação das mensagens, o status code, contendo um número inteiro que identifica o resultado do processamento do *REQUEST* recebido e o *reason phrase* trazendo uma informação textual resumida do significado do status code [22].

Existem 6 classes de código de resposta ou status code, sendo 5 delas derivadas do HTTP e a última criada para o SIP, são elas [8]:

- 1xx – Informativo: Informa que a mensagem *REQUEST* foi recebida e está sendo processada. Ainda não é a resposta final. Por exemplo: quando um UAC envia uma requisição *INVITE*, esta *RESPONSE* é retornada a ele para que o mesmo não continue a enviar a mesma requisição;
- 2xx – Sucesso: Indica que a mensagem foi processada e aceita. No caso de um *INVITE*, ao receber uma *RESPONSE* deste tipo, deve-se retornar uma *ACK* confirmando;
- 3xx – Redirecionamento: Geralmente enviado pelo *Redirect Server* para indicar que o destinatário da chamada não se encontra naquela localização e retorna junto dela o atual local dele;
- 4xx - Erro do Cliente: Informa que houve um pedido com sintaxe inválida ou um erro na solicitação por parte do cliente. Este deverá refazer a solicitação de acordo com o status code retornado;

- 5xx - Erro do Servidor: Informa que a requisição não pode ser enviada com sucesso devido a um erro no servidor, então o cliente poderá refazer a solicitação e mandar a outro servidor ou aguardar para que o problema seja resolvido;
- 6xx - Erro Global: Notifica que ocorreu um erro e que essa mensagem irá falhar ao ser enviada para qualquer servidor disponível. Quando o servidor envia esse tipo de *RESPONSE*, significa que ele possui total conhecimento do destinatário;

2.3.1.3 Cabeçalho SIP

O cabeçalho do SIP segue as mesmas regras do HTTP no escopo *header-field*, onde o *header* é um marcador que representa o nome do campo de cabeçalho e *field* é um marcador que representa a informação do cabeçalho. Ambos são *case insensitive*, ou seja, o que não for compreendido pelo servidor será descartado [8]. Alguns dos marcadores mais comuns nas mensagens SIP são:

- *Request line* e *Status line*: Se for uma requisição, se trata da *Request line* e é onde se encontra o tipo da mesma, acompanhado de um SIP URI, indicando o usuário ou serviço o qual o pedido está sendo enviado e a versão do SIP. Caso seja uma resposta, temos o *Status line* composto pela versão do SIP, o status code e a *reason phrase* da resposta;
- *Via*: Composto pela versão do SIP, o protocolo de transporte utilizado, que pode ser: TCP, UDP, TLS (transporte com camada de criptografia), WS (*WebSocket*) ou WSS (*WebSocket Secure*);
- *From*: Um nome opcional e a SIP URI do solicitante;
- *To*: Nome também opcional e SIP URI do destinatário, este pode ser alterado pelo *Proxy Server* quando necessário;
- *Call-ID*: Cabeçalho indispensável em uma comunicação SIP que traz um identificador único e global para a mensagem. Ele é gerado localmente e de forma randômica composto pelo número, um "@" e em seguida o endereço IP do transmissor;
- *CSeq*: Também é um cabeçalho obrigatório que funciona como um contador de requisições feitas, exceto para os métodos *ACK* e *CANCEL*. Ele é utilizado pelo UAS para verificar se a mensagem é uma nova requisição ou apenas uma retransmissão e utilizado pelo UAC para identificar a qual requisição uma determinada resposta pertence;

- *Expires*: Este se trata de um cabeçalho que determina a quem recebe esta mensagem por quanto tempo (em segundos) essa mensagem é válida. No caso do método *REGISTER*, ele é quem determina quanto tempo um usuário deve ser mantido registrado no servidor até que envie uma nova requisição de registro ou não;
- *Authorization*: É um campo importante, pois, por questões de segurança, o servidor não autoriza certas requisições, como *REGISTER*, sem ela para que não trafegue os dados do usuário de forma desprotegida. Por isso o servidor utiliza de uma autenticação semelhante ao *HTTP Digest* para que o usuário criptografe suas credenciais e reenvie para o servidor e também faz uso do TLS (*Transport Layer Security*) para garantir mais segurança ainda;

Desses cabeçalhos: *Via*, *From*, *To*, *Call-ID* e *CSeq* sempre são replicados na respectiva *RESPONSE*.

2.3.2 RTP

À medida que serviços como rádio, telefonia, música por demanda, vídeo por demanda etc. foram se tornando mais comuns na rede de dados, percebeu-se que cada uma destas aplicações estava reinventando aproximadamente o mesmo protocolo de transporte em tempo real. Logo, houve a necessidade de se criar um único protocolo genérico para várias aplicações [16].

Então, foi criado o RTP (*Real-time Transport Protocol*), definido na RFC 1889 [25], com o objetivo de fornecer um meio uniforme para dados com requisitos temporais no transporte ponto-a-ponto e também com capacidade de transferência de dados para múltiplos destinos, através da multidifusão (*multicast*). O RTP se encontra em uma posição muito relativa na pilha de protocolos, fazendo parte tanto da camada de Aplicação como também da camada de Transporte do modelo TCP/IP, sendo executado sobre o UDP, embora funcione também com o TCP, como apresentado na Figura 5, pois se trata de um protocolo de transporte rápido, simples e não orientado a conexão que, somado ao RTP, permite multiplexar diversos fluxos de informações multimídias e assim atender às exigências de tempo real [16].

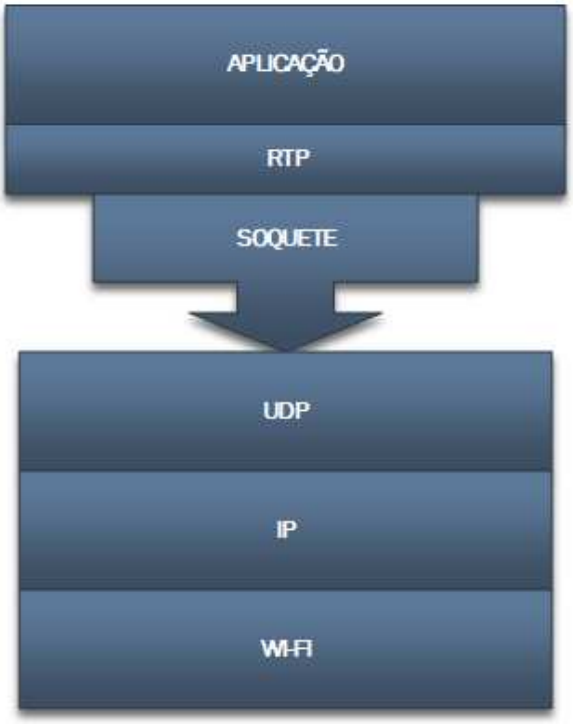


Figura 5: Exemplo do RTP na pilha de protocolos. Fonte: Autor.

Seu funcionamento consiste basicamente em armazenar na biblioteca RTP, encontrada junto à aplicação multimídia, o fluxo de dados a ser enviado onde será realizado a multiplexação e codificação desses fluxos em pacotes RTP, que são inseridos em um soquete. Este soquete funciona como uma interface entre a aplicação e o kernel do sistema operacional de modo que este último recebe os pacotes RTP, os encapsula em segmentos UDP que, em seguida, é incorporado em um pacote IP prosseguindo com os demais encapsulamentos até finalmente ser enviado. Ao chegar ao destinatário, o receptor extrai do UDP o pacote RTP e deste obtém o fragmento da mídia que será passado ao reprodutor de mídia [19].

Desta forma, o protocolo RTP é bem confuso quanto a sua localização na pilha de protocolos. A melhor resolução deste problema é descrita a seguir.

> Como consequência dessa estrutura, é um pouco difícil dizer em que camada o RTP está. Como ele funciona no espaço do usuário e está vinculado ao programa aplicativo, certamente parece ser um protocolo de aplicação. Por outro lado, ele é um protocolo genérico e independente das aplicações que apenas fornecem recursos de transporte, e assim também é semelhante a um protocolo de transporte. Talvez a melhor descrição do RTP seja como um protocolo de transporte implementado na camada de aplicação[…]. (TANENBAUM, A. S; WETHERALL, D. 2011, p. 343).

Nota-se em seu funcionamento que o RTP não possui nenhum mecanismo para garantir se um dado chegou na data prevista, se foi entregue, se chegou em ordem ou outra qualidade de serviço. Porém, o RTP permite identificar qual é o tipo de informação contida no pacote, acrescentar indicadores temporais e aplicar números sequenciais em cada pacote de forma que, no caso em que a ordem dos pacotes é alterada, é possível reconstruir as informações da mídia corretamente. Caso um pacote da mídia esteja faltando, a retransmissão não é uma opção viável visto que o pacote chegaria tarde demais para ser útil e, assim, a melhor medida a ser tomada é fazer uma aproximação do valor perdido através de métodos de interpolação [16].

Para manter a característica de um protocolo genérico o RTP pode conter várias amostras de mídias com diversas codificações possíveis através da definição de perfis, podendo ser apenas um ou vários, para um único fluxo de mídia, onde cada um destes perfis traz um formato de codificação. Assim, pode-se ter um fluxo de áudio, por exemplo, codificado em amostras PCM de 8 *bits* a 8 kHz, codificação GSM (*Groupe Special Mobile*), codificação MP3 (*Moving Picture Experts Group 1 Layer-3*) e entre outros [16].

Como pode-se perceber, é necessário informar ao destinatário as diversas características da mídia, como a codificação por exemplo, que ele está recebendo para que possa ter conhecimento de como realizar os vários tratamentos sobre aqueles dados recebidos e obter a mídia de forma correta. Desta forma, o RTP apresenta um campo de cabeçalho onde o emissor irá inserir as várias informações necessárias sobre o pacote que está enviando.

2.3.2.1 Cabeçalho RTP

O cabeçalho do RTP apresenta a estrutura da Figura 6:

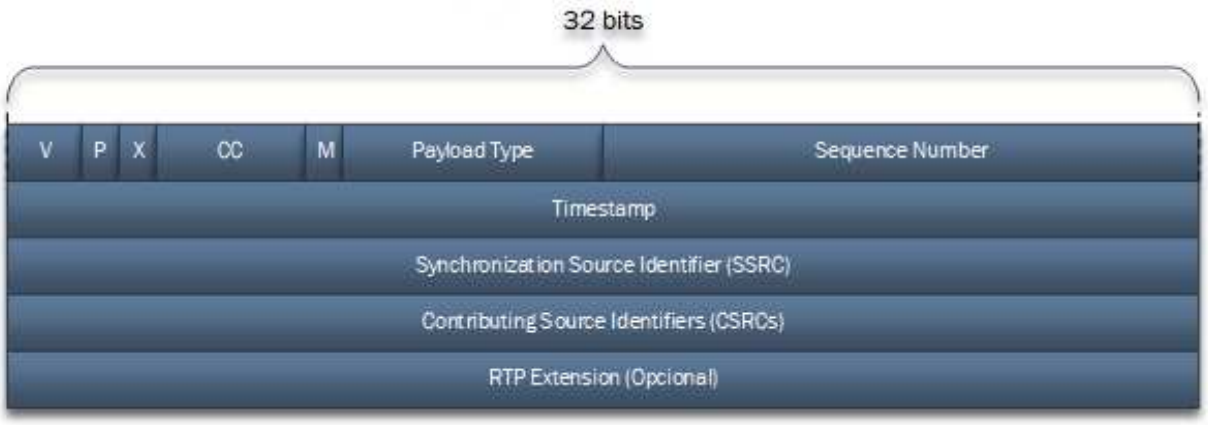


Figura 6: Cabeçalho RTP. Fonte: Autor.

Conforme [26], para cada um dos componentes do cabeçalho, temos as seguintes definições:

- V: de 2 *bits*, este campo identifica a versão do protocolo;
- P: com 1 *bit*, informa a adição ou não de *bits* no conteúdo, geralmente para uso de algoritmos de criptografia ou quando o conteúdo é pequeno. Se tem o valor 1, indica que o pacote foi completado até chegar a um múltiplo de 4 *bytes* [16], onde o último *byte* do preenchimento informa quantos *bytes* foram adicionados. Caso seja 0, apenas indica que não houve adição;
- X: também com 1 *bit*. Quando apresenta o valor 1, informa que há um aumento do cabeçalho normal para um estendido com mais informações. O formato e significado da extensão não é definida no cabeçalho, apenas é informado o comprimento dele na primeira palavra do mesmo;
- CC: de 4 *bits*, apresenta o número de identificadores CSRC que acompanha o cabeçalho, contendo um valor que vai de 0 a 15;
- M: apenas 1 *bit*, é um marcador de uso da aplicação. Pode ser usado para marcar o começo de um quadro de vídeo ou uma palavra em um canal de áudio, por exemplo, de forma que a aplicação possa reconhecer;
- *Payload Type*: com 7 *bits*, este campo identifica o tipo de algoritmo de codificação (codec) utilizado sobre o dado enviado. Graças a este campo, é possível que a codificação mude durante a sessão sem apresentar problemas;
- *Sequence Number*: possui 16 *bits* e tem a função de ordenar os pacotes a serem transmitidos de modo que o primeiro pacote receba um número aleatório e a cada nova

transmissão ele é incrementado. Este campo fecha a primeira palavra de 32 *bits* do cabeçalho RTP;

- *Timestamp*: de 32 *bits*, informa o momento em que o primeiro octeto dos dados foi gerado;
- SSRC: com 32 *bits*, é chamado de *Synchronization source identifier* e serve para identificar a que integrante o pacote pertence. É escolhido de maneira aleatória por cada integrante para identificá-lo na sessão de modo que duas ou mais fontes de sincronização na mesma sessão RTP não venha a ter o mesmo identificador SSRC. Embora a probabilidade de ocorrer que dois ou mais integrantes tenham o mesmo identificador ser baixa, o RTP está preparado para resolver as colisões. Assim, ele é utilizado para multiplexar e demultiplexar os vários fluxos de dados em um único fluxo de pacotes UDP;
- CSRC: também com 32 *bits*. Definido como *Contributing source identifier*, este identifica as fontes que contribuíram para a formatação dos dados contidos no pacote. É utilizado em casos onde há mixers de áudio, por exemplo, em que este passará a ser a fonte de sincronização e os fluxos que estão sendo mixados serão listados neste campo.

2.3.3 WebRTC

O WebRTC é um projeto open-source padronizado pela *World Wide Web Consortium* (W3C) e, também, pelo IETF para permitir que aplicativos e sites da *Web* possam capturar e transmitir dados de mídias, bem como trocar informações entre eles sem a necessidade de um intermediador [33]. Seu primeiro lançamento foi realizado pelo Google em março de 2011 quando criou o projeto WebRTC que, atualmente é suportado pela Apple, Google, Microsoft, Mozilla entre outros [34].

Este projeto consiste em diversas APIs (*Application Programming Interface*) que trabalham juntas. Estas APIs estão disponíveis em JavaScript em quase todos os navegadores modernos, assim como também está presente na maioria dos clientes nativos (como Android e iOS) através de bibliotecas [34].

Segundo [33], o processo de conexão através do WebRTC se dá:

- A conexão entre a aplicação local e o ponto remoto é criada utilizando a interface *RTCPeerConnection*. Esta interface também é responsável pela manutenção, monitoramento e fechamento da conexão;
- Uma vez que a conexão é estabelecida, *MediaStreams* e/ou *RTCDataChannels* podem ser adicionadas à conexão;
- As transmissões de mídia feitas pela interface *MediaStream* podem conter diversas faixas de dados de mídia, podendo ser áudio, vídeo ou texto. Essas transmissões podem ser o envio e recebimento de mídias em tempo real (tais como *lives* ou conferências) assim como também para transmissão de dados salvos (como filmes);
- Também pode ser realizada a troca de dados arbitrários utilizando o *RTCDataChannel*. Esses canais podem ser utilizados para trafegar diversos tipos de dados tais como *metadatas* e transferência de arquivos.

2.3.4 SDP

Esta parte da mensagem se trata do corpo ou *body* e serve para descrever a sessão a ser iniciada ou modificada, podendo aparecer tanto na requisição como na resposta, como já foi mencionado. Para tal, o SDP dispõe de vários parâmetros, muitos opcionais, porém os principais e obrigatórios são [27]:

- v (*protocol version*): campo que contém a versão do SDP;
- o (*owner/creator and session identifier*): Traz informações sobre o criador da sessão e identificadores de sessão, contém os seguintes campos:
    - <*username*>: Contém o host do usuário originador;
    - <*session-id*>: é um identificador aleatório exclusivo da sessão;
    - <*version*>: Funciona como o *CSeq* do cabeçalho SIP, logo ele é incrementado a cada alteração da sessão;
    - <*network-type*>: Informa o tipo da rede. Inicialmente, “IN” define que a rede é a Internet, mas outros valores podem ser associados;
- s (*session name*): Contém o nome da sessão.

2.3.5 Arquitetura Limpa

Conforme os projetos de *software* vão crescendo, fica cada vez mais complexa a manutenção e escalabilidade do mesmo, desta forma antes de iniciar um grande projeto é

necessário planejar a arquitetura do mesmo. Existem vários conceitos para se estruturar a arquitetura de um projeto, dentre eles o mais comum é o *Clean Archtecture* (Código Limpo).

Idealizado por Robert C. Martin (conhecido como *Uncle Bob*), a Arquitetura Limpa consiste na separação das obrigações do projeto em camadas. Assim, o projeto se torna:

- Independente de agentes externos: a regra de negócio da aplicação, que é a parte mais importante do projeto, é totalmente independente da tecnologia em que está sendo implementada, da interface de usuário e até mesmo do Banco de Dados. Assim, podemos extrair a regra de negócio para ser utilizada com outra tecnologia ou realizar mudanças nos recursos que o projeto utiliza sem grandes problemas;
- Testabilidade: com a estruturação do projeto em camadas, a aplicação se torna facilmente testável, já que podemos aplicar os testes sem depender de elementos externos como o Banco de Dados ou a Interface do Usuário.

As camadas são organizadas em círculos concêntricos que representam cada área do *software*, onde quanto mais se adentra nas camadas, mais abstratas ficam, ou seja, mais independente de agentes externos como pode ser mostrado na Figura 7. Outra característica é a Regra de Dependência, onde cada camada é dependente somente das camadas mais internas. Assim, é necessário que qualquer camada da estrutura não tenha qualquer conhecimento das camadas externas à mesma.

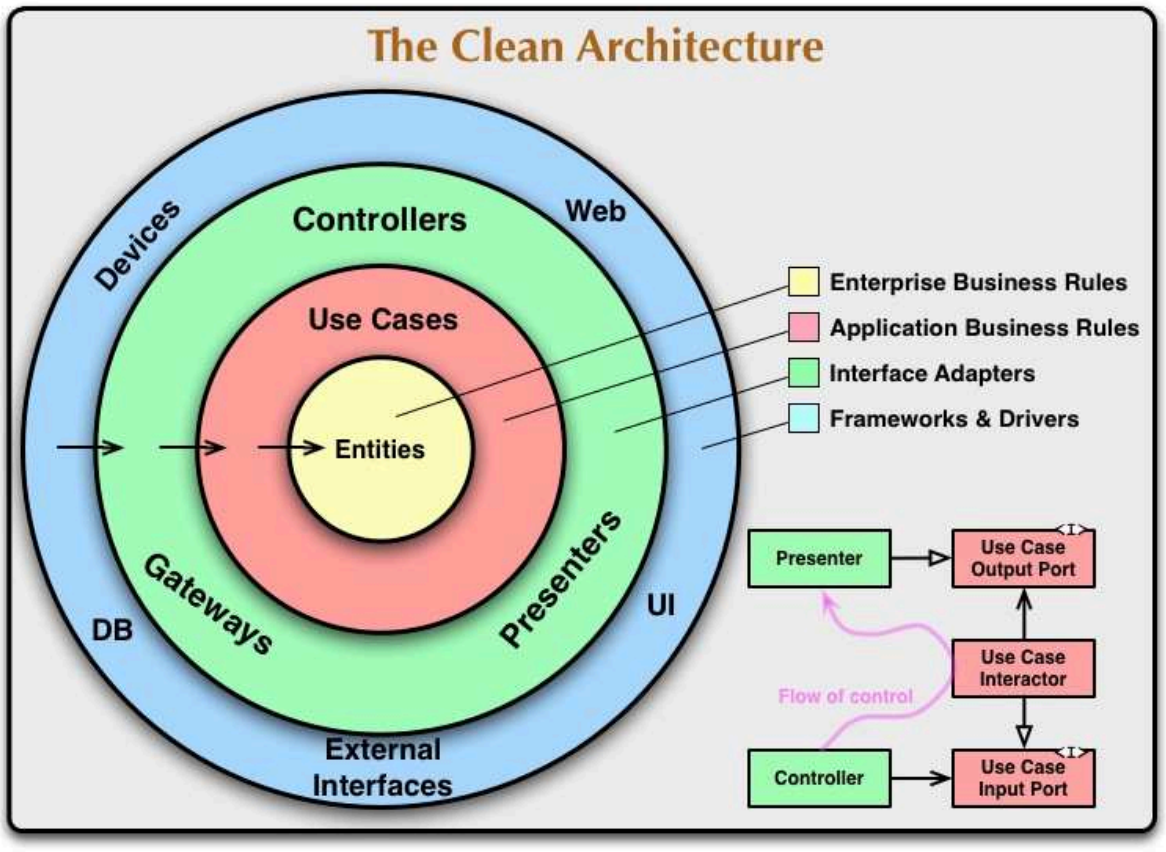


Figura 7: Camadas da Arquitetura Limpa. Fonte: The Clean Code Blog [36].

Geralmente a Arquitetura Limpa divide o projeto em 3 camadas, são elas:

- Domínio: é a camada mais interna do projeto, onde se encontra a regra de negócio. É composto pelas Entidades, que representa os dados e regras da aplicação, *Use Cases*, que orquestram o fluxo de dados da aplicação, e a declaração dos Repositórios, que traz as informações da camada de dados para os *Use Cases*;
- Dados: onde é realizado a aquisição e/ou salvamento dos dados da aplicação. A camada de dados inclui a implementação dos Repositórios e as Fontes de Dados, onde é feito o fluxo de dados com o sistema;
- Apresentação: nesta camada se encontra a Interface do Usuário, onde são implementados as telas, interfaces de API e etc. Também é onde se encontram os gerenciadores de estado da interface.

Com isso, o sistema que implementa os conceitos da Arquitetura Limpa apresenta baixo acoplamento (cada parte do sistema possui pouca dependência das demais) e alta coesão (responsabilidades bem definidas) [36].

3. DESENVOLVIMENTO

O aplicativo VoIP UEMA, atualmente publicado na *Play Store*, foi desenvolvido para Android na linguagem oficial no momento de seu desenvolvimento, que é o Java. A escolha pela plataforma Android foi dada pelo grande número de usuários que o sistema oferece em relação à segunda plataforma mais utilizada, o iOS. Porém, houve a necessidade do desenvolvimento de uma versão do aplicativo também para o sistema iOS para atender toda a comunidade.

Segundo [35], entre fevereiro de 2020 e fevereiro de 2021, o mercado de sistemas operacionais de *smartphones* no Brasil é composto por:

- Android: 86,03%;
- iOS: 13,66%;
- Samsung: 0,26%;
- Windows: 0,02%;
- Playstation: 0,01%;
- Desconhecidos: 0,01.

Desta forma, ao disponibilizar o aplicativo para as plataformas Android e iOS, cobrimos 99,69% dos usuários de *smartphones*. Além do mais, os 0,31% dos usuários restantes que não possuem dispositivos com as plataformas selecionadas podem acessar o serviço através do Discador Web.

No entanto, desenvolver uma nova versão do aplicativo para o sistema iOS em qualquer de suas linguagens oficiais (Objective C ou Swift) se torna inviável devido aos seguintes fatores:

- É um novo aplicativo, ou seja, o desenvolvimento será feito do zero sem poder aproveitar o código do aplicativo desenvolvido para Android;
- O desenvolvimento nas linguagens Objective C e Swift para iOS requerem, exclusivamente, uma computador com Mac OS, visto que as ferramentas de desenvolvimento (Xcode e Simulador de iOS) são disponíveis apenas para o sistema MacOS;
- Um computador com o sistema oficial MacOS tem um custo muito elevado;
- A manutenção do sistema é dobrado, pois cada um deles vai apresentar peculiaridades específicas;

- A implementação de novas funcionalidades é muito trabalhosa porque serão implementados em cada um dos sistemas separadamente e sem garantia de funcionamento igual para ambas as plataformas;
- Manter os dois projetos sincronizados com as mesmas funcionalidades é trabalhoso.

O aplicativo VoIP UEMA, disponível na *Play Store*, está utilizando uma nova biblioteca para sinalização SIP e transporte em tempo real RTP , o PJSIP, desenvolvida pelo grupo Teluu Ltd. apresentando código aberto e contribuição de uma comunidade de desenvolvedores espalhada pelo mundo. Esta ferramenta foi escrita utilizando código nativo C que possibilita a utilização dos recursos disponíveis no sistema com grande performance, além disso esse projeto apresenta diversas funcionalidades além da implementação do protocolo SIP e RTP, como a capacidade de integração com a biblioteca OpenSSL, também escrita em C, que atribuí uma camada de segurança na sinalização SIP e a implementação do transporte de mídia seguro do protocolo RTP que é o SRTP e também conta com diversos codecs além de possibilitar a integração com outros codecs, como o Opus [32].

Porém, uma das características mais vantajosas nesse projeto é que ele foi desenvolvido para múltiplas plataformas, tanto *mobile* como *desktop*, dentre elas temos o Android e iOS. Com essa biblioteca, pode-se então compilar o projeto para utilizar no Android, assim como também compilar o projeto para utilização no iOS. Devido a essas grandes vantagens, o projeto PJSIP foi incorporado no aplicativo VoIP UEMA substituindo a antiga biblioteca MJSIP.

3.1 Pesquisa de ferramenta para desenvolvimento multiplataforma

Sabe-se que o problema do desenvolvimento para *smartphones* em multiplataformas é comum a todos os desenvolvedores *mobile* e, para satisfazer essa demanda, surgiram diversos *frameworks* com tecnologias variadas para alcançar seus objetivos. Então, foi realizada uma pesquisa sobre essas ferramentas disponíveis com objetivo de selecionar o *framework* que melhor atendesse a necessidade do aplicativo VoIP UEMA, dentre elas, temos:

- Performance: a exigência de performance é trivial para qualquer bom produto, o *framework* escolhido teria que apresentar uma boa performance para ambos os dispositivos;

- Comunicação com o Nativo (Java, Kotlin, Objective-C e Swift): é preciso que a ferramenta tenha uma comunicação performática e simples com a parte do código escrito na linguagem oficial de cada dispositivo, visto que a biblioteca PJSIP gera o código nativo (C/C++) que se comunica através de ferramentas específicas com a linguagem oficial, como o NDK no Android;
- Fácil configuração e codificação.

Muitas das ferramentas disponíveis se utilizam de motores de renderização de páginas *Web*, utilizando o Google Chrome (Android) e Safari (iOS), que possibilitam criar aplicativos escritos em Javascript, HTML e CSS e executar em um aplicativo nativo. Muito bom para quem vem do desenvolvimento *Web*, porém sua performance é precarizada pois executa um *browser* (navegador) para renderizar o *layout* do aplicativo, além de sua comunicação com a parte nativa serem difíceis ou até impossíveis em face da necessidade de executar alguns componentes nativos tais como a renderização do vídeo.

Existem outros *frameworks* que não se utilizam de motores de renderização para desenvolvimento multiplataforma, mas muitos deles sempre apresentam problemas de performance e complexidade de código. Porém, existem atualmente dois *frameworks* no mercado mundial dos dispositivos móveis que superam todos os demais citados: React Native e Flutter.

O React Native é um *framework* de código aberto desenvolvido pelo Facebook e lançado em 2015. Esta ferramenta constrói aplicações de renderização nativas, ou seja, utilizando componentes nativos próprios de cada um dos sistemas operacionais (Android e iOS). A linguagem adotada no *framework* foi o JavaScript, onde é implementado toda a lógica de negócio da aplicação, e o JSX (HTML/CSS) responsável pela construção da estrutura dos elementos gráficos que irão ser exibidos na tela do dispositivo [29]. Para funcionar, são criados as *threads JSCore virtual machine*, para executar o código JavaScript, e o Shadow Queue, para cálculos de *layout*. Porém, para possibilitar a renderização dos componentes gráficos nativos na UI do dispositivo, executados na chamada *main thread*, é necessário uma conexão entre elas e este é o papel da Bridge que é um meio de comunicação serializado e agrupado entre a *thread* do JavaScript e a *thread* nativa (Java ou Objective-C/Swift) onde a UI informa para a parte JavaScript o evento que ocorreu, por exemplo um toque na tela, e este retorna a ação a ser feita e vice-versa [30].

O Flutter é um *framework* de código aberto desenvolvido pela Google para atender a necessidade do mercado de uma ferramenta de fácil utilização com performance nativa. Ele utiliza a linguagem Dart, lançada pela Google em 2011, que é compilado diretamente para as arquiteturas dos dispositivos (ARM e x86). Utilizando a ferramenta de renderização 2D chamada Skia, o *framework* desenha todos os componentes que serão utilizados na aplicação numa "tela em branco" chamada *canvas*, isso torna os elementos gráficos dos aplicativos bastante leves, por não converter para uma linguagem nativa, e garante que cada componente será desenhado exatamente igual (quanto ao *design*, não necessariamente em dimensão) em qualquer dispositivo. O *framework* também conta com uma gama de componentes gráficos, chamados de *widgets*, muito maior que a do React Native, por exemplo, com esses componentes criando uma espécie de árvore onde alguns *widgets* podem conter outros *widgets*, isto é chamada de árvore de *widgets* no Flutter [28].

Para realizar a comunicação da porção do código escrito em Dart com a parte nativa do dispositivo (Java e Objective-C/Swift) é utilizado a API *PlatformChannel*, baseado em mensagens, pode-se realizar uma chamada ao código nativo enviando uma mensagem que pode conter ou não variáveis como parâmetros e receber uma resposta depois desse processamento e vice-versa. A passagem de mensagens e aguardo da resposta é feita de forma assíncrona para garantir que a interface se mantenha responsiva.

Analisando essas duas ferramentas, podemos criar uma tabela comparativa entre elas:

Tabela 3: Comparação entre o Flutter e o React Native.

	React Native	Flutter
Linguagem	JavaScript (popular)	Dart (não é comum)
Documentação	Complicada	Simples e estruturada
Componentes de UI	Poucos (dependente do nativo)	Bastante (sem dependência do nativo)
Performance	Um pouco baixa, devido a necessidade da Bridge para se comunicar com os módulos nativos	Ótima, visto que realiza sua própria renderização e garantem performance de 60 FPS e 120 FPS(para os que suportam)
Produtividade	Cai com a complexidade	Produtiva, independente da complexidade
Comunidade	Grande	Pequena, porém crescente

Comunicação com o Nativo	Complexo	Simples

Devido às características apresentadas na Tabela 3, principalmente quanto à produtividade e comunicação com o nativo, o Flutter foi a ferramenta escolhida para atender a necessidade do desenvolvimento multiplataforma.

3.2 Mudança do PJSIP para o dart-sip-ua

Durante a criação do projeto utilizando o *framework* Flutter, conseguiu-se reaproveitar a infraestrutura nativa (Java e C/C++) do aplicativo VoIP UEMA. Com ela foi possível realizar o registro e a chamada de áudio com sucesso utilizando a nova ferramenta. Porém, mostrou-se ser um grande desafio a implementação da chamada de vídeo, visto que teria que realizar a *streaming* de vídeo do ambiente nativo para o código em Dart.

No entanto, o maior desafio foi compilar o projeto PJSIP para iOS. Apesar da excelente documentação, contava-se com recursos limitados visto que, como dito anteriormente, o desenvolvimento nativo para iOS requer um computador com o sistema operacional macOS. Com isso, foram feitas diversas tentativas sem sucesso de compilar o projeto PJSIP para gerar os binários executáveis no sistema.

3.2.1 Projeto Flutter-WebRTC

Em paralelo às tentativas de criação do projeto utilizando o PJSIP, foi procurado na comunidade do Flutter possíveis candidatos que viessem a substituir o projeto PJSIP devido à demanda exigida pelo mesmo.

Então chegou-se ao projeto Flutter-WebRTC. Este projeto é mantido pela CloudWebRTC, desenvolvedora de projetos VoIP/RTC, através de um repositório no GitHub, uma plataforma de hospedagem de código-fonte, e publicado no Pub.dev, o gerenciador oficial de pacotes Dart/Flutter, que conta com diversos contribuidores. Este projeto consiste em uma biblioteca que estabelece uma comunicação WebRTC de aplicações desenvolvidas em Flutter (Dart) através de APIs que foram criadas segundo a padronização do projeto WebRTC, originalmente em JavaScript. Porém, a biblioteca conta apenas com o WebRTC, necessitando, assim, da implementação da camada SIP para o completo funcionamento no projeto VoIP UEMA.

3.2.2 Projeto dart-sip-ua

Posteriormente, o mesmo desenvolvedor do projeto Flutter-WebRTC cria e publica no Pub.dev uma biblioteca puramente escrita em Dart chamada “dart-sip-ua”. Essa biblioteca insere uma camada de sinalização SIP acima do *plugin* Flutter-WebRTC possibilitando a sinalização SIP utilizando *WebSockets*, com a capacidade de realizar chamadas de áudio, vídeo e texto através do WebRTC, podendo aplicar criptografia na comunicação.

Sendo assim, a substituição do projeto PJSIP pelo plugin dart-sip-ua se deu pelos seguintes motivos:

- É um projeto puramente em Dart: diferente do projeto PJSIP, que apresentaria extensos códigos nativos de cada plataforma e binários próprios de cada arquitetura de processadores que ambos os sistemas abrangem, o dart-sip-ua é um projeto escrito 100% em Dart (apesar do projeto Flutter-WebRTC apresentar códigos nativos), isso facilita a implementação no projeto VoIP UEMA já que não será mais necessário a compilação e criação da infraestrutura nativa para cada um dos sistemas operacionais;
- Utiliza o WebRTC: este é um projeto mantido por grandes empresas, logo é uma ferramenta com garantias de longas manutenções. Além disso, ambos os projetos (dart-sip-ua e Flutter-WebRTC) contam com vários contribuidores que os mantém atualizados;
- Segurança: o projeto já apresenta a implementação das camadas de segurança (sinalização e mídia) necessárias para atender aos mesmos requisitos atendidos quando foi implementado a biblioteca PJSIP.
- Qualidade da mídia: os primeiros testes realizados com a biblioteca mostraram um resultado muito satisfatório, superando até mesmo o resultado do projeto PJSIP;
- As dificuldades do PJSIP: como mencionado anteriormente, a utilização do projeto PJSIP apresentava diversas barreiras, como a compilação para iOS e a realização de chamada de vídeo na versão Android;
- Implementação simplificada: além de ser escrito puramente em Dart, a estrutura do projeto é simples e intuitiva.

3.3 Desenvolvimento do aplicativo

Após a escolha do projeto dart-sip-ua e os devidos testes de viabilidade para o projeto VoIP UEMA, iniciou-se o desenvolvimento da aplicação com a interface do projeto.

Inicialmente, desenvolveu-se o *mockup* das interfaces do aplicativo. Esse *mockup* define o *design* que será apresentado no aplicativo, tal como cores, ícones e fontes de texto. Esta etapa é extremamente importante, pois visa implementar uma interface amigável através de uma boa UI/UX (*User Interface/User Experience*) que venha a agradar o público alvo: os membros da comunidade acadêmica da UEMA.

Em seguida, foi desenvolvido um MVP (*Minimum Viable Product*) para testar a integração com o ambiente do VoIP UEMA. Neste MVP foi possível realizar o registro diretamente com o servidor SIP hospedado na UEMA utilizando credenciais de teste e realizar chamadas de áudio e vídeo com sucesso que foram apresentadas na defesa de Qualificação.

Após esta etapa, partiu-se para a inicialização do projeto oficial onde, inicialmente, foi validado a melhor forma de criar um projeto escalável (visando a implementação de novas funcionalidades), personalizável (para a implantação em outros ambientes de modo rápido e fácil) e com boa arquitetura (prezando pela produtividade).

Então, foi criado o projeto VuCore. Trata-se de um projeto *plugin* que acomoda toda a lógica de autenticação com WebService, registro no servidor SIP, a realização e recebimento de chamadas (tanto áudio como vídeo), além de baixar a lista de contatos, registro de chamadas e etc. Desta forma, toda a regra de negócio da aplicação móvel fica contida neste *plugin*, bastando apenas realizar a instalação em um projeto de aplicativo móvel feito em Flutter e inserir dados como a URL do WebService, do servidor SIP e outras personalizações.

Posteriormente, foi criado o projeto VoIP UEMA App que importa o pacote VuCore, mencionado anteriormente, com toda a regra de negócio. Nele são criadas as interfaces que serão apresentadas na tela do *smartphone*, seguindo o design estabelecido pelo *mockup* e toda personalização exclusiva da UEMA, tais como a utilização do SIGUEMA e a disponibilidade da lista de contatos, e as conecta ao *plugin* VuCore.

Desta forma o projeto do aplicativo VoIP UEMA foi concluído apresentando código limpo e escalabilidade, devido a aplicação do *Clean Code* e *Clean Architecture*. Além do mais, o projeto intitulado VoIP University pode ser implantado em outros ambientes graças

à característica personalizável do projeto, já que toda a lógica de negócio se encontra separada no pacote VuCore.

Algumas capturas de telas retiradas de um iPhone são apresentadas nas Figuras 8, 9 e 10.

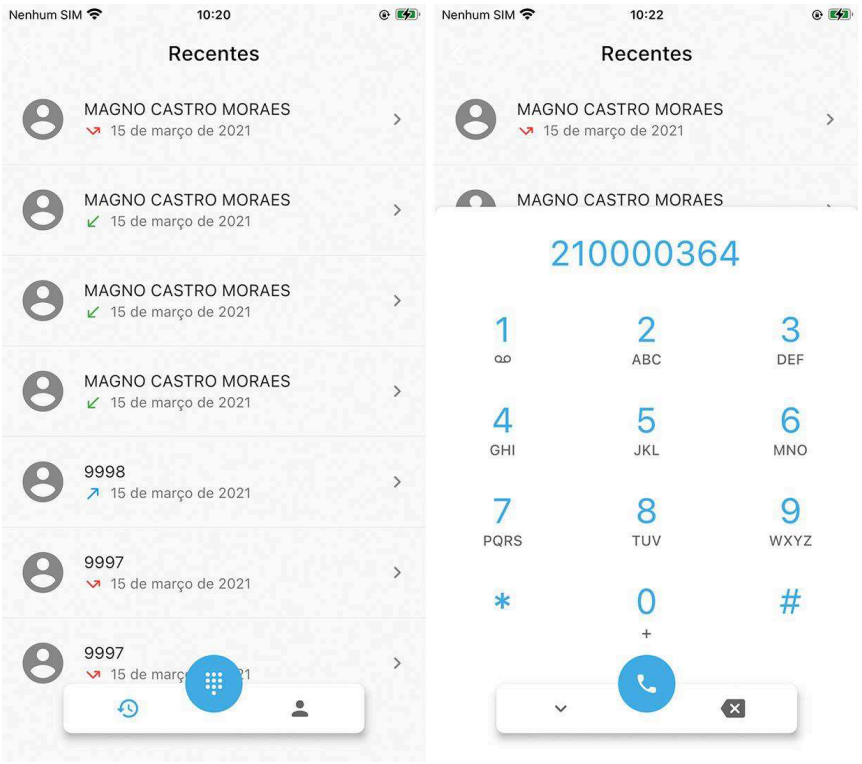


Figura 8: Captura da tela principal com discador ativado. Fonte: Autor.

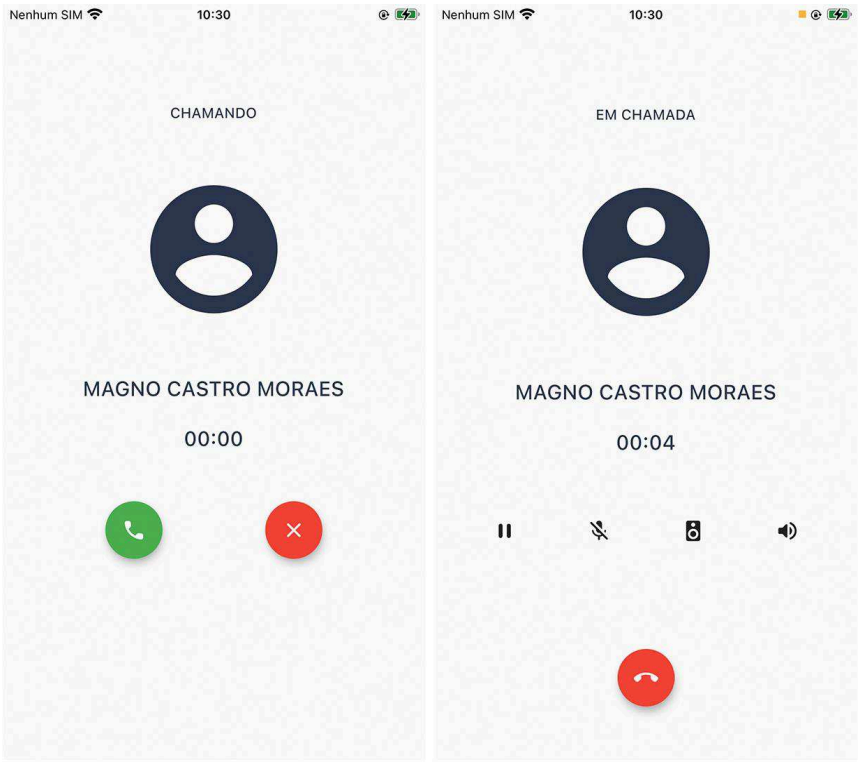


Figura 9: Captura da tela de chamada de áudio. Fonte: Autor.

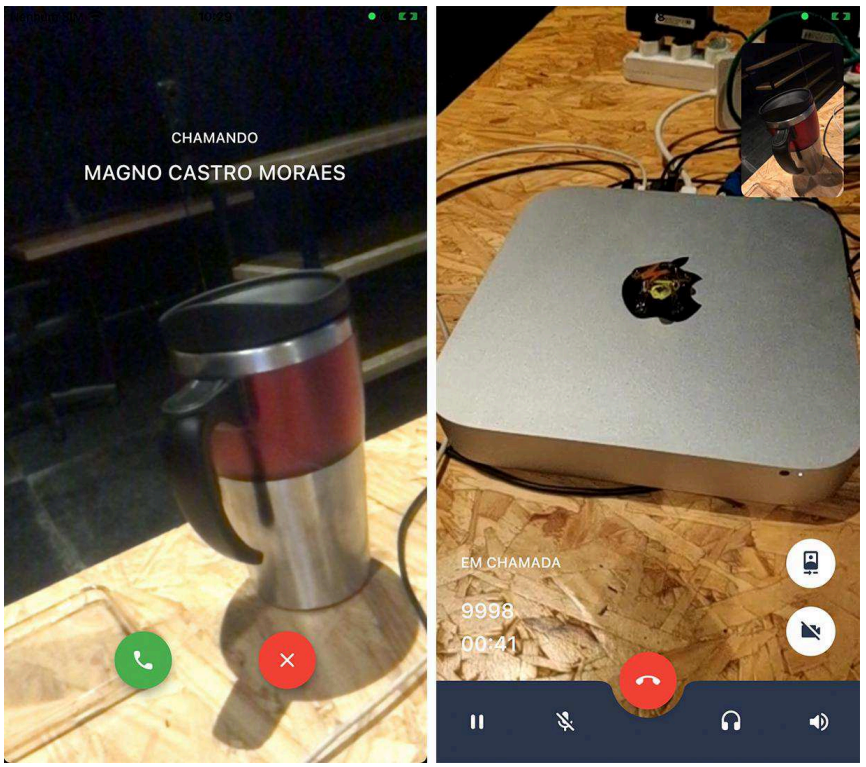


Figura 10: Captura da tela de chamada de vídeo. Fonte: Autor.

3.4 Submissão de *paper*

A solução VoIP University no seu projeto implementado na UEMA, o VoIP UEMA, gerou um *paper* que foi submetido ao *American Council on Science and Education* (ACSE) [37] conforme resultado na Figura 8.

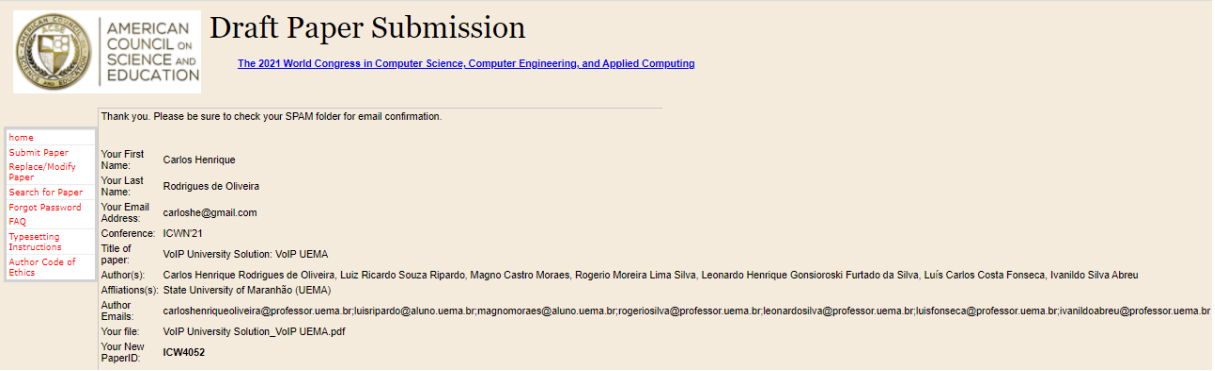


Figura 8: Resumo da submissão de *paper*. Fonte: *American Council on Science and Education* [37].

3.5 Certificado de Registro de Programa de Computador

A solução VoIP UEMA também conta com certificados de registro de programa de computador realizados junto ao NIT-UEMA. Dentre eles, podemos destacar o certificado com o processo BR512019002241-6 intitulado “Aplicativo VoIP UEMA para serviços quadriplay na nuvem versão 1.1” que foi submetido no decorrer deste projeto.

4. CONCLUSÃO E TRABALHOS FUTUROS

O projeto VoIP UEMA poderá crescer bastante com os projetos desenvolvidos neste trabalho. Graças a isso, é possível implementar o projeto VoIP UEMA na universidade contemplando os usuários com um *softphone* que cobre todos os membros da comunidade.

Com isso, já foi possível aplicar a solução VoIP UEMA para os sistemas iOS. Além do mais, com o Flutter, há uma expectativa maior na interface com usuário, devido a sua natureza performática, com atualizações a 60 fps, além de implementações de recursos visuais mais atrativos, graças à simplicidade de sua implementação.

Para trabalhos futuros, temos:

- Desenvolver uma versão para *browser*, mantendo padrão de cliente e implementando videochamada;
- Desenvolvimento de uma solução também para computadores com Windows, Linux e MacOS;
- Implementar a mensagem de texto;
- Avançar no projeto para permitir a integração com o Café RNP, através de seu serviço de telefonia IP, o fone@RNP.
- Avançar no projeto para adoção por secretarias do governo e pela iniciativa privada.

- **REFERÊNCIAS**

[1] YAMAMOTO, R. et al. **Validation of VoIP System for University Network**. Chiba, Japão. 2008.

[2] **Information and Technology Services, University of Michigan**. Disponível em: <https://its.umich.edu/communication/telephone/voip>. Acesso em 08 de março de 2021.

[3] PACIEVITCH, T. **Alexander Graham Bell**. Disponível em: <http://www.infoescola.com/biografias/alexander-graham-bell/>. Acesso em: 23 de abril de 2019.

[4] MUSARDO, F. **Primeiro computador digital eletrônico: ENIAC**. 2015. Disponível em: <http://musardos.com/1946/08/31/primeiro-computador-digital-eletronico-eniac/>. Acesso em: 23 de abril de 2019.

[5] FERNANDES, A. O. et al. **Teoria Geral de Sistemas**. UFSC. 2005. Disponível em: <http://www.agenorzapparoli.com.br/index.php?p=ahistoria.html>. Acesso em: 29 de abril de 2019.

[6] BEZERRA R. M. S., **Um estudo do protocolo SIP e sua utilização em redes de telefonia móvel**, IFBA, 2010. Disponível em: <http://www2.ufba.br/~romildo/downloads/trabalho_apl_conv.pdf>. Acesso em: 02 de abril de 2019.

[7] ANDRADE, S. R. et al. **Comunicação através da tecnologia VoIP**. UNIVAR.

[8] DUARTE, O. C. M. B. D. et al. **SIP (Session Initiation Protocol)**. UFRJ. 2006. Disponível em: < https://www.gta.ufrj.br/grad/06_1/sip/index.html> Acesso em: 23 de maio de 2019.

[9] SIMONE, H. **Requisitos e Proposta para Implantação de um Servidor VoIP**. UFRS, 2008. Disponível em: <http://www.lume.ufrgs.br/handle/10183/15984>. Acesso em: 20 de maio de 2019.

[10] RIBEIRO, G. S. **Voz sobre IP I: A Convergência de Dados e Voz**. 2011. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialvoipconv/>. Acesso em: 01 de junho de 2019.

[11] JUNIOR, J. M. S. **Uma aplicação de voz sobre IP baseada no Session Initiation Protocol**. UFP. 2003. Disponível em:

<http://repositorio.ufpe.br:8080/xmlui/handle/123456789/5597>. Acesso em: 01 de junho de 2019.

[12] LINS, R. D.; BARBOSA, D. C. P.; NASIMENTO, V. C. de O. **VoIP: conceitos e aplicações**. Rio de janeiro: Brasport, 2011.

[13] WEI, X.; BOUSLIMANI, Y.; SELLAL, K. **VoIP Based Solution for the Use Over a Campus Environment**. IEEE Canadian Conference on Electrical and Computer Engineering – University of Moncton, New Brunswick, Canadá, 2012.

[14] CRUZ, D. H. C. **Solução de voz sobre IP para o Centro de Ciências Tecnológicas da Universidade Estadual do Maranhão**. UEMA, 2016.

[15] WALKER, John Q.; HICKS, Jeffrey T. **The Essential Guide to VoIP Implementation and Management**. NetIQ Corporation, 2002.

[16] TANENBAUM, A. S; WETHERALL, D. **Redes de Computadores**. 5. ed. São Paulo: Pearson Education, 2011.

[17] FERNANDES, Nelson Luiz Leal. **Voz sobre IP: uma visão geral**. Artigo científico. São Paulo: EDUSP, 2002.

[18] RODRIGUES, A. A. B. et al. **Apostila de telefonia**. UFRJ, 2003.

[19] KUROSE, James F.; ROSS, K. W. **Redes de computadores e a Internet: uma abordagem top-down**. 6. ed. São Paulo: Pearson Education do Brasil, 2013.

[20] HANDLEY, M. et al. **RFC 2543: SIP: Session Initiation Protocol.** Internet Engineering Task Force (IETF). 1999. Disponível em: <https://tools.ietf.org/html/rfc2543>. Acesso em: 06 de abril de 2019.

[21] ROSENBERG, J. et al. **RFC 3261: SIP: Session Initiation Protocol**. Internet Engineering Task Force (IETF). 2002. Disponível em: <https://tools.ietf.org/html/rfc3261>. Acesso em: 05 de abril de 2019.

[22] JUNIOR, P. N. L. F. **Push-to-Talk no Celular II: Protocolos**. 2007. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialpushtotalk2/>. Acesso em: 25 de maio de 2019.

[23] FIELDING, R. et al. **RFC 2068: Hypertext Transfer Protocol -- HTTP/1.1**. Internet Engineering Task Force (IETF). 1997. Disponível em: <https://tools.ietf.org/html/rfc2068>. Acesso em: 20 de abril de 2019.

[24] YERGEAU, F. et al. **RFC 2279: UTF-8, a transformation format of ISO 10646**. Internet Engineering Task Force (IETF). 1998. Disponível em: <https://tools.ietf.org/html/rfc2279>. Acesso em: 12 de abril de 2019.

[25] SCHULZRINNE, H. et al. **RFC 1889: RTP: A Transport Protocol for Real-Time Applications**. Internet Engineering Task Force (IETF). 1996. Disponível em: <https://tools.ietf.org/html/rfc1889>. Acesso em: 10 de abril de 2019.

[26] CCM. **Os protocolos RTP/RTCP**. 2017. Disponível em: <http://br.ccm.net/contents/281-os-protocolos-rtp-rtcp>. Acesso em: 23 de maio de 2019.

[27] W3II.COM. **SIP Tutorial**. 2017. Disponível em: <http://www.w3ii.com/pt/session_initiation_protocol/default.html>. Acesso em: 01 de junho de 2019.

[28] **Flutter Documentation**. Disponível em: <https://flutter.dev/docs>. Acesso em: 16 de setembro de 2019.

[29] **React Native Documentation**. Disponível em: <https://facebook.github.io/react-native/docs/getting-started>. Acesso em: 17 de setembro de 2019.

[30] CÂMARA, R. **O que você deve saber sobre o funcionamento do React Native**. 2018. Disponível em: <https://medium.com/tableless/o-que-voce-deve-saber-sobre-o-funcionamento-do-react-native-7e3c610aa268>. Acesso em: 17 de setembro de 2019.

[31] **Opus Interactive Audio Codec**. Disponível em: <http://opus-codec.org/>. Acesso em: 21 de outubro de 2019.

[32] **PJSIP Getting Started**. Disponível em: <https://trac.pjsip.org/repos>. Acesso em 12 de setembro de 2019.

[33] **WebRTC API**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/API/WebRTC_API>. Acesso em 23 de maio de 2020.

[34] **Real-time communication for the web**. Disponível em: <https://webrtc.org/>. Acesso em 22 de maio de 2020.

[35] **Statcounter GlobalStats**. Disponível em: <https://gs.statcounter.com/os-market-share/mobile/brazil>. Acesso em 03 de março de 2021.

[36] **The Clean Code Blog**. Disponível em: <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>. Acesso em 05 de março de 2021.

[37] **American Council on Science and Education**. Disponível em: <https://american-cse.org/>. Acesso em 08 de março de 2021.