UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA

MARIA GABRIELA LIMA DAMASCENO

ANALISADOR DE LOGS DE REDES MÓVEIS EM DISPOSITIVOS ANDROID NA LINGUAGEM PYTHON

Manaus
2023

MARIA GABRIELA LIMA DAMASCENO

ANALISADOR DE LOGS DE REDES MÓVEIS EM DISPOSITIVOS ANDROID NA LINGUAGEM PYTHON

Projeto de pesquisa desenvolvido durante a disciplina de Trabalho de Conclusão de Curso II e apresentado à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Engenheiro Eletricista.

Orientador: Angilberto Muniz Ferreira Sobrinho, Dr.

Manaus
2023

Universidade do Estado do Amazonas – UEA
Escola Superior de Tecnologia - EST

Reitor:
André Luiz Nunes Zodahib
Vice-Reitor:
Kátia do Nascimento Coureiro
Diretora da Escola Superior de Tecnologia:
Ingrid Sammyne Gadelha Figueiredo
Coordenador do Curso de Engenharia Elétrica:
Israel Gondres Torné

Banca Avaliadora composta por: *Data da defesa: 30/03/2023*
Prof. Angilberto Muniz Ferreira Sobrinho *(Orientador)*
Prof. Jozias Parente de Oliveira
Prof. Fábio de Sousa Cardoso

CIP – Catalogação na Publicação

Damasceno, Maria Gabriela Lima

Analisador de Logs de Redes Móveis em Dispositivos Android na Linguagem Python / Maria Gabriela Lima Damasceno; [orientado por] Angilberto Muniz Ferreira Sobrinho. – Manaus: 2023.
67 p.: il.

Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica). Universidade do Estado do Amazonas, 2023.

1. Redes Móveis. 2. Android. 3. Python. I. Sobrinho, Angilberto Muniz Ferreira.

MARIA GABRIELA LIMA DAMASCENO

ANALISADOR DE LOGS DE REDES MÓVEIS EM DISPOSITIVOS ANDROID NA LINGUAGEM PYTHON

Projeto de pesquisa desenvolvido durante a disciplina de Trabalho de Conclusão de Curso II e apresentado à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Engenheiro Eletricista.

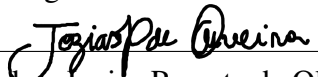
Nota obtida: 9,5 (nove vírgula cinco)

Aprovada em 30/03/2023

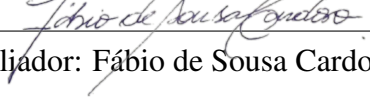
Área de concentração: Engenharia de Telecomunicações

BANCA EXAMINADORA

Orientador: Angilberto Muniz Ferreira Sobrinho, Dr.



Avaliador: Jozias Parente de Oliveira, Dr.



Avaliador: Fábio de Sousa Cardoso, Dr.

Manaus

2023

Dedicatória

Para todas aquelas que eram as únicas de suas salas.

AGRADECIMENTO

À Deus, que mais do que conhecimento sempre me deu fé.

Agradeço à minha família, por sempre ter me apoiado em todos esses anos de dedicação aos estudos, principalmente minha mãe Graciete Lima por ter me ensinado sempre que posso fazer tudo o que sonhar em fazer.

Agradeço aos engenheiros e engenheiras que trabalham comigo e que sempre me incentivam a estudar mais nesse mundo gigante de Telecomunicações e que disponibilizaram seu tempo para me ajudar na coleta de dados, muito obrigada, estou devendo chocolate eternamente para todos.

Agradeço a toda equipe que compõe a Universidade do Estado do Amazonas, principalmente ao meu orientador por sempre estar presente quando eu precisei e por ter apoiado esse projeto.

Agradeço aos meus amigos e amigas da faculdade, do Laboratório de Física, do CAEEL e do DAETEC, que sempre estiveram ao meu lado em todos os momentos bons e ruins nesses cinco anos, em especial aos meus amigos do Trio Elétrico que são meus melhores amigos desde o primeiro dia e as minhas amigas do Engmanas que acompanharam todo o processo de construção desse projeto.

Agradeço ao meu amor, Gabriel Coelho, que nesses anos todos, nunca, em nenhum momento, duvidou de mim, mesmo quando até eu duvidava e sempre apoiou meus planos infalíveis.

O mais importante sempre fica para o final, então aqui agradeço especialmente ao meu pai, Marcelo Mendes, por sempre mostrar que eu posso ser a melhor versão de mim mesma e nunca deixando eu me contentar com nada que não fosse bem feito.

A todos aqueles que em algum momento contribuíram para a minha formação, muito obrigada, eu jamais esquecerei.

Você sabe o que me manteve de pé durante todos esses anos de exílio? Fé. Não em deuses. Não em mitos e lendas. Em mim.

Daenerys Targaryen

RESUMO

No mundo atual é quase impossível fazer atividades cotidianas sem o uso das Redes de Comunicação Móvel, seja por dados de internet ou seja por voz. As tecnologias em relação ao funcionamento dessas redes têm uma evolução cada vez maior e mais rápida, sendo o fator principal de diferença entre cada uma delas a capacidade de fornecer serviços com maior eficiência. Dentre os aparelhos de comunicação móvel estão os dispositivos Android, cujo sistema operacional é de código aberto, esses dispositivos geram arquivos de texto, denominados de logs, que indicam quais eventos ocorreram no sistema em um determinado espaço de tempo. Atualmente, uma das funcionalidades do Engenheiro de Telecomunicações é analisar esse tipo de arquivo para poder identificar o funcionamento ou falhas que podem estar ocorrendo nas redes de comunicação móvel. Entretanto esse trabalho é bastante manual e demorado. Sendo assim esta pesquisa tem como fim criar uma ferramenta em linguagem de programação Python, do tipo Dashboard, para auxiliar nas análises e relatórios realizados pelos Engenheiros de Telecomunicações. Durante o processo de criação serão estudadas referências relacionados a Arquitetura das Redes Móveis, Handover, IMS, Potência dos Sinais e Protocolo SIP, assim como também conceitos sobre Arquitetura Android, Coleta de Logs e Linguagem Python. Ao final é descrito o funcionamento de cada função do código responsável pelo funcionamento do Dashboard de forma local em navegadores Web. Assim como, são expostos os gráficos das Redes Móveis utilizadas, os gráficos de potência, intensidade e qualidade do sinal 4G, o gráfico de potência do sinal 3G e Tabela com as mensagens SIP trocadas entre o aparelho móvel e a rede.

Palavras chave: Redes Móveis, Android, Python

ABSTRACT

In today's world, it is almost impossible to carry out daily activities without the use of Mobile Communication Networks, either by internet data or by voice. The technologies in relation to the operation of these networks are evolving more and more quickly, with the main difference between each of them being the ability to provide services with greater efficiency. Among the mobile communication devices are Android devices, whose operating system is open source, these devices generate text files, called logs, which indicate which events occurred in the system in a given period of time. Currently, one of the functions of the Telecommunications Engineer is to analyze this type of file in order to identify the functioning or failures that may be occurring in mobile communication networks. However, this work is quite manual and time-consuming. Therefore, this research aims to create a tool in Python programming language, Dashboard type, to assist in the analyzes and reports carried out by Telecommunications Engineers. During the creation process, references related to Mobile Network Architecture, Handover, IMS, Signal Power and SIP Protocol will be studied, as well as concepts on Android Architecture, Log Collection and Python Language. At the end, the operation of each function of the code responsible for the operation of the Dashboard locally in Web browsers is described. As well as, the graphics of the Mobile Networks used, the graphics of power, intensity and quality of the 4G signal, the graph of the strength of the 3G signal and the Table with the SIP messages exchanged between the mobile device and the network are exposed.

Keywords: Mobile Networks, Android, Python

Lista de Figuras

Lista de Tabelas

LISTA DE ABREVIATURAS, SIGLAS E SÍMBOLOS

3 GPP	3rd Generation Partnership Project (Projeto de Parceria de 3ª Geração)
CSCF	Call/Session Control Functions (Funções de controle de chamada/sessão)
GSM	Global System for Mobile Communications (Sistema Global para Comunicações Móveis)
HSPA	High Speed Packet Access (Acesso a Pacotes de Alta Velocidade)
HSS	Home Subscriber Server (Servidor de Assinante Doméstico)
I-CSCF	Interrogating CSCF (Interrogando CSCF)
IMS	IP Multimedia Subsystem (Subsistema IP de Multimídia)
IP	Internet Protocol (Protocolo de Internet)
LTE	Long Term Evolution (Evolução a Longo Prazo)
MAC	Media Access Control (Controle de Acesso de Mídia)
MME	Mobility Management Entity (Entidade de Gestão de Mobilidade)
NGN	New Generation Network (Rede de Nova Geração)
NR	New Radio (Novo Rádio)
NSA	Non Standalone (Não autônomo)
OSI	Interconexão de Sistemas Abertos
PCRF	Policy and Charging Rules Function (Função de Política e Regras de Cobrança)
P-CSCF	Proxy-CSCF (Procuração CSCF)
P-GW	Packet Data Network Gateway (Portão de Rede de Dados de Pacote)
RAN	Radio Access Network (Rede de Acesso por Rádio)
RSCP	Received Signal Code Power (Potência do Código de Sinal Recebido)
RSRP	Reference Signal Received Power (Potência Recebida do Sinal de Referência)

RSRQ Reference Signal Received Quality (Qualidade do Sinal Recebido de Referência)

RSSI Received Signal Strength Indication (Indicação de Intensidade do Sinal Recebido)

SA Standalone (Autônomo)

S-CSCF Serving CSCF (Servindo CSCF)

SIP Session Initiation Protocol (Protocolo de Iniciação de Sessão)

S-GW Serving-Gateway (Portão de Serviço)

SMS Short Message Service (Serviço de Mensagens Curtas)

SRVCC Single Radio Voice Call Continuity (Continuidade de Chamada de Voz de Rádio Único)

UE User Equipment (Equipamento de Usuário)

UMTS Universal Mobile Telecommunications System (Sistema Universal de Telecomunicações Móveis)

VILTE Video over LTE (Vídeo sobre LTE)

VOLTE Voice over LTE (Voz sobre LTE)

WCDMA Wide-Band Code-Divison Multiple Access (Acesso Múltiplo por Divisão de Código de Banda Larga)

SUMÁRIO

INTRODUÇÃO

No mundo de hoje é praticamente improvável fazer boa parte das atividades cotidianas sem o uso de equipamentos tecnológicos como computadores, dispositivos móveis, por exemplo celulares e tablets, e redes móveis ou sem fio. Desta forma é necessário que constantemente esses aparelhos sejam atualizados para que suportem novas tecnologias de redes móveis, que sempre estão em constante evolução (SANTANA, 2016).

As redes móveis mais utilizadas atualmente no Brasil são as das redes 3G, 4G e, muito em breve, a do 5G. Sendo assim, para que sejam criadas essas novas tecnologias e para que haja manutenção constante entre elas é importante que exista o Engenheiro de Telecomunicações, que será capaz de desenvolver, analisar, corrigir e melhorar o funcionamento dos equipamentos móveis que estarão conectados a essas tecnologias (TELECO, 2023a).

Quando se fala de rede móveis ou redes sem fio é essencial entender que essas tecnologias não são visíveis, ou seja, para que sejam feitas análises é preciso utilizar logs, estes que são arquivos que se apresentam sob a forma de ficheiros de texto, recolhendo cronologicamente todos os eventos que afetaram um sistema de informática como softwares, aplicações, servidores, e todas as ações que resultaram desses eventos (CONNECT, 2021).

Na Engenharia de Elétrica, dentro da área da Engenharia de Telecomunicações, tem-se as subáreas de Redes de Computadores e Redes Móveis, uma das funções de um Engenheiro de Telecomunicações que atua na área de redes móveis é, na maioria das vezes, identificar através dos arquivos de log o que aconteceu entre a rede e o dispositivo. A etapa de identificação de logs é bastante manual e indispensável já que, através dela, é possível saber todas as informações necessárias para efeitos de manutenção, correção de erros e etc (DEVELOPERS, 2023b).

Tendo em vista esse contexto, a implementação de uma ferramenta automatizada que pudesse auxiliar nessa atividade seria bastante útil, visto que, as informações coletadas no log poderiam ser informadas de forma visual e gráfica, melhorando o processo de análise e facilitando também que ingressantes na área de desenvolvimento de softwares para a Engenharia de Telecomunicações possam adquirir conhecimento de redes móveis de forma prática e visual.

Atualmente é fundamental que a Engenharia de Telecomunicações ande em conjunto com a Engenharia de Software, desta forma, o desenvolvimento da ferramenta pretende buscar maior intersecção entre as duas áreas da engenharia, portanto, este projeto de pesquisa teve

como objetivo o desenvolvimento de uma ferramenta, utilizando Linguagem Python que analise logs de redes móveis e demonstre os eventos que ocorreram entre dispositivos móvel e rede de forma visual, fazendo uso de tabelas e gráficos para que seja possível automatizar o processo de análise do Engenheiro de Telecomunicações.

Para a implementação da ferramenta foram utilizados conhecimentos obtidos na formação acadêmica de Engenharia Elétrica, principalmente os presentes nas disciplinas: Linguagem de Programação, Sistemas de Comunicações Móveis, Redes de Computadores, Sistemas de Telecomunicações e Comunicações Digitais. O trabalho pode ser dividido em 4 secções.

Seção I - Referencial teórico: nesta secção é feita a apresentação de todos os parâmetros relacionados a construção da ferramenta, desde a definição de parâmetros fundamentais relacionados a redes móveis, o funcionamento dos logs no android e como é feita a construção de gráficos, tabelas e dashboards na linguagem python.

Seção II - Materiais e Métodos: já nesta seção é definido a forma que são coletados os dados da pesquisa, que são os arquivos de log, demonstrando como pode ser feita essa coleta através do logcat e as funções criados no código em python que permitem a leitura e busca de termos chaves no arquivo.

Seção III - Desenvolvimento do Dashboard: aqui nesta seção é definido que a ferramenta para análise de logs é um dashboard que funciona de forma local nos navegadores Web, nela também são apresentados as funções que fazem parte da construção dos gráficos e tabelas que fazem parte do dashboard.

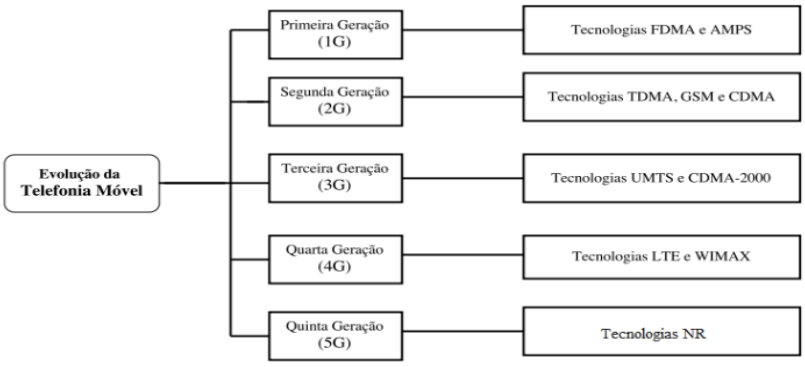
Seção IV - Resultados: por fim, nesta seção é mostrado o produto final, que é a construção da ferramenta com todos os seus recursos visuais demonstrando o funcionamento de cada um deles e explicando de que forma podem contribuir para uma análise de logs de redes móveis em dispositivos android mais eficiente.

1 REFERENCIAL TEÓRICO

1.1 REDES MÓVEIS

É possível definir que as redes de telefonia móvel celular são redes de telecomunicações projetadas para o provisionamento de serviços de telefonia móvel, assim como para a comunicação entre uma ou mais estações móveis. As redes móveis funcionam por meio de radiofrequências conectando-se aos aparelhos móveis, como celulares e tablets. Cada área geográfica é dividida entre células, e cada uma destas há uma estação de rádio base, formada por antenas com receptores e emissores de sinais ligados a uma central telefônica (CORDEIRO, 2012). As redes móveis estão em constante evolução, devido a isso, conforme evoluem são desenvolvidas novas "gerações" como as do 1G, 2G, 3G, 4G e, atualmente, o 5G. Cada uma dessas gerações está associada a alguma tecnologia, como é possível ver na Figura 1.

Figura 1 – Gerações das Redes Móveis



Fonte: (SANTANA, 2016)

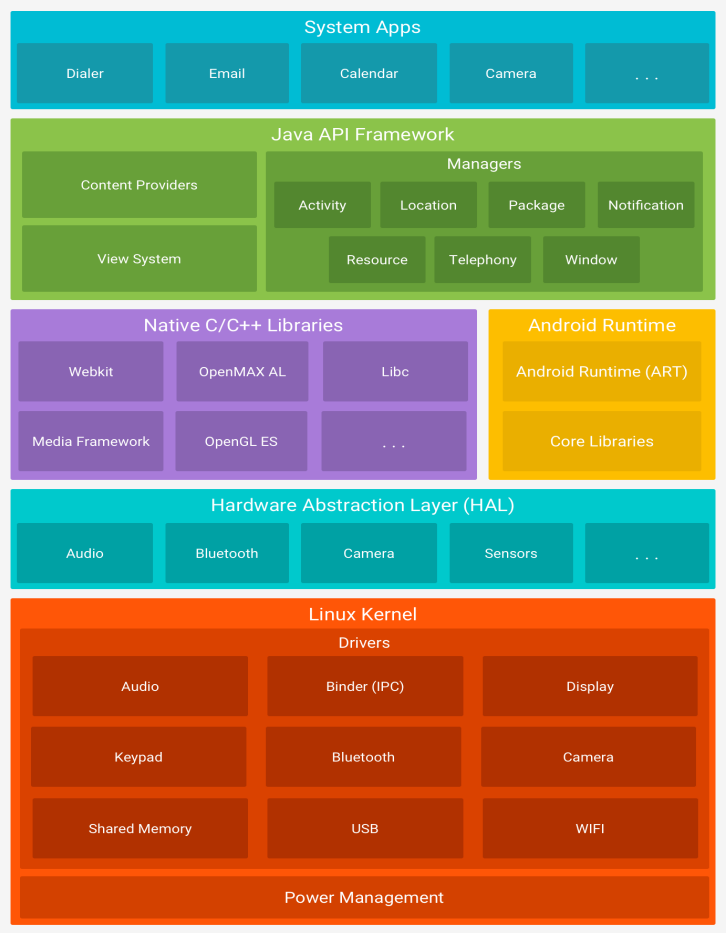
A associação responsável por definir o funcionamento de um sistema móvel é determinada por *3rd Generation Partnership Project* (3GPP), que reúne uma série de órgãos normativos de telecomunicações. O 3GPP define especificações, estruturadas em versões, para sistemas móveis como redes de acesso de rádio, redes principais e funcionamento das arquiteturas de redes móveis. Com a crescente demanda do consumidor por diversos aplicativos de dados e multimídia sem fio e internet móvel, qualidade e aumento da capacidade das redes sem fio tornaram-se uma prioridade para as organizações de desenvolvimento de padrões, como 3GPP (AHMADI, 2013, tradução nossa).

Segundo (TELECO, 2023a), o 3GPP define especificações *Global System for Mobile Communications* (GSM), *Wide-Band Code-Divison Multiple Access* (WCDMA), *High Speed Packet Access* (HSPA), *Long Term Evolution* (LTE) e *New Radio* (NR) para um sistema móvel completo, incluindo aspectos relativos aos terminais, redes de acesso de rádio, redes principais, e partes da rede de serviços. No mundo todo os Órgãos de normatização têm autorização para receber os resultados do 3GPP e publicá-los na sua região, como normas ou recomendações formais.

1.2 PLATAFORMA ANDROID

O Android pode ser definido como o sistema operacional do Google para dispositivos móveis baseado no Linux. Sistemas operacionais são programas que gerenciam todas as tarefas de um dispositivo, e nos fornece uma interface visual para que possamos interagir com um sistema eletrônico sem necessariamente saber o que acontece dentro dele (TECHTUDO, 2011).

Figura 2 – Arquitetura da Plataforma Android



Fonte: (DEVELOPERS, 2023a)

Na Figura 2, é mostrado os componentes da arquitetura Android, neste projeto usa-se especificamente a recursos da APIs programadas em linguagem Java, especificamente na API *Telephony* que é usada para, entre outras coisas, para monitorar as informações do telefone, incluindo os estados atuais do telefone, conexões, rede, etc.

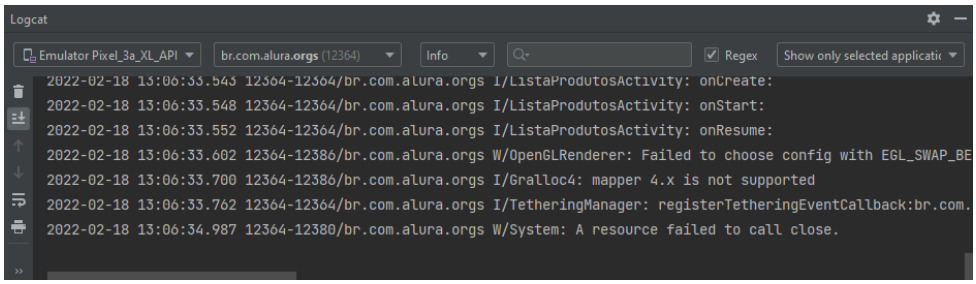
1.3 LOGS DO ANDROID

Arquivos de log são mensagens geradas por computadores ou dispositivos móveis, como celulares e tablets que registram eventos, processos e outras informações durante a operação. Neste trabalho, os logs serão utilizados principalmente para a coleta de dados de informações relacionadas a parte de telecomunicações do aparelho móvel como o registro na rede, envio e

recebimento de chamadas ou mensagens e etc.

Uma das formas de coleta e visualização de logs é a janela do Android Studio chamada Logcat. Como descrito por Bittencourt (2022), o Logcat exibe, em tempo real, mensagens do sistema e as mensagens que adicionamos ao aplicativo por meio da classe Log, além de guardar um histórico das mensagens mais antigas para consulta, como na Figura 3.

Figura 3 – Arquivo de Log no Logcat



Fonte: (BITTENCOURT, 2022)

Os logs analisados são todos do sistema Android que possuem a classe Log (ANDROID, 2022), que permite gravar os registros de eventos através de métodos. Através desses métodos é possível criar mensagens de registro que são geradas e exibidas posteriormente com isso a ferramenta, através da linguagem Python, pode ler essas mensagens de registro e buscar os dados referentes às redes móveis.

1.4 TECNOLOGIA 1G, 2G E 3G

A Geração 1G de redes móveis fazia uso da tecnologia analógica, sendo elaborada no começo dos anos 70. Foi na 1ª Geração que um usuário em movimento pode manter uma ligação telefônica em qualquer região dentro da área de serviço de um operador de redes móveis.

Já com a geração 2G foi possível converter sinal analógico em digital, com essa nova tecnologia foi possível ter uma melhora nos aspectos técnicos e comerciais, podendo suportar mais usuários e ter uma melhor qualidade de comunicação (MENDES, 2013).

A partir da Geração 3G os usuários não estavam apenas usando os celulares para realizar chamadas e enviar SMS, nesse momento passou a ser utilizada a internet e serviços multimídia, havendo um aumento significativo na capacidade de voz e suporte a serviços.

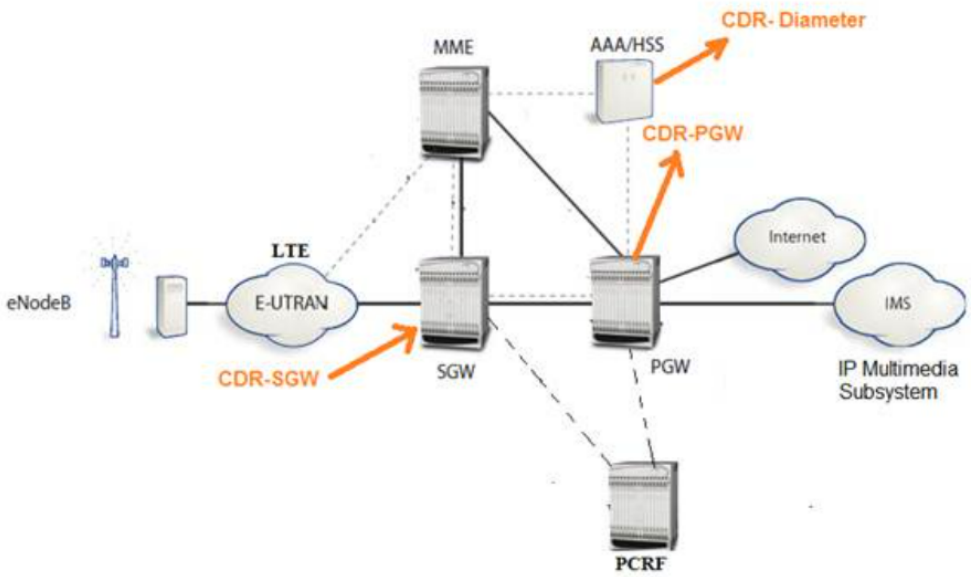
Segundo Teleco (2023b) o *Universal Mobile Telecommunications System* (UMTS) é o termo adotado para designar o padrão de 3ª Geração estabelecido para a rede das operadoras de celular como evolução para operadoras de GSM e que utiliza como interface rádio o *Wideband CDMA* (WCDMA) e suas evoluções.

1.5 TECNOLOGIA 4G LTE

A arquitetura da rede 4G LTE tem como principal característica a implementação de serviços baseados em IP, nas comunicações móveis, tal como a completa otimização do desempenho da rede. O aumento da velocidade das conexões e a qualidade do acesso a esta infraestrutura são considerados fatores importantes para o desenvolvimento de novos conteúdos e aplicações multimídia (TELECO, 2022a).

Os elementos da figura 4 podem ser descritos abaixo:

Figura 4 – Arquitetura LTE



Fonte: (TELECO, 2022a)

- *Home Subscriber Server* (HSS): É o banco de dados da rede que guarda as informações dos usuários.
- *Mobility Management Entity* (MME): É o autenticador da rede lida com a sinalização e controle, a gestão da mobilidade e a distribuição da paginação das mensagens para o eNodeB, a Estação Rádio Base. Fazendo também a gestão do acesso do UE, equipamento do usuário, à rede através da interação com o HSS de forma a autenticar os utilizadores. Fornece a função do plano de controle para permitir a mobilidade contínua entre o LTE e redes móveis 2G/3G e também suporta as intercepções legais de sinalização (TELECO, 2022a).
- *Serving-Gateway* (S-GW): Faz o encaminhamento dos pacotes de dados para a estação rádio base e contabiliza o controle dos dados do utilizador.
- *Packet Data Network Gateway* (P-GW): Atua como ponto de entrada e de saída do tráfego de dados do equipamento do usuário e de interface entre as redes LTE e as redes de

pacotes de dados tais como a Internet ou redes fixas e móveis baseadas em protocolo de iniciação da sessão (SIP) ou protocolo internet de subsistemas de multimédia (IMS) (TELECO, 2022a).

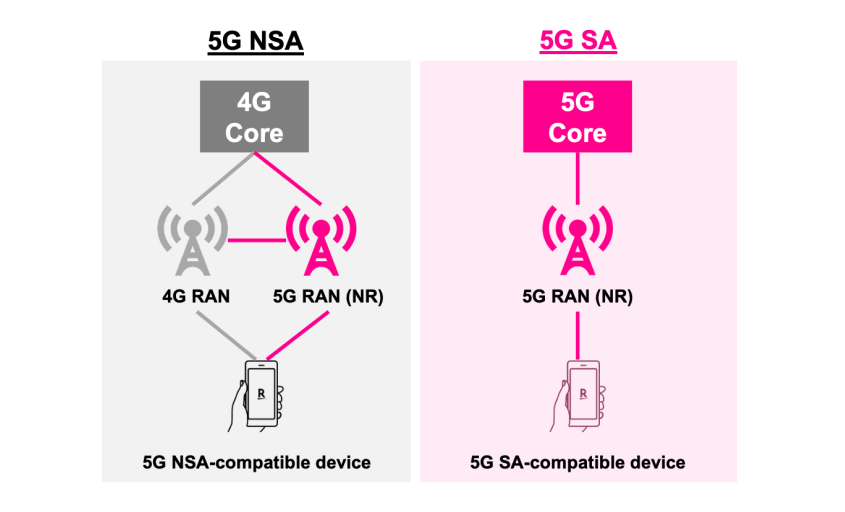
- *Policy and Charging Rules Function* (PCRF): fornece as regras de tarifação com base no fluxo de serviços de dados para o P-GW, dando ou rejeitando permissões de envio de pacotes de multimídia como voz, vídeos, mensagens de texto e etc.

1.6 TECNOLOGIA 5G

A Tecnologia que ainda está sendo implantada no Brasil atualmente é a do 5G, que tem como objetivo viabilizar aplicações relacionadas a Internet das Coisas e ao Acesso Banda Larga Wireless Fixo, assim como aumentar a capacidade para baixar o custo por bit da banda larga móvel. O 5G tem duas implementações, mostradas na Figura 5, definidas pelo 3GPP (2023, tradução nossa):

- 5G *Non Standalone* (NSA): Neste modo a conexão é ainda feita utilizando a rede central da arquitetura 4G (Core 4G), mas portadoras do 5G NR (5G) são utilizadas para aumentar as taxas de dados e tirar proveito da latência reduzida com as 4G RAN (Radio Access Network) e 5G RAN (TELECO, 2022c).
- 5G *Standalone* (SA): Já este modelo irá permitir o 5G NR autônomo com o usuário e o plano de controle usando a rede central de próxima geração 5G (Core 5G) e a 5G RAN (TELECO, 2022c).

Figura 5 – Arquitetura NR



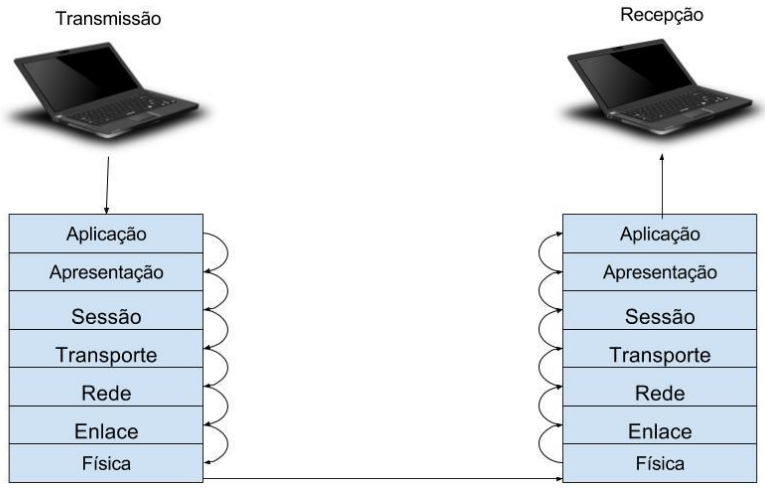
Fonte: (MOBILE, 2021)

1.7 MODELO OSI

O modelo de Interconexão de Sistemas Abertos (OSI) é um padrão para os protocolos de rede, os quais nada mais são do que regras de comunicação usadas para conectar dois ou

mais computadores. O que o modelo OSI faz é agrupar esses protocolos em grupos específicos, ou camadas (MATHEUS, 2018). O modelo está dividido em sete camadas hierárquicas, nas quais cada camada usa as funções da própria ou da camada anterior para transparecer de modo simples as operações ao usuário, seja ele um programa ou alguma outra camada. As camadas são empilhadas na seguinte ordem, como também vistas na Figura 6:

Figura 6 – Camadas do Modelo OSI



Fonte: (MATHEUS, 2018)

a) Camada 7 - Aplicação: Nesta camada estão os programas que garantem a interação humano-máquina. Nela é possível enviar e-mails, transferir arquivos, acessar websites, acessar outras máquinas diretamente e etc.

b) Camada 6 - Apresentação: Nesta camada ocorre a conversão de códigos para caracteres, compactação dos dados e criptografia, caso seja necessário.

c) Camada 5 - Sessão: Esta camada é responsável por estabelecer e encerrar a conexão entre hosts, que são qualquer máquina conectada a uma rede, que conta com número de IP e nome definidos. Esta camada é quem inicia e sincroniza os hosts, realizando o estabelecimento das sessões e dando suporte aos hosts como registros de log e realização de tarefas de segurança.

d) Camada 4 - Transporte: Esta camada garante o envio e o recebimento dos pacotes vindos da camada 3, lidando bastante com a qualidade do serviço para que os dados sejam entregues com consistência, ou seja, sem estarem errados ou duplicados.

e) Camada 3 - Rede: Nesta camada é feito o roteamento entre a origem e destino dos pacotes, no qual o endereço MAC é o endereço físico de quem envia o pacote, já o endereço IP é a identificação da máquina na rede.

f) Camada 2 - Enlace de dados: Nesta camada são definidas as tecnologias como as VLans, ou topologias como a ponto-a-ponto. Os dispositivos como os switches passam a funcionar nesta camada.

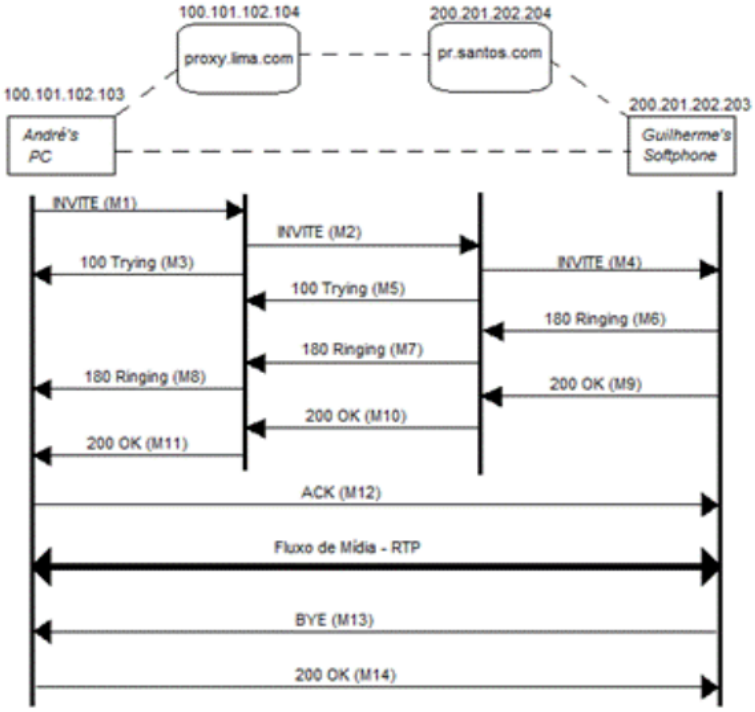
g) Camada 1 - Física: Camada onde são especificados os dispositivos, como hubs e os meios de transmissão, como os cabos de rede.

1.8 PROTOCOLO SIP

O Protocolo de Iniciação de Sessão (SIP) é um protocolo textual baseado em requisições e respostas, que possui campos de cabeçalho e um corpo. No corpo de uma mensagem SIP são encapsulados protocolos de descrição de sessão responsáveis pela negociação dos tipos e formatos de mídias que serão trocadas entre as partes que estão se comunicando.

O SIP é um protocolo da camada de sessão do modelo OSI que pode estabelecer, modificar e terminar sessões multimídias - onde a sessão é considerada uma troca de dados entre uma associação de UAs - como uma chamada telefônica pela Internet (TELECO, 2022b). O fluxo de envio das mensagens SIP pode ser conferido na Figura 7 que representa desde o estabelecimento de uma sessão até o seu encerramento.

Figura 7 – Fluxo das Mensagens SIP



Fonte: (TELECO, 2022b)

Conforme dito por Pisa (2008):

> O protocolo SIP pertence ao grupo dos protocolos de sinalização fim-a-fim baseado em texto, o qual sinaliza o início, a modificação e o encerramento das sessões. As sessões se baseiam no modelo cliente-servidor e independem do tipo de dado trafegado dentro do canal estabelecido. Os protocolos de sinalização para VoIP, como o SIP, devem contemplar a codificação de voz, a configuração das chamadas, o transporte de dados, o modo de autenticação, os requisitos e as tecnologias de segurança, as

> primitivas de comunicação, o formato do cabeçalho e do endereçamento e a sintaxe das mensagens.

É necessário destacar que respostas à requisições SIP são sempre compostas por um número de três dígitos, seguido por uma frase informativa, como em: “100 Trying”, “180 Ringing”, “200 OK”. Estas respostas são classificadas de acordo com o primeiro dígito de seu número. Os dois dígitos seguintes servem como um subtipo, como pode ser visto na Tabela 1.

Tabela 1 – Classe das Respostas SIP

Resposta	**Classe**
1xx	Informativas
2xx	Sucesso
3xx	Redirecionamento
4xx	Falha na Requisição
5xx	Falha no Servidor
5xx	Falha Global

Fonte: (TELECO, 2022b)

Na Tabela 2 são descritos os significados das frases informativas enviadas pelas mensagens SIP, essas mensagens podem ser encontradas nos logs encontrados em aparelhos Android, através delas é possível descrever o fluxo de mídia trocada entre dois comunicantes.

Tabela 2 – Mensagens SIP

Mensagens	**Descrição**
INVITE	Indica que o UE está sendo convidado para participar de uma sessão de chamada.
ACK	Confirma que o método INVITE foi recebido.
REGISTER	Permite que o UE se registre no endereço listado (geralmente na forma user@domain) no campo do destinatário do cabeçalho da mensagem SIP.
BYE	Termina uma sessão SIP.
CANCEL	Usado para desfazer uma sessão antes dela ser estabelecida.
OPTIONS	Usado para consultar as capacidades do servidor.

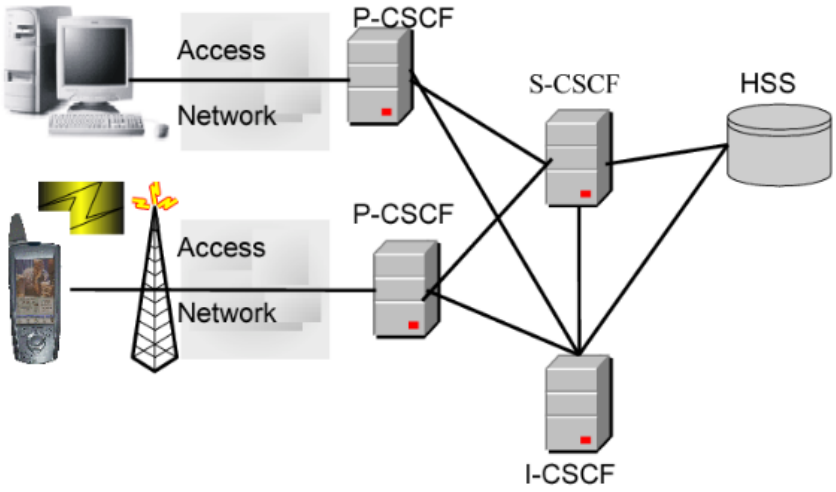
Fonte: (RFC, 2008)

1.9 IMS

IP Multimedia Subsystem (IMS) é uma arquitetura que surgiu com os estudos de evolução da Rede Celular Móvel e foi adotada como core do *New Generation Network* (NGN), que é utilizado para evolução da Rede Fixa para as operadoras de telecomunicações que desejam oferecer serviços de telefonia fixa e móvel multimídia (UFRJ, 2010). Ela usa VoIP baseada em uma implementação *Session Initiation Protocol*, Protocolo de Iniciação de Sessão, (SIP), usado sobre o padrão Internet Protocol (IP) (ANDRADE, 2012).

A Arquitetura IMS pode ser dividida em três camadas, sendo a Camada de Aplicação, com os servidores de aplicações SIP: Camada de Controle, com os servidores CSCF, Call Session Control Function e a Camada de Conectividade com roteadores e switches. O CSCF tem três divisões principais, que podem ser vistas na Figura 8:

Figura 8 – Arquitetura IMS



Fonte: (ABDALLA; VENKATESAN, 2011)

- *Proxy CSCF* (P-CSCF) - Responsável pelo primeiro contato de um terminal com a Rede IMS. Acionado quando, em um primeiro momento, um terminal recebe um IP e entra na rede, gerando um evento de registro.
- *Interrogating CSCF* (I-CSCF) - Ponte entre o P-CSCF e o S-CSCF, que rege as funções das aplicações e os serviços.
- *Serving CSCF* (S-CSCF) - Controlador de sessão, é designado para atender os usuários finais durante sua conexão com a rede IMS (ABDALLA; VENKATESAN, 2011, tradução nossa).

Atualmente a rede IMS é usada em conjunto com a tecnologia LTE, com isto, tem-se amplamente utilizado nos dias de hoje os serviços de Voz sobre LTE (VoLTE), Vídeo sobre LTE (ViLTE), SMS através do LTE (SMS over IP) e etc. Os serviços da rede IMS são os principais dados a serem analisados pelo analisador de log, visto que, ele tem como objetivo analisar logs de redes móveis, dentre elas, a da rede IMS em LTE.

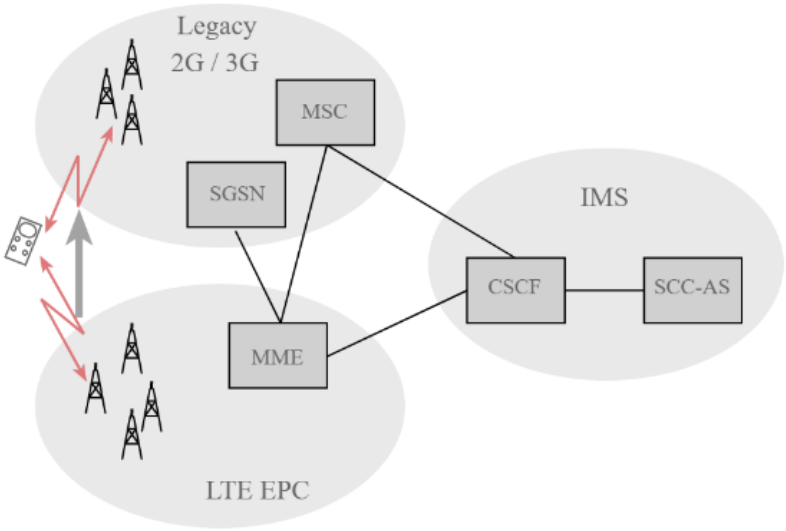
1.10 SRVCC HANDOVER

A tecnologia da rede IMS é a solução do 3GPP de voz sobre IP que está sendo implantada nas redes LTE como implementação para chamadas de voz. Quando um equipamento do usuário (UE) estiver acampado na rede LTE, poderá ser necessário que ele retorne às redes 2G/3G quando a cobertura ou a capacidade da rede exigirem. Sendo assim, o SRVCC fornece a solução para o UE transferir a chamada de voz do IMS/IP para o domínio CS padrão sem que haja qualquer prejuízo na chamada de voz do usuário, como mostrado na Figura 9 (KEYSIGHT, 2013).

De acordo com o (3GPP, 2012):

> Single Radio Voice Call Continuity (SRVCC) refere-se à continuidade de chamada de voz entre o acesso IMS sobre PS e o acesso CS para chamadas ancoradas em IMS quando o UE é capaz de transmitir/receber apenas em uma dessas redes de acesso em um determinado momento.

Figura 9 – SRVCC Handover



Fonte: (ELECTRONICS, 2022)

O funcionamento do SRVCC Handover acontece como descrito na Figura 9 onde o MME é quem envia o comando de troca da tecnologia 4G para a rede 2G/3G. O processo de SRVCC e de Handover são bastante utilizados nas redes móveis para que o aparelho móvel adeque-se à célula que melhor forneça sinal ou área de cobertura, podendo ser analisado também nos logs de redes móveis ou logs de IMS.

1.11 POTÊNCIAS DO SINAL

A Potência do sinal é um dos fatores a ser considerado para uma conexão de dados e de voz de celular bem-sucedida, onde os valores medidos e relatados variam de acordo com o modem, a operadora e o ambiente de rede. Embora a intensidade do sinal possa parecer

adequada, as velocidades de transferência podem variar devido às dependências das cargas da torre de celular.

Vários fatores podem influenciar a potência do sinal, incluindo como: carga da torre, proximidade com a torre de celular, sinais concorrentes, barreiras físicas (montanhas, edifícios, trens, etc.) e até mesmo o clima. Atualmente as tecnologias de dados e voz mais atualizadas são as da tecnologia 3G e 4G, sendo que a potência do sinal recebido pode ser identificada com os parâmetros do sinal de referência, que é uma amostragem do sinal real que está sendo enviado, abaixo:

a) *Received Signal Code Power* (RSCP), potência do sinal de referência em dBm para o 3G.

b) *Reference Signal Received Power* (RSRP), potência do sinal de referência em dBm para o 4G.

c) *Received Signal Strength Indication* (RSSI), potência do sinal de referência em dBm para o 2G, 3G e 4G.

d) *Reference Signal Received Quality* (RSRQ), qualidade do sinal de referência em dB para o 4G.

Nas tecnologias do 2G, 3G e 4G o RSSI representa toda a potência recebida, incluindo a potência desejada da célula servidora, assim como toda a potência do canal e outras fontes de ruído, sendo indicado através do modem por um valor dBm negativo, como mostrado na Figura 10.

Figura 10 – Valores para RSSI 4G

4G signal levels

RSSI	Signal strength	Description
> -65 dBm	Excellent	Strong signal with maximum data speeds
-65 dBm to -75 dBm	Good	Strong signal with good data speeds
-75 dBm to -85 dBm	Fair	Fair but useful, fast and reliable data speeds may be attained, but marginal data with drop-outs is possible
-85 dBm to -95 dBm	Poor	Performance will drop drastically
<= -95 dBm	No signal	Disconnection

Fonte: (TELECOM, 2023)

Já o RSRP fornece ao UE informações essenciais sobre a força das células a partir das que podem ser utilizadas em algoritmos para determinar as configurações ideais para operar na rede (3GPP, 2017). Os valores de RSRP podem ser vistos na Figura 11.

Segundo (TELECOM, 2023, tradução nossa):

> O RSRP faz um trabalho melhor que o RSSI ao medir a potência do sinal de um setor específico, ao mesmo tempo em que exclui ruídos e interferências de outros setores.

O RSRQ é a Qualidade Recebida do Sinal de Referência, este valor é calculado através do RSRP e RSSI (3GPP, 2010). Já que o RSRQ é uma proporção de duas potências de sinal com a mesma unidade, ou seja, dBm, ele usa dB como unidade de medida, como na Figura 12.

Figura 11 – Valores para RSRP

RSRP

RSRP	Signal strength	Description
>= -80 dBm	Excellent	Strong signal with maximum data speeds
-80 dBm to -90 dBm	Good	Strong signal with good data speeds
-90 dBm to -100 dBm	Fair to poor	Reliable data speeds may be attained, but marginal data with drop-outs is possible. When this value gets close to -100, performance will drop drastically
<= -100 dBm	No signal	Disnonnection

Fonte: (TELECOM, 2023)

Figura 12 – Valores para RSRQ

RSRQ

RSRQ	Signal quality	Description
>= -10 dB	Excellent	Strong signal with maximum data speeds
-10 dB to -15 dB	Good	Strong signal with good data speeds
-15 dB to -20 dB	Fair to poor	Reliable data speeds may be attained, but marginal data with drop-outs is possible. When this value gets close to -20, performance will drop drastically
<= -20 dB	No signal	Disconnection

Fonte: (TELECOM, 2023)

Como dito em Teltonika (2018, tradução nossa):

> A medição RSRQ fornece informações adicionais quando RSRP não é suficiente para fazer uma transferência confiável ou decisão de nova seleção de células.

Como dito em Telecom (2023) o RSCP faz referência a potência medida por um receptor, o celular, em um canal de comunicação físico específico. O RSCP pode ser medido em inicialmente tanto no downlink (conexão da rede para o dispositivo) quanto no uplink (conexão do dispositivo para a rede), porém ele é definido apenas para o downlink, desta forma é medido pelo UE e relatado a nó da rede. Os valores de RSCP estão de acordo com a Figura 13.

Figura 13 – Valores para RSCP

RSCP

RSCP	Signal strength	Description
-60 to 0	Excellent	Strong signal with maximum data speeds
-75 to -60	Good	Strong signal with good data speeds
-85 to -75	Fair	Fair but useful, fast and reliable data speeds may be attained
-95 to -85	Poor	Marginal data with drop-outs is possible
-124 to -95	Very poor	Performance will drop drastically, closer to -124 disconnects are likely

Fonte: (TELECOM, 2023)

1.12 LINGUAGEM PYTHON

Segundo a documentação do Python (PYTHON, 2022a), lançada em 1990 por Guido van Rossum, a linguagem Python é uma linguagem de programação de alto nível orientada a objetos, Figura 14. De acordo com a própria documentação, a linguagem foi criada para seguir

o princípio de priorizar a legibilidade do código sobre a velocidade ou expressividade, combinando uma sintaxe clara e concisa com os métodos de sua biblioteca padrão e por módulos e frameworks desenvolvidos por terceiros, como por exemplo as Bibliotecas Pandas e Matplotlib.

Figura 14 – Logo Python

Fonte: (PYTHON, 2022b)

A linguagem Python foi utilizada no desenvolvimento do projeto em questão pois através dela é possível combinar a busca pelas as informações necessárias para compreender o funcionamento do aparelho móvel, através dos seus arquivos de logs, e a construção da parte visual do programa que terá como objetivo expor dos dados coletados de forma para o usuário da ferramenta.

1.13 BIBLIOTECAS PANDAS E MATPLOTLIB

O objetivo da Biblioteca Pandas, Figura 15, é desenvolver software de código aberto para ingestão de dados, preparação de dados, análise de dados e visualização de dados para a linguagem de programação Python (RIBEIRO, 2020). A biblioteca foi utilizada para o tratamento de dados encontrados nos arquivos de log para serem transformados em gráficos e tabelas para serem expostos ao usuário.

Figura 15 – Logo Pandas

Fonte: (PYDATA, 2022)

Segundo a Documentação do Matplotlib (MATPLOTLIB, 2022), em sua essência o Matplotlib, Figura 16, é orientado a objetos, desta forma foi através desta biblioteca que foram feitos boa parte dos tratamentos de dados, do ponto de vista matemático, que irão gerar gráficos e tabelas na ferramenta.

Figura 16 – Logo Matplotlib

Fonte: (MATPLOTLIB, 2022)

1.14 CRIAÇÃO DE DASHBOARDS COM O DASH

O Dashboard pode ser definido como um painel que oferece uma interface gráfica com visualizações rápidas e interativas. A biblioteca Dash é construída em torno do Plotly.js e do React.js e permite construir e realizar o desenvolvimento de painéis interativos customizados de acordo com o usuário (PLOTLY, 2023).

Segundo Filho (2022), dashboards são extremamente úteis para criar Web Apps que permitem monitorar diversos indicadores, em tempo real e de forma automatizada. Os painéis podem ser construídos de diversas formas, inclusive com a linguagem Python em conjunto com a biblioteca Dash como na Figura 17.

Figura 17 – Dashboards com Dash

Fonte: (PLOTLY, 2023)

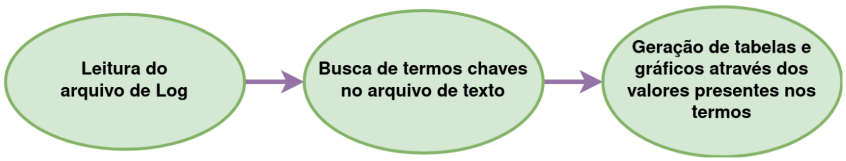
2 MATERIAIS E MÉTODOS

2.1 METODOLOGIA

O trabalho desenvolvido é uma pesquisa aplicada e teve como objetivo realizar uma pesquisa exploratória relacionada ao material bibliográfico apresentado e a construção da ferramenta, sendo abordados os procedimentos técnicos da pesquisa bibliográfica. Para a elaboração do trabalho foram utilizados os métodos de abordagem hipotético-dedutivo e método de procedimento monográfico. Seguidamente, para a coleta de dados foi empregada a documentação indireta, posteriormente, a análise de dados ocorreu de forma qualitativa de maneira global.

Para a construção da ferramenta, que agora pode ser definida como um Dashboard na qual serão expostos os gráficos e tabelas relacionadas aos parâmetros das comunicações móveis do dispositivo, os principais itens utilizados foram os Logs coletados do sistema Android, a Linguagem Python juntamente com as bibliotecas do Matplotlib, Pandas e Dash e o Navegador Web presente no computador.

Figura 18 – Fluxograma da Construção da Ferramenta



Fonte: autoria própria.

Ao clicar em executar o código da ferramenta desenvolvida em Python é possível acessar localmente em qualquer navegador web do computador, onde será gerada uma janela com a página inicial do dashboard, nesta página inicial é onde deve ser anexada o arquivo de log e seguirá o fluxograma da Figura 18, começando pela leitura do arquivo de log, busca de termos chaves no arquivo de texto e a geração de tabelas e gráficos através dos valores encontrados nos termos.

Cada função do funcionamento da ferramenta, para a construção do dashboard é definida nas próximas subseções e o fluxograma do funcionamento da ciração do Dashbpard pode ser conferido no Apêndice A. O código fonte em Python completo também pode ser conferido no repositório online: https://github.com/gabrieladamasceno/analisador.

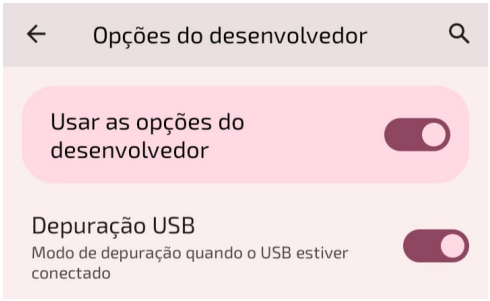
2.2 COLETA DOS ARQUIVOS DE LOG

Para coletar os arquivos de log do celular é utilizada a janela Logcat no Android Studio, que mostra mensagens do sistema como quando em tempo real e mantém um histórico para que possa ser conferida as mensagens mais antigas. Para poder salvar os arquivos de log direto no computador é preciso instalar a ferramenta adb conforme dito por Developers (2023b).

Segundo Studio (2023) com o Android Studio e o adb instalados é possível coletar os logs através do passo a passo a seguir:

1. No dispositivo android é preciso ir em: Configurações, Sobre o Telefone, Clicar sete vezes em Número de Versão, Ativar a opção de desenvolvedor e Ativar Depuração USB, como na figura 19 (CONNECT, 2021).
2. Com o aparelho conectado ao computador é executado o comando adb no terminal do computador como “adb logcat”, de acordo com a Figura 20.
3. A partir desse momento os acontecimentos do aparelho vão estar sendo coletados.
4. Para salvar o arquivo de log basta digitar no computador CTRL+C.
5. Colocando :/adb/ no explorador de arquivos é possível encontrar o arquivo .txt, Figura 21, com os logs.

Figura 19 – Modo Desenvolvedor e Depuração USB



Fonte: autoria própria.

Figura 20 – Coleta de Logs pelo Logcat

```
\TCC\Programa\platform-tools_r33.0.3-windows\platform-tools> adb logcat Redes_Móveis.txt
```

Fonte: autoria própria.

Figura 21 – Exemplo de Log de Redes Móveis no Android

```
01-07 11:21:54.227 10038  1228  1422 D MobileStatusTracker(0/1): onSignalStrengthsChanged signalStrength=SignalStrength:{mCdma=Invalid,mGsm=Invalid,mWcdma=Invalid,mTdscdma=Invalid
01-07 11:21:55.566 10038  1228  1422 D MobileStatusTracker(0/1): onSignalStrengthsChanged signalStrength=SignalStrength:{mCdma=Invalid,mGsm=Invalid,mWcdma=Invalid,mTdscdma=Invalid
01-07 11:21:55.581 10038  1228  1422 D MobileStatusTracker(0/1): onSignalStrengthsChanged signalStrength=SignalStrength:{mCdma=Invalid,mGsm=Invalid,mWcdma=Invalid,mTdscdma=Invalid
01-07 11:21:56.558 10038  1228  1422 D MobileStatusTracker(0/1): onSignalStrengthsChanged signalStrength=SignalStrength:{mCdma=Invalid,mGsm=Invalid,mWcdma=Invalid,mTdscdma=Invalid
01-07 11:21:57.996 10038  1228  1422 D MobileStatusTracker(0/1): onSignalStrengthsChanged signalStrength=SignalStrength:{mCdma=Invalid,mGsm=Invalid,mWcdma=Invalid,mTdscdma=Invalid
01-07 11:21:59.801 10038  1228  1422 D MobileStatusTracker(0/1): onSignalStrengthsChanged signalStrength=SignalStrength:{mCdma=Invalid,mGsm=Invalid,mWcdma=Invalid,mTdscdma=Invalid
01-07 11:22:01.973 10038  1228  1422 D MobileStatusTracker(0/1): onSignalStrengthsChanged signalStrength=SignalStrength:{mCdma=Invalid,mGsm=Invalid,mWcdma=Invalid,mTdscdma=Invalid
01-07 11:22:04.956 10038  1228  1422 D MobileStatusTracker(0/1): onSignalStrengthsChanged signalStrength=SignalStrength:{mCdma=Invalid,mGsm=Invalid,mWcdma=Invalid,mTdscdma=Invalid
01-07 11:22:08.558 10038  1228  1422 D MobileStatusTracker(0/1): onSignalStrengthsChanged signalStrength=SignalStrength:{mCdma=Invalid,mGsm=Invalid,mWcdma=Invalid,mTdscdma=Invalid
01-07 11:22:10.103 10038  1228  1422 D MobileStatusTracker(0/1): onSignalStrengthsChanged signalStrength=SignalStrength:{mCdma=Invalid,mGsm=Invalid,mWcdma=Invalid,mTdscdma=Invalid
01-07 11:22:12.658 10038  1228  1422 D MobileStatusTracker(0/1): onSignalStrengthsChanged signalStrength=SignalStrength:{mCdma=Invalid,mGsm=Invalid,mWcdma=Invalid,mTdscdma=Invalid
01-07 11:22:12.666 10038  1228  1422 D MobileStatusTracker(0/1): onSignalStrengthsChanged signalStrength=SignalStrength:{mCdma=Invalid,mGsm=Invalid,mWcdma=Invalid,mTdscdma=Invalid
01-07 11:22:14.261 10038  1228  1422 D MobileStatusTracker(0/1): onSignalStrengthsChanged signalStrength=SignalStrength:{mCdma=Invalid,mGsm=Invalid,mWcdma=Invalid,mTdscdma=Invalid
01-07 11:22:15.598 10038  1228  1422 D MobileStatusTracker(0/1): onSignalStrengthsChanged signalStrength=SignalStrength:{mCdma=Invalid,mGsm=Invalid,mWcdma=Invalid,mTdscdma=Invalid
```

Fonte: autoria própria.

2.3 FUNÇÕES PARA LEITURA DOS ARQUIVOS DE LOG

Para a leitura do arquivo de log é feita a função save_file(), na Figura 22, e parse_contents(), na Figura 23, que podem ser conferidas no Apêndice B, nas quais a primeira é responsável por ler o arquivo e identificar que ele é das extensões .log ou .txt e a outra é responsável por salvar o arquivo na memória. Se o arquivo anexado não for dos tipos suportados, uma mensagem de erro aparecerá para o usuário informando que houve um erro de processamento.

Figura 22 – Função para Salvar do Arquivo

```
def save_file(filename, filecontent):
    f = open(
        os.path.join(os.path.abspath('.'),'data',filename),
        "w")
    f.write(filecontent)
    f.close()
```

Fonte: autoria própria.

Figura 23 – Função para Leitura o Arquivo

```
def parse_contents(contents, filename, date):
    content_type, content_string = contents.split(',')

    decoded = base64.b64decode(content_string)
    try:
        if 'log' in filename:
            # Confere se o arquivo é do tipo log
            df = decoded.decode('iso-8859-1')

        if 'txt' in filename:
            # Confere se o arquivo é do tipo txt
            df = decoded.decode('utf-8')

        save_file(filename="redes_moveis.log",filecontent=df)
    except Exception as e:
        print(e)
        return html.Div([
            'Houve um erro de processamento deste arquivo.'
        ])
```

Fonte: autoria própria.

2.4 FUNÇÕES PARA BUSCA DE TERMOS CHAVES

Para encontrar os termos chaves no arquivo de log foi necessário criar variáveis constantes com determinados valores que retornam informações específicas relacionadas a potência de sinal e ao tipo de tecnologia na qual se encontra o aparelho móvel, elas são utilizadas pela função de busca que as procura no arquivo de texto e salva seus valores em uma determinada lista, assim como salva o tempo em que foi encontrada essa palavra em outra lista na posição correspondente para que, posteriormente. seja possível construir os gráficos.

2.4.1 Classes Buscadas do Android

As constantes fazem referência às classes encontradas na própria documentação Android, que pode ser conferida em (PLATAFORMA, 2023a) e tem significados importantes para o funcionamento do sistema, como visto na Figura 24, no que diz respeitos às redes móveis, sendo as principais:

- getRilDataRadioTechnology: Retorna quais as tecnologias estão sendo usadas no rádio do telefone para os serviços de dados (PLATAFORMA, 2023b).
- getRilVoiceRadioTechnology: Retorna quais as tecnologias estão sendo usadas no rádio do telefone para os serviços de voz (PLATAFORMA, 2023b).
- CellSignalStrength: Retorna as tags que fazem contém valores das potências relativos aos sinais de referência nas tecnologias de rádio, como é descrito em Documentation (2023, tradução nossa).
- SIPMSG: Retorna as mensagens SIP entre o dispositivo e a rede, de acordo como é dito em GSMA (2014, tradução nossa).

Figura 24 – Constantes da Documentação Android

```
SIGNAL_TYPE = 'getRilDataRadioTechnology'
SIGNAL_VOICE = 'getRilVoiceRadioTechnology'
SIGNAL_STRENGTH_LTE = 'mLte=CellSignalStrengthLte'
SIGNAL_STRENGTH_WCDMA = 'mWcdma=CellSignalStrengthWcdma'
RSRP = 'rsrp'
RSCP = 'rscp'
RSSI = 'rssi'
RSRQ = 'rsrq'
SIPMSG1 = 'SIPMSG[1]: ['
SIPMSG0 = 'SIPMSG[0]: ['
TECH_X = 'Tempo'
TECH_Y = 'Tecnologia'
SIGNAL_X = 'Tempo'
SIGNAL_Y = 'Potência do Sinal (dBm)'
SIGNAL_Y_RSRQ = 'Potência do Sinal (dB)'
RADIO = 'radio'
PROCESS_1 = '10038'
PROCESS_2 = '1000'
MOBILE = 'MobileStatusTracker'
NETWORK_CONTROLLER = 'NetworkController.MobileSignalController'
```

Fonte: autoria própria.

Já as constantes relacionadas à potência do sinal também são declaradas como variáveis para a construção dos gráficos relacionados, como é visto na Figura 25. Os valores usados para a potência do sinal podem ser encontrados em (3GPP, 2017) e (3GPP, 2010).

2.4.2 Função de Busca de Classes

Uma das principais funções do programa é a função search_in_log_file(), Figura 26 que também está escrita detalhadamente no Apêndice C, ela é responsável por procurar palavras

Figura 25 – Constantes da Documentação do 3GPP

```
#Valores para RSRP e RSCP
VERY_POOR = -95
POOR = -85
MODERATE = -75
GOOD = -60
#Valores para RSSI LTE
VERY_POOR_RSSI_LTE = -95
POOR_RSSI_LTE = -85
MODERATE_RSSI_LTE = -75
GOOD_RSSI_LTE = -65
#Valores para RSRQ LTE
VERY_POOR_RSRQ_LTE = -25
POOR_RSRQ_LTE = -20
MODERATE_RSRQ_LTE = -15
GOOD_RSRQ_LTE = -10
```

Fonte: autoria própria.

específicas, como as descritas na seção anterior, conforme for encontrando, ela vai anexando nos dicionários criados para guardar os valores para cada gráfico.

O processo de saber se deve ou não adicionar a palavra na lista se deve ao fato de que a função já procura as constantes que foram previamente definidas nas linhas do arquivo de texto, através se elas estiverem naquela linha é anexado a lista juntamente com o tempo associado.

Se o if proposto para cada gráfico for verdadeiro, ocorre uma append(), ou seja, uma adição nos dicionários relacionados ao gráfico, como é visto na Figura 27, esse processo é feito para cada um dos gráficos. A função search_in_log_file() é diretamente integrada com a função do dashboard que é responsável por fazer o upload do arquivo de log a save_files().

Figura 26 – Dicionários e função de busca

```
#Funções para leitura e busca de termos chaves
def search_in_log_file(user_input):
    data = {
        'tech': [],
        'tech_time': [],
        'tech_voice': [],
        'tech_time_voice': [],
        'signal_lte': [],
        'signal_time_lte': [],
        'signal_wcdma': [],
        'signal_time_wcdma': [],
        'signal_rssi_lte': [],
        'signal_rssi_time_lte': [],
        'signal_rsrq_lte': [],
        'signal_rsrq_time_lte': [],
        'sipmsg_left': [],
        'sipmsg_right': [],
        'sipmsg_time': [],
        'sipmsg_arrow': [],
        'sipmsg_info': []
    }
    # Algoritimo para procurar palavras chaves
    user_input = user_input.replace('\"', "")
    assert os.path.exists(user_input), "Não foi possível encontrar o arquivo:" + str(user_input)
    f = open(user_input, 'r', encoding="ISO-8859-1")
    for line in f:
        for key_word in (key_word for key_word in key_words_list if key_word in line):
            usual_searchs.append(line)
```

Fonte: autoria própria.

Figura 27 – Adição das Listas

```
        if SIGNAL_STRENGTH_LTE in line:
            signal_lte, signal_time_lte = make_signal_lte(line)
            signal_rssi_lte, signal_rssi_time_lte = make_signal_rssi_lte(line)
            signal_rsrq_lte, signal_rsrq_time_lte = make_signal_rsrq_lte(line)
            if signal_lte:
                data['signal_lte'].append(signal_lte)
                data['signal_time_lte'].append(signal_time_lte)
            if signal_rssi_lte:
                data['signal_rssi_lte'].append(signal_rssi_lte)
                data['signal_rssi_time_lte'].append(signal_rssi_time_lte)
            if signal_rsrq_lte:
                data['signal_rsrq_lte'].append(signal_rsrq_lte)
                data['signal_rsrq_time_lte'].append(signal_rsrq_time_lte)
```

Fonte: autoria própria.

3 DESENVOLVIMENTO DO DASHBOARD

3.1 FUNÇÕES PARA CRIAÇÃO DOS GRÁFICOS E TABELAS

Para o desenvolvimento do dashboard primeiramente foi necessário criar as funções que construíram os gráficos e tabelas. O programa pode ser dividido em três tipos de funções principais para criar gráficos: Gráficos de Tecnologia, Gráficos da Potência do Sinal e Tabela das mensagens SIP.

3.1.1 Gráficos de Tecnologia

Para gerar o gráfico de tecnologias que irão tanto analisar as tecnologias das redes móveis como GSM, WCDMA, UMTS, LTE e etc para dados quanto para voz são feitas duas funções: a função make_signal_radio_tech_line(), Figura 28, que é responsável por extrair da linha onde a constante getRilDataRadioTechnology é mencionada as siglas das tecnologias usadas e a função df_chart_radio_techlogy(), Figura 29, que gera o gráfico através da geração de um Dataframe, ou seja uma tabela, da qual foi possível gerar um gráfico de forma que em cada espaço de tempo seja exposto em qual tecnologia estava o aparelho.

Figura 28 – Função para criar as listas de Tecnlogia

```
#Lista das Tecnologia de Dados
def make_signal_radio_tech_line(line: object) -> object:
    splitted = line.split()
    idx = find_index(splitted, SIGNAL_TYPE)
    tech_time = splitted[0] + '#' + splitted[1]
    tech = splitted[idx][26:].replace(',', '')
    return tech_time, tech
```

Fonte: autoria própria.

Figura 29 – Função para criar os gráficos de Tecnologia

```
#Plotar Gráfico da Tecnologia de Dados
def df_chart_radio_techlogy(tech_time, tech):
    tech_table = [tech_time, tech]
    tech_dataframe = pd.DataFrame(tech_table).transpose()
    tech_dataframe.columns = [TECH_X, TECH_Y]
    tech_dataframe['Tecnologia de Dados'] = 1
    tech_chart = px.histogram(tech_dataframe, x=TECH_X, y='Tecnologia de Dados', color=TECH_Y,
                              title='Tempo x Tecnologia de Dados')
    tech_chart.update_xaxes(categoryorder='category ascending')
    tech_chart.update_yaxes(dtick=1, range=[0, 1])
    return tech_chart
```

Fonte: autoria própria.

3.1.2 Gráficos de Sinal

Para os gráficos de sinal, que vão analisar fatores como potência recebida, intensidade e qualidade do sinal através dos parâmetros RSRP, RSSI, RSRQ E RSCP, conforme está no Apêndice C. Semelhante ao que ocorre nos gráficos de tecnologia, nos gráficos de sinal também é preciso construir uma função que irá extrair de cada lista a constantes da classe CellSignalStrength feita pela função make_signal_lte(), Figura 30.

Figura 30 – Função para criar as listas de Sinal

```
#Listas Sinal LTE
def make_signal_lte(line):
    words = (MOBILE, NETWORK_CONTROLLER)
    for word in words:
        if word in line:
            splitted = line.split()
            idx = find_index(splitted, RSRP)
            if splitted[idx][5] == '-':
                signal_time_lte = splitted[0] + '#' + splitted[1]
                signal_lte = splitted[idx][5:]
                return signal_lte, signal_time_lte
    return (None,None)
```

Fonte: autoria própria.

Já para a construção dos gráficos é feito também um Dataframe, na função df_chart_signal_lte() onde para cada intervalo entre as constantes de sinal da Figura 31, é defininda uma coloração diferente para poder identificar os momentos em que o sinal está melhor ou pior de acordo com os parâmetros definidos em 3GPP (2010).

Figura 31 – Função para criar os gráficos de Sinal

```
#Plotar Grafico do RSRP
def df_chart_signal_lte(signal_lte, signal_time_lte):
    signal = list(map(int, signal_lte))
    if signal != []:
        signal_max = max(signal)
        signal_min = min(signal)

    signal_table = [signal_time_lte, signal]
    signal_dataframe = pd.DataFrame(signal_table).transpose()
    signal_dataframe.columns = [SIGNAL_X, SIGNAL_Y]
    signal_dataframe = signal_dataframe.sort_values(by=SIGNAL_X)

    if signal != []:
        signal_chart = px.line(signal_dataframe, x=SIGNAL_X, y=SIGNAL_Y,
                               title='Tempo x Potência do Sinal 4G (RSRP)')
        signal_chart.add_hrect(y0=GOOD, y1=GOOD + 60,
                               line_width=0, fillcolor="green", opacity=0.5)
        signal_chart.add_hrect(y0=MODERATE, y1=GOOD,
                               line_width=0, fillcolor="green", opacity=0.4)
        signal_chart.add_hrect(y0=POOR, y1=MODERATE,
                               line_width=0, fillcolor="yellow", opacity=0.4)
        signal_chart.add_hrect(y0=VERY_POOR, y1=POOR,
                               line_width=0, fillcolor="orange", opacity=0.4)
        signal_chart.add_hrect(y0=VERY_POOR - 30, y1=VERY_POOR,
                               line_width=0, fillcolor="red", opacity=0.4)
        signal_chart.update_yaxes(tick0=MODERATE, dtick=5,
                                  range=[signal_min - 1, signal_max + 1])
        return signal_chart
```

Fonte: autoria própria.

3.1.3 Tabela de Mensagens SIP

Para a construção da tabela das mensagens SIP faz uso da função make_sipmsg(), Figura 32, que adiciona em cada lista dentro da função as palavras contidas em cada coluna do arquivo txt, já que nesse tipo de arquivo a separação de colunas pode ser feita através de espaços em branco.

Figura 32 – Função para criar as listas das Mensagens SIP

```
#Lista das Mensagens SIP
def make_sipmsg(line: object) -> object:
    splitted = line.split()
    sipmsg_time = splitted[0] + '#' + splitted[1]
    if splitted[7] == '[-->]':
        sipmsg_left = '    ' + splitted[8]
        sipmsg_arrow = '-------->'
        sipmsg_right = '         | '
        sipmsg_info = splitted[9] + ' ' + splitted[10]
    if splitted[7] == '[<--]':
        sipmsg_right = '    ' + splitted[8]
        sipmsg_arrow = '<--------'
        sipmsg_left = '         | '
        sipmsg_info = splitted[9] + ' ' + splitted[10]
    return sipmsg_time, sipmsg_left, sipmsg_arrow, sipmsg_right, sipmsg_info
```

Fonte: autoria própria.

Com as listas criadas, na função df_table_sipmsg() também é criado um Datafram, porém nesse caso o resultado não é um gráfico e sim uma tabela criada através da função do Plotly create_table(), como vista na Figura 33.

Figura 33 – Função para criar a Tabela das Mensagens SIP

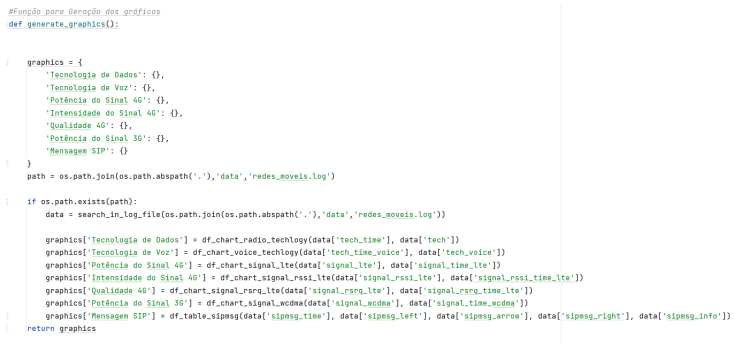
```
#Plotar Tabela das Mensagens SIP
def df_table_sipmsg(sipmsg_time, sipmsg_left, sipmsg_arrow, sipmsg_right, sipmsg_info):
    sipmsg_table = [sipmsg_time, sipmsg_left, sipmsg_arrow, sipmsg_right, sipmsg_info]
    df_sipmsg = pd.DataFrame(sipmsg_table).transpose()
    df_sipmsg.columns = ['Tempo', 'UE', 'Direção', 'Rede', 'Info']
    df_sipmsg['  '] = '  '
    df_sipmsg['   '] = '   '
    df_sipmsg['    '] = '    '
    df_sipmsg['     '] = '     '
    df_sipmsg['      '] = '      '
    df_sipmsg['       '] = '       '
    fig = ff.create_table(df_sipmsg, colorscale=[[0, '#800080'], [.5, '#d8b1d4'], [1, '#ecd8e9']])
    return fig
```

Fonte: autoria própria.

3.2 FUNÇÕES PARA CRIAÇÃO DO DASHBOARD

Os gráficos são exportados para outras funções do dashboard através da função generate_graphics(), Figura 34 que retorna todos os gráficos montados para as funções que montam atributos do dashboard como Layouts, Callbacks e Menu, essas funções estão descritas no Apêndice D.

Figura 34 – Função que envia os gráficos para o Dashboard



```
#Função para Geração dos gráficos
def generate_graphics():

    graphics = {
        'Tecnologia de Dados': {},
        'Tecnologia de Voz': {},
        'Potência do Sinal 4G': {},
        'Intensidade do Sinal 4G': {},
        'Qualidade 4G': {},
        'Potência do Sinal 3G': {},
        'Mensagem SIP': {}
    }
    path = os.path.join(os.path.abspath('.'),'data','redes_moveis.log')

    if os.path.exists(path):
        data = search_in_log_file(os.path.join(os.path.abspath('.'),'data','redes_moveis.log'))

        graphics['Tecnologia de Dados'] = df_chart_radio_techlogy(data['tech_time'], data['tech'])
        graphics['Tecnologia de Voz'] = df_chart_voice_techlogy(data['tech_time_voice'], data['tech_voice'])
        graphics['Potência do Sinal 4G'] = df_chart_signal_lte(data['signal_lte'], data['signal_time_lte'])
        graphics['Intensidade do Sinal 4G'] = df_chart_signal_rssi_lte(data['signal_rssi_lte'], data['signal_rssi_time_lte'])
        graphics['Qualidade 4G'] = df_chart_signal_rsrq_lte(data['signal_rsrq_lte'], data['signal_rsrq_time_lte'])
        graphics['Potência do Sinal 3G'] = df_chart_signal_wcdma(data['signal_wcdma'], data['signal_time_wcdma'])
        graphics['Mensagem SIP'] = df_table_sipmsg(data['sipmsg_time'], data['sipmsg_left'], data['sipmsg_arrow'], data['sipmsg_right'], data['sipmsg_info'])
    return graphics
```

Fonte: autoria própria.

3.2.1 Callbacks

O callback é uma das funções principais do Dash, através dele que pode ser definido que quando o usuário fizer qualquer iteração com a página do navegador haverá uma resposta do programa.

Para cada gráfico é gerada uma função app.callback(), Figura 35, que através da função generate_graphics() recebe os dicionários que estão associados a cada gráfico, associa que naquela página aquele gráfico deve ser exibido, nessas funções também são atribuídos os ids desses gráficos que serão utilizados pelos arquivos de Menu e Layouts do Dashboard.

Figura 35 – Callbacks

```
from app import app
from dash.dependencies import Output, Input


@app.callback(
    Output('technology-data-graphic', 'figure'),
    Input('store', 'data'))
def update_data_technology(data):
    if not data:
        return {}
    return data['Tecnologia de Dados']

@app.callback(
    Output('technology-data-voice', 'figure'),
    Input('store', 'data'))
def update_voice_technology(data):
    if not data:
        return {}
    return data['Tecnologia de Voz']
```

Fonte: autoria própria.

3.2.2 Menu

No arquivo de Menu, menu.py, Figura 36, é definido em qual página ficará cada gráfico criando botões na barra superior da página para que o usuário possa transitar entre os gráficos através da tag id que é definida nos callbacks.

Figura 36 – Menu

```
from dash import html, dcc

layout = html.Div([

    html.Ul([
        html.Li([
            dcc.Link('Home', href='/', className='nav-link')
        ], className='nav-item'),
        html.Li([
            dcc.Link('Tecnologia de Dados', href='/technology-data', className='nav-link')
        ], className='nav-item'),
        html.Li([
            dcc.Link('Tecnologia de Voz', href='/technology-voice', className='nav-link')
        ], className='nav-item'),
        html.Li([
            dcc.Link('Potência do Sinal 4G', href='signal-lte', className='nav-link')
        ], className='nav-item'),
        html.Li([
            dcc.Link('Intensidade do Sinal 4G', href='signal-rssi-lte', className='nav-link')
        ], className='nav-item'),
        html.Li([
            dcc.Link('Qualidade do Sinal 4G', href='rsrq', className='nav-link')
        ], className='nav-item'),
        html.Li([
            dcc.Link('Potência do Sinal 3G', href='signal-wcdma', className='nav-link')
        ], className='nav-item'),
        html.Li([
            dcc.Link('Mensagens SIP', href='sip', className='nav-link')
        ], className='nav-item'),
```

Fonte: autoria própria.

3.2.3 Layouts

Para cada página do dashboard é feita uma função de Layout, Figura 37, através de funções da biblioteca do Dash. No layout é onde é definido atributos como título, alinhamento, tamanho da fonte e etc. Os gráficos que vão aparecer em cada página são identificados através da tag id daquele determinado gráfico.

Figura 37 – Layouts

```
from dash import html, dcc
from layouts.pages import menu, header, footer
#from utils.parser_utils import generate_graphics


layout = html.Div([
    menu.layout,
    header.layout,
    html.Div([
        html.Div([
            html.H2("Potência do Sinal 4G",style={'text-align':'center'})
        ]),
    ], className='form-control container-fluid'),
    html.Br(),
    html.Div([
        dcc.Graph(
            id='signal-lte',
            config={"displayModeBar": True},
        )
    ]),
    footer.layout
],className="container-fluid")
```

Fonte: autoria própria.

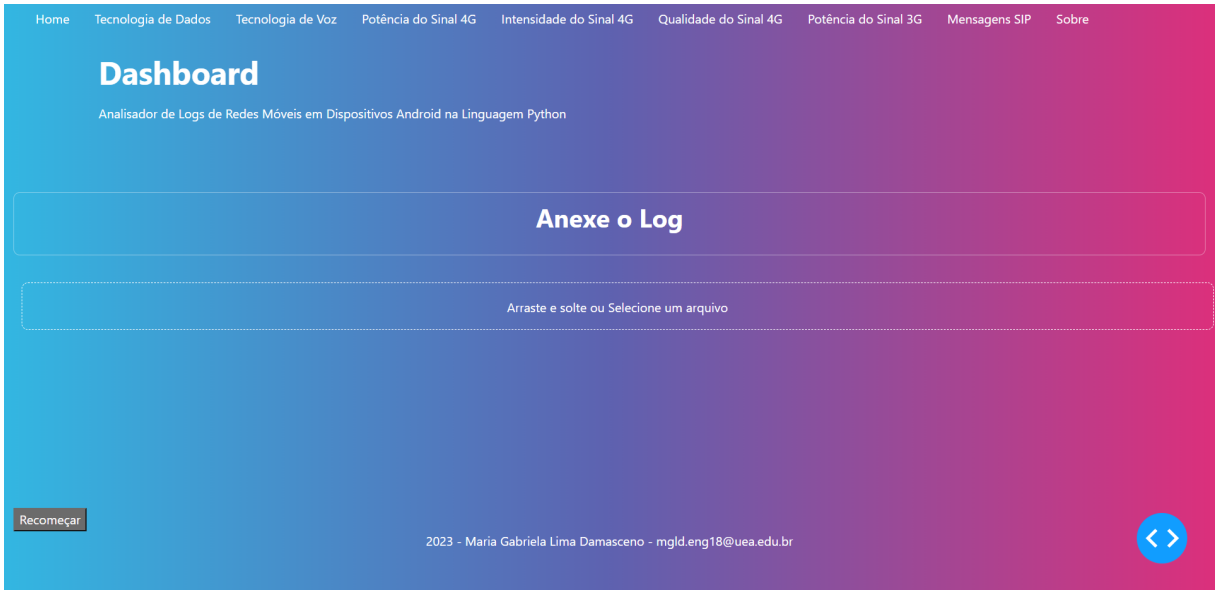
4 RESULTADOS

4.1 DASHBOARD

Ao final da função é gerado o Dashboard na porta 8050 de forma local em qualquer Navegador Web, digitando o http://localhost:8050/ já ocorre o direcionamento para a página inicial. Em cada gráfico existem botões interativos como botão de zoom, botão de mover e botão para download da imagem, esses botões auxiliam bastante na visualização dos gráficos pelo usuário que, utilizando eles, pode selecionar as curvas nos pontos em que achar mais pertinente para a sua análise, relatório técnico e etc.

Na página inicial, Home na Figura 38, é possível visualizar os botões na barra superior, que estão presentes em todas as páginas, uma caixa de upload para inserir o arquivo de log e um botão de Recomeçar que apaga os dados para que um novo arquivo possa ser enviado.

Figura 38 – Home



Fonte: autoria própria.

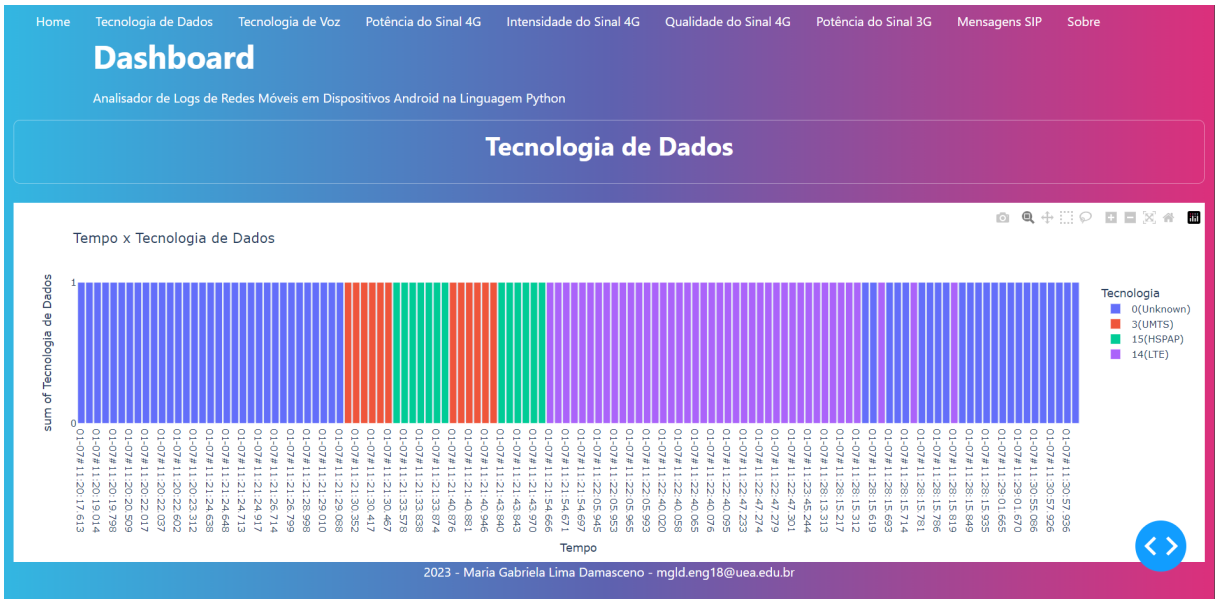
4.2 TECNOLOGIAS DAS DADOS E TECNOLOGIA DE VOZ

Nas páginas de Tecnologia de Dados e de Tecnologia de Voz, Figura 39 e Figura 40, são visualizados os gráficos em forma de colunas que demonstram que ao longo do tempo o aparelho estava conectado em várias tecnologias de redes móveis diferentes.

Com os gráficos e tabelas gerados no Dashboard é possível fazer análises sobre o arquivo de log utilizado, uma das primeiras observações é sobre em qual tipo de rede o aparelho móvel estava conectado em um determinado espaço de tempo através do Gráfico de Tecnologia de Dados, Figura 39, e do Gráfico de Tecnologia de Voz, Figura 40, também nesses gráficos outro ponto a ser observado é que nem sempre a tecnologia de dados é a mesma da tecnologia

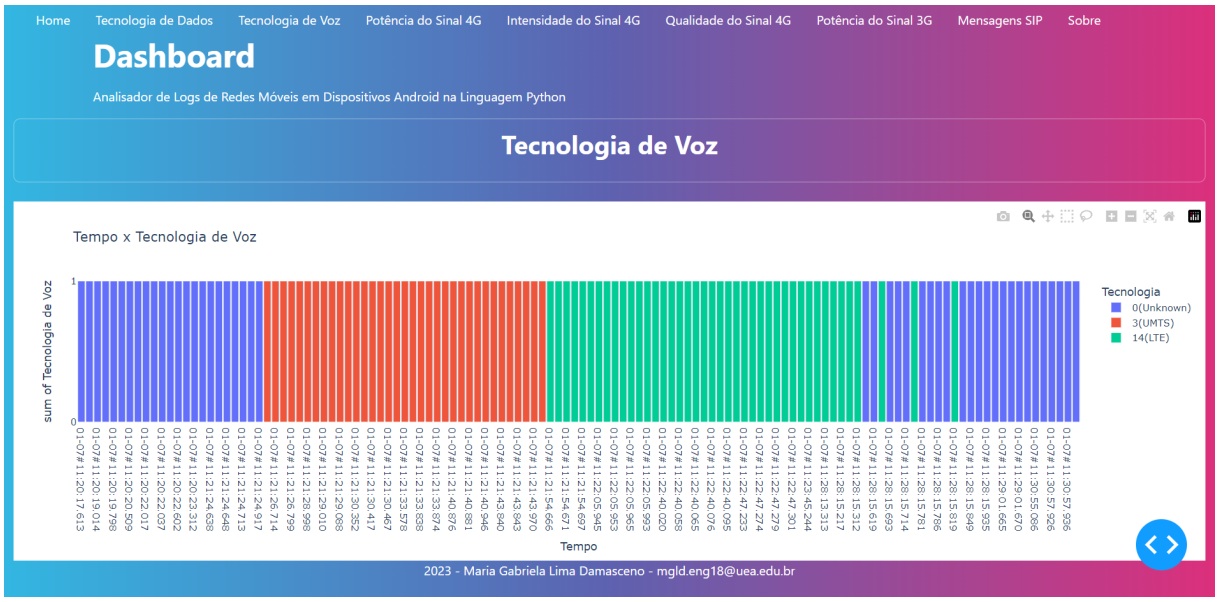
de voz, tomando por exemplo o espaço de tempo 11:21:43.840 do dia 07/01/2023 em que a tecnologia de dados de internet era HSPA enquanto a de voz era UMTS.

Figura 39 – Tecnologia de Dados



Fonte: autoria própria.

Figura 40 – Tecnologia de Voz



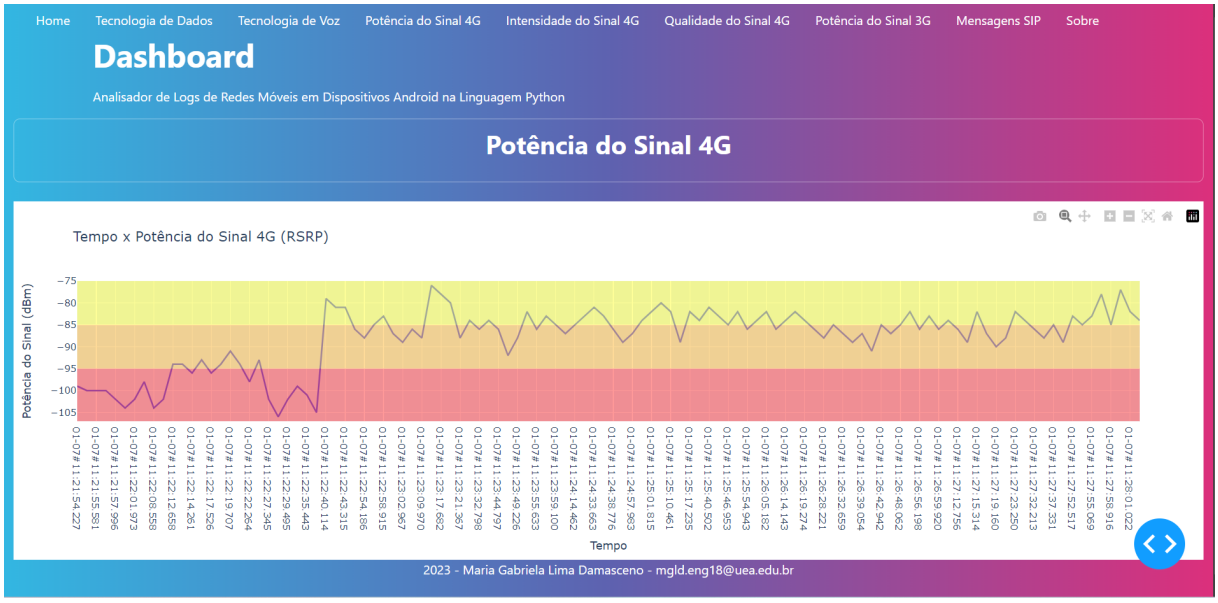
Fonte: autoria própria.

4.3 QUALIDADE DO SINAL 4G

Já nas páginas de Potência do Sinal 4G, Intensidade do Sinal 4G, Qualidade do Sinal 4G e Potência do Sinal 3G, nas figuras a seguir, são mostradas as curvas de cada sinal ao longo do tempo, onde as cores demonstram se o sinal de referência está forte, quando perto da cor verde, ou fraco, perto da cor vermelha.

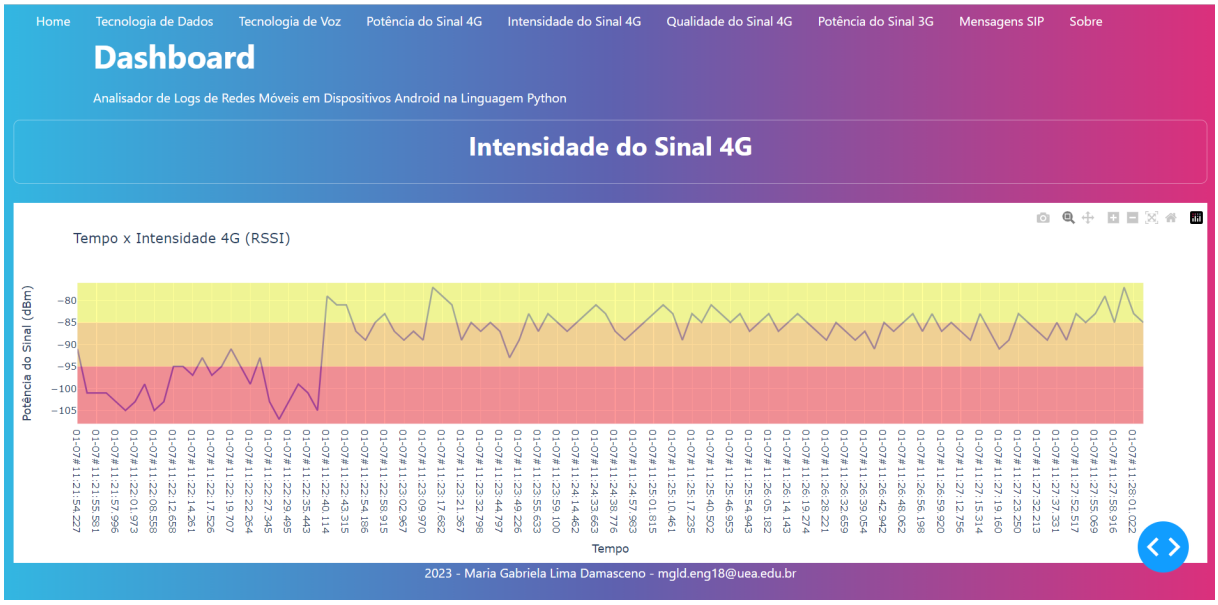
Nos gráficos de Potência do Sinal 4G, Figura 41, e Intensidade do Sinal 4G, Figura 42, é visto que a curva é muito semelhante, tendo pouquíssimos pontos de diferença, isso se deve ao fato de que o RSRP e o RSSI, como visto em (TELTONIKA, 2018), tem valores muito próximos. Essa semelhança entre os dois parâmetros reflete na qualidade do sinal, como visto no gráfico da Qualidade do Sinal 4G, Figura 43, já que o sinal RSRQ é diretamente proporcional às duas potência, devido a isso o gráfico da qualidade do sinal permanece com valores bastante estáveis.

Figura 41 – Potência do Sinal 4G



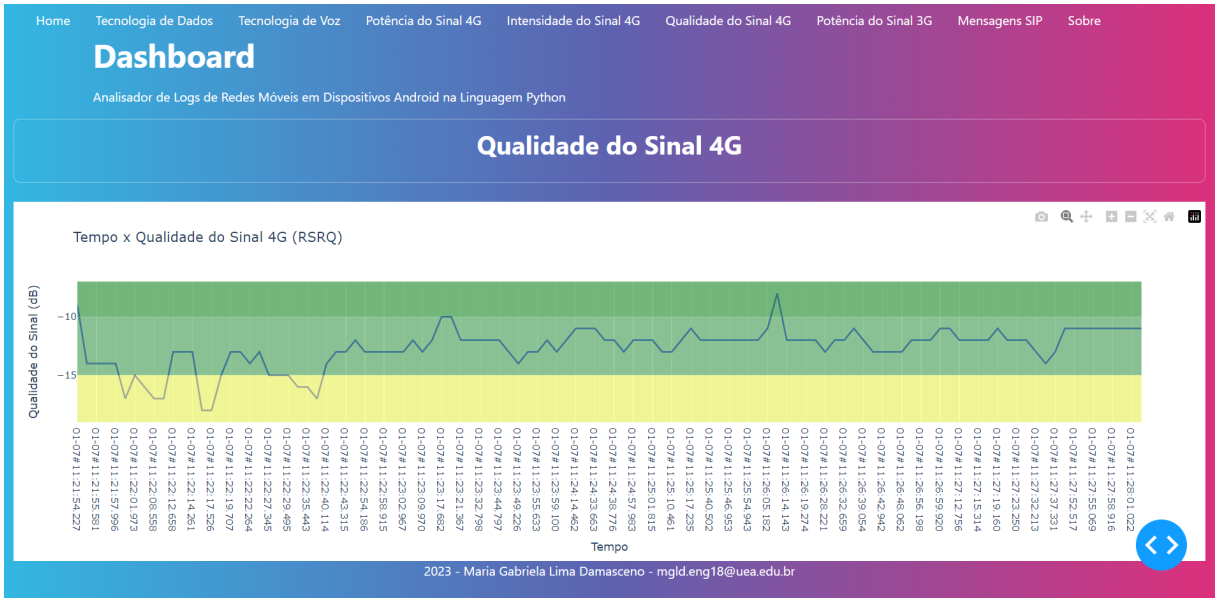
Fonte: autoria própria.

Figura 42 – Intensidade do Sinal 4G



Fonte: autoria própria.

Figura 43 – Qualidade do Sinal 4G



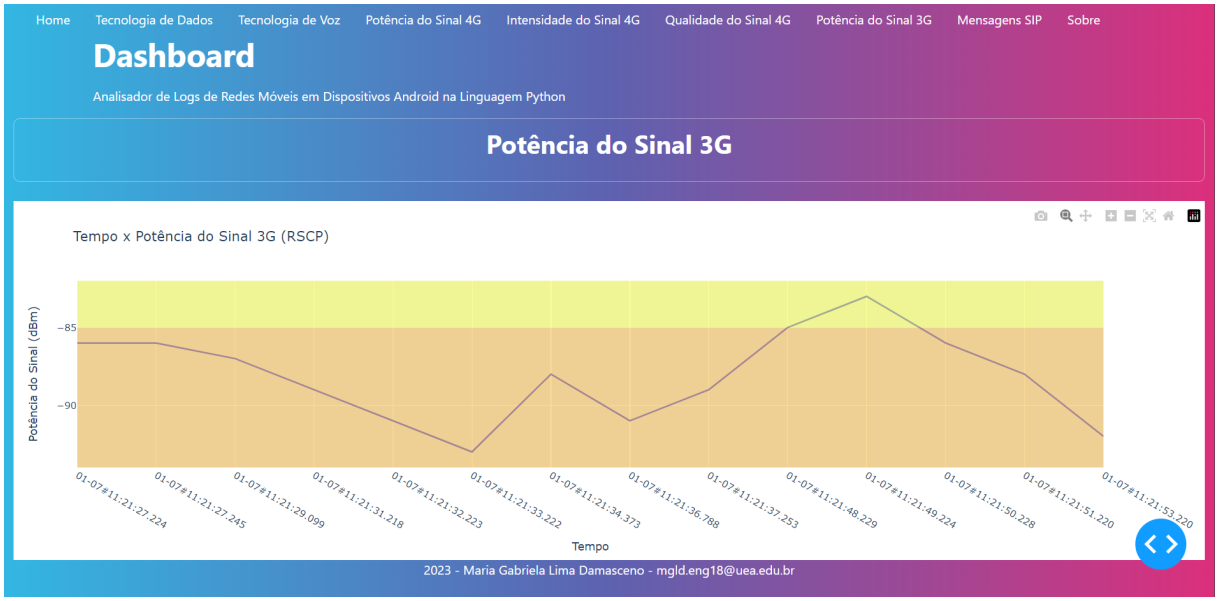
Fonte: autoria própria.

4.4 HANDOVER

Relacionando os gráficos de Potência do Sinal 4G, Figura 41, e Potência do Sinal 3G, Figura 44, é notório que que o gráfico do 4G começa no momento 11:21:54.227 enquanto o do 3G havia terminado em 11:21:53.220, nesse intervalo de tempo de segundos o processo que

ocorreu foi o de Handover entre a tecnologia 3G e 4G já que, como pode ser visualizado nos gráficos, a potência do sinal 3G estava caindo no momento em que houve a troca de tecnologia (KEYSIGHT, 2013).

Figura 44 – Potência do Sinal 3G



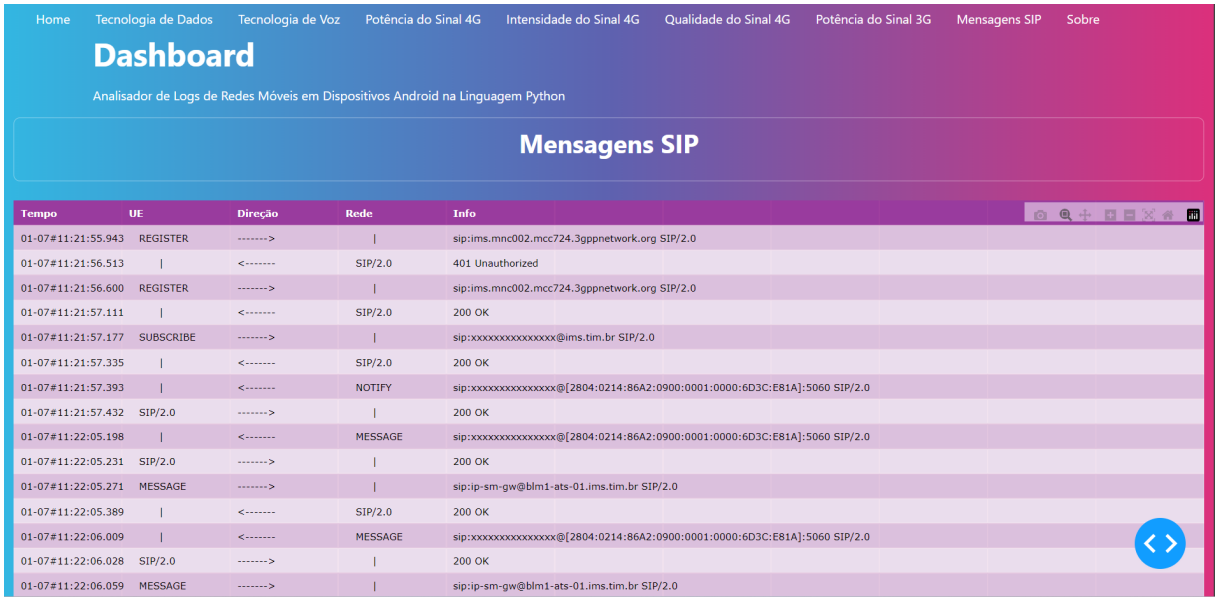
Fonte: autoria própria.

4.5 REGISTRO, SMS E CHAMADAS DE VOZ EM IMS

Na página das Mensagens SIP, Figura 45, é mostrada a tabela que mostra em cada coluna o tempo da mensagem SIP, a ação do lado do equipamento do usuário (UE), as setas da direção da mensagem, a ação do lado da rede e a informação em cada mensagem.

Através da tabela pode-se encontrar as funcionalidades da rede IMS que o aparelho móvel fez uso, entre 11:21:55.943 e 11:21:57.432 houve o registro na rede IMS que começa sempre com o celular tentando se registrar na rede e sendo desautorizado na primeira tentativa e depois conseguindo se registrar (GSMA, 2014). Após isso, entre 11:22:05.271 e 11:22:06.184 ocorre o processo de envio de mensagens SMS pela mensagem MESSAGE. Por fim, entre 11:22:53.384 e 11:23:44.598 ocorre uma chamada de voz, Figura 46 que começa no INVITE e termina no BYE.

Figura 45 – Mensagens SIP: Registro e SMS



Tempo	UE	Direção	Rede	Info
01-07#11:21:55.943	REGISTER	------->	|	sip:ims.mnc002.mcc724.3gppnetwork.org SIP/2.0
01-07#11:21:56.513	|	<-------	SIP/2.0	401 Unauthorized
01-07#11:21:56.600	REGISTER	------->	|	sip:ims.mnc002.mcc724.3gppnetwork.org SIP/2.0
01-07#11:21:57.111	|	<-------	SIP/2.0	200 OK
01-07#11:21:57.177	SUBSCRIBE	------->	|	sip:xxxxxxxxxxxxxxx@ims.tim.br SIP/2.0
01-07#11:21:57.335	|	<-------	SIP/2.0	200 OK
01-07#11:21:57.393	|	<-------	NOTIFY	sip:xxxxxxxxxxxxxxx@[2804:0214:86A2:0900:0001:0000:6D3C:E81A]:5060 SIP/2.0
01-07#11:21:57.432	SIP/2.0	------->	|	200 OK
01-07#11:22:05.198	|	<-------	MESSAGE	sip:xxxxxxxxxxxxxxx@[2804:0214:86A2:0900:0001:0000:6D3C:E81A]:5060 SIP/2.0
01-07#11:22:05.231	SIP/2.0	------->	|	200 OK
01-07#11:22:05.271	MESSAGE	------->	|	sip:ip-sm-gw@blm1-ats-01.ims.tim.br SIP/2.0
01-07#11:22:05.389	|	<-------	SIP/2.0	200 OK
01-07#11:22:06.009	|	<-------	MESSAGE	sip:xxxxxxxxxxxxxxx@[2804:0214:86A2:0900:0001:0000:6D3C:E81A]:5060 SIP/2.0
01-07#11:22:06.028	SIP/2.0	------->	|	200 OK
01-07#11:22:06.059	MESSAGE	------->	|	sip:ip-sm-gw@blm1-ats-01.ims.tim.br SIP/2.0

Fonte: autoria própria.

Figura 46 – Mensagens SIP: Chamadas de Voz

01-07#11:22:53.384	INVITE	------->	|	sip:xxxxxxxxxxxxxxx;phone-context=ims.tim.br@ims.tim.br;user=phone SIP/2.0
01-07#11:22:53.547	|	<-------	SIP/2.0	100 Trying
01-07#11:22:55.240	|	<-------	SIP/2.0	183 Session
01-07#11:22:55.253	PRACK	------->	|	sip:[xxxx:xxxx:xxxx:xxxx]:5060;Hpt=nw_7f_63b9641d_11f1c32b_ex_8ea2_16;CxtId=3;TRC=ffffffff-ffffffff SIP/2.0
01-07#11:22:55.395	|	<-------	SIP/2.0	200 OK
01-07#11:23:00.048	|	<-------	SIP/2.0	183 Session
01-07#11:23:00.060	PRACK	------->	|	sip:[xxxx:xxxx:xxxx:xxxx]:5060;Hpt=nw_7f_63b9641d_11f1c32b_ex_8ea2_16;CxtId=3;TRC=ffffffff-ffffffff SIP/2.0
01-07#11:23:00.220	|	<-------	SIP/2.0	200 OK
01-07#11:23:00.493	|	<-------	SIP/2.0	180 Ringing
01-07#11:23:00.501	PRACK	------->	|	sip:[xxxx:xxxx:xxxx:xxxx]:5060;Hpt=nw_7f_63b9641d_11f1c32b_ex_8ea2_16;CxtId=3;TRC=ffffffff-ffffffff SIP/2.0
01-07#11:23:00.669	|	<-------	SIP/2.0	200 OK
01-07#11:23:00.712	|	<-------	SIP/2.0	180 Ringing
01-07#11:23:00.721	PRACK	------->	|	sip:[xxxx:xxxx:xxxx:xxxx]:5060;Hpt=nw_7f_63b9641d_11f1c32b_ex_8ea2_16;CxtId=3;TRC=ffffffff-ffffffff SIP/2.0
01-07#11:23:00.903	|	<-------	SIP/2.0	200 OK
01-07#11:23:07.697	|	<-------	SIP/2.0	200 OK
01-07#11:23:07.708	ACK	------->	|	sip:[xxxx:xxxx:xxxx:xxxx]:5060;Hpt=nw_7f_63b9641d_11f1c32b_ex_8ea2_16;CxtId=3;TRC=ffffffff-ffffffff SIP/2.0
01-07#11:23:44.598	BYE	------->	|	sip:[xxxx:xxxx:xxxx:xxxx]:5060;Hpt=nw_7f_63b9641d_11f1c32b_ex_8ea2_16;CxtId=3;TRC=ffffffff-ffffffff SIP/2.0
01-07#11:23:45.074	|	<-------	SIP/2.0	200 OK

Fonte: autoria própria.

CONCLUSÃO

Este trabalho tinha o intuito de desenvolver uma ferramenta que pudesse auxiliar os Engenheiros de Telecomunicações na análise de Logs de Redes Móveis de dispositivos Android, visto que atualmente esse processo é bastante demorado e manual. Foram apresentados conceitos de Redes Móveis, Arquitetura Android e Linguagem Python para que a construção da ferramenta fosse bem fundamentada.

Ao decorrer do trabalho foi sendo definido que a ferramenta Analisador de Logs seria um Dashboard que mostraria gráficos e tabelas presentes no log que fizessem referência a fatores como potência, intensidade e qualidade do sinal, tipo de tecnologia utilizada e mensagens trocadas entre o aparelho e a rede através do protocolo SIP.

Com o Dashboard funcionando localmente em qualquer navegador Web é possível que qualquer usuário, com conhecimentos em Redes Móveis e Android, em posse do código-fonte, possa utilizá-lo. Com o Analisador de Logs, eventos que acontecem em um pequeno espaço de tempo, como o Handover, podem ser mais facilmente observados e identificados com o auxílio dos gráficos, assim como conceitos já pré-estabelecidos como a relação entre os parâmetros de sinal do 4G LTE: RSRP, RSSI e RSRQ também podem ser validados.

Para trabalhos futuros pode destacar-se a possibilidade de que a ferramenta seja testada por outros Engenheiros de Telecomunicações para obter o resultado do quanto ela pode aumentar o desempenho das atividades de análises de log e também a comparação com outras ferramentas ou programada de análises logs voltadas para a área de Redes Móveis.

Com a evolução da tecnologia cada vez mais rápida é preciso que as análises do funcionamento de seus sistemas também acompanhem essa velocidade e entreguem cada vez mais qualidade e evidências em relação aos eventos que ocorrem. Desta forma, o dashboard intitulado de Analisador de Logs pode cumprir o seu objetivo principal de servir como uma ferramenta que ande lado a lado com o Engenheiro de Telecomunicações.

REFERÊNCIAS

3GPP. ***Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer - Measurements (3GPP TS 36.214 version 9.1.0 Release 9)***. 2010. Disponível em: <https://www.etsi.org/deliver/etsi_ts/136200_136299/136214/09.01.00_60/ts_136214 v090100p.pdf>. Acesso em: 21 mar. 2023.

3GPP. ***Single Radio Voice Call Continuity (SRVCC). Stage 2 (3GPP TS 23.216 versão 8.8.0 Release 8)***. 2012. Disponível em: <https://www.etsi.org/deliver/etsi_ ts/123200_123299/123216/ 08.08.00_60/ts_123216 v080800p.pdf>. Acesso em: 10 set. 2022.

3GPP. ***Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) procedures in idle mode. (3GPP TS 36.304 version 13.4.0 Release 13)***. 2017. Disponível em: <https://www.etsi.org/deliver/etsi_ts/136300_136399/136304/13.04.00_60/ts_136304 v130400p.pdf>. Acesso em: 21 mar. 2023.

3GPP. ***Service requirements for the 5G system***. 2023. Disponível em: <https://portal.3gpp.org/desktopmodules/Specifications/Specification Details.aspx?specificationId=3107>. Acesso em: 21 mar. 2023.

ABDALLA, I.; VENKATESAN, S. Notification based s-cscf load balancing in ims networks. In: IEEE. ***2011 Wireless Telecommunications Symposium (WTS)***. [S.l.], 2011. p. 1–5.

AHMADI, S. ***LTE-Advanced: a practical systems approach to understanding 3GPP LTE releases 10 and 11 radio access technologies***. [S.l.]: Academic Press, 2013.

ANDRADE, P. ***IP Multimedia Subsystem. Pólis de Tecnologia***. 2012. Disponível em: ⟨https://polisdetecnologia.com.br/ip-multimedia-subsystem/index.html⟩. Acesso em: 10 set. 2022.

ANDROID, D. ***Documentação para desenvolvedores de App***. 2022. Disponível em: ⟨https://developer.android.com/docs?hl=pt-br⟩. Acesso em: 12 set. 2022.

BITTENCOURT, J. ***O que é logcat e como visualizar logs no Android***. 2022. Disponível em: ⟨https://www.alura.com.br/artigos/o-que-e-logcat-como-visualizar-logs-android⟩. Acesso em: 12 set. 2022.

CONNECT, V. C. ***Collecting Logs from Android Devices***. 2021. Disponível em: ⟨https://kb.vmware.com/s/article/2960948?lang=pt_BR⟩. Acesso em: 21 mar. 2023.

CORDEIRO, G. R. 4g: Quarta geração de telefonia móvel. ***Universidade Federal do Paraná***, 2012.

DEVELOPERS, A. ***Arquitetura da plataforma***. 2023. Disponível em: ⟨https://developer. android.com/guide/platform?hl=pt-br\#:~:text=A\\%20plataforma\\%20Android\\%

20fornece\\%20as,e\\%203D\\%20no\\%20seu\\%20aplicativo.⟩ Acesso em: 18 mar. 2023.

DEVELOPERS, A. ***Escrever e visualizar registros com o Logcat***. 2023. Disponível em: ⟨https://developer.android.com/studio/debug/am-logcat?hl=pt-br⟩. Acesso em: 21 mar. 2023.

DOCUMENTATION, A. ***CellSignalStrength***. 2023. Disponível em: ⟨https://developer.android.com/reference/android/telephony/CellSignalStrength⟩. Acesso em: 21 mar. 2023.

ELECTRONICS. ***LTE SRVCC: Single Radio Voice Call Continuity***. 2022. Disponível em: ⟨https://www.electronics-notes.com/articles/connectivity/4g-lte-long-term-evolution/what-is-srvcc-single-radio-voice-call-continuity.php⟩. Acesso em: 9 set. 2022.

FILHO, L. H. B. ***Como criar Dashboards com Python?*** 2022. Disponível em: ⟨https://analisemacro.com.br/data-science/dicas-de-rstats/como-criar-dashboards-com-o-python/\#:~:text=A\\%20biblioteca\\%20Dash\\%20\\%C3\\%A9\\%20constru\\%C3\\%ADda,de\\%20acordo\\%20com\\%20o\\%20usu\\%C3\\%A1rio⟩. Acesso em: 11 mar. 2023.

GSMA. ***DNS and ENUM Guidelines for Service Providers and GRX and IPX Providers Version 10.0***. 2014. Disponível em: ⟨https://www.gsma.com/newsroom/wp-content/uploads/IR.67-v10.0.pdf⟩. Acesso em: 21 mar. 2023.

KEYSIGHT. ***SRVCC Handover Overview***. 2013. Disponível em: ⟨https://rfmw.em.keysight.com/rfcomms/refdocs/gsmgprs/gprsla_gen_bse_srvcc_handover.html⟩. Acesso em: 9 set. 2022.

MATHEUS, Y. ***O modelo OSI e suas camadas***. 2018. Disponível em: ⟨https://www.electronics-notes.com/articles/connectivity/4g-lte-long-term-evolution/what-is-srvcc-single-radio-voice-call-continuity.php⟩. Acesso em: 20 set. 2022.

MATPLOTLIB. ***Matplotlib***. 2022. Disponível em: ⟨https://matplotlib.org/stable/api/index.html⟩. Acesso em: 10 set. 2022.

MENDES, J. R. R. 5g: A quinta geração. In: IEEE. ***Faculdade de Engenharia - Universidade do Porto, Portugal***. [S.l.], 2013. p. 40–45.

MOBILE, R. ***Rakuten Mobile Successfully Verifies Data Transfer on 5G Standalone Mobile Network***. 2021. Disponível em: ⟨https://corp.mobile.rakuten.co.jp/english/news/press/2021/0712_01/⟩. Acesso em: 21 mar. 2023.

PISA, P. S. ***SIP (Session Initial Protocol)***. 2008. Disponível em: ⟨https://www.gta.ufrj.br/ensino/eel879/Anos-anteriores/2008-2/trabalhos_vf/pisa/OProtocolo.html⟩. Acesso em: 20 set. 2022.

PLATAFORMA, A. ***Android Code Search***. 2023. Disponível em: ⟨https://cs.android.com/⟩. Acesso em: 21 mar. 2023.

PLATAFORMA, A. ***Atualizações para restrições de interface não SDK no Android 12***. 2023. Disponível em: ⟨https://developer.android.com/about/versions/12/non-sdk-12?hl=pt-br⟩. Acesso em: 21 mar. 2023.

PLOTLY. ***Dash Python User Guide***. 2023. Disponível em: ⟨https://dash.plotly.com/⟩. Acesso em: 11 mar. 2023.

PYDATA. ***Pandas***. 2022. Disponível em: ⟨https://pandas.pydata.org/about/governance.html⟩. Acesso em: 10 set. 2022.

PYTHON. ***License Python***. 2022. Disponível em: ⟨https://docs.python.org/pt-br/3/license.html⟩. Acesso em: 10 set. 2022.

PYTHON. ***Logo Python***. 2022. Disponível em: ⟨https://www.python.org/community/logos/⟩. Acesso em: 9 set. 2022.

RFC. ***SIP: Session Initiation Protocol***. 2008. Disponível em: ⟨https://www.rfc-editor.org/rfc/rfc3261⟩. Acesso em: 20 set. 2022.

RIBEIRO, L. ***Introdução à Biblioteca Pandas***. 2020. Disponível em: ⟨https://medium.com/tech-grupozap/introdu\\%C3\\%A7\\%C3\\%A3o-a-biblioteca-pandas-89fa8ed4fa38⟩. Acesso em: 9 set. 2022.

SANTANA, Í. L. D. C. Do 1g ao 5g: Evolucao das redes de telefonia movel. ***Universidade Federal do Reconcavo da Bahia***, 2016.

STUDIO, A. ***Notas das versões do SDK Plataform Tools***. 2023. Disponível em: ⟨https://developer.android.com/studio/releases/platform-tools?hl=pt-br⟩. Acesso em: 21 mar. 2023.

TECHTUDO. ***Afinal, o que é Android?*** 2011. Disponível em: ⟨https://www.techtudo.com.br/noticias/2011/01/afinal-o-que-e-android.ghtml⟩. Acesso em: 18 mar. 2023.

TELECO. ***Arquitetura da Rede 4G***. 2022. Disponível em: ⟨https://www.teleco.com.br/tutoriais/tutorialltecdr/pagina_1.asp\#:~:text=A\\%20arquitetura\\%20da\\%20rede\\%204G,optimiza\\%C3\\%A7\\%C3\\%A3o\\%20do\\%20desempenho\\%20da\\%20rede⟩. Acesso em: 10 set. 2022.

TELECO. ***SIP um Protocolo da camada de sessão***. 2022. Disponível em: ⟨https://www.teleco.com.br/tutoriais/tutorialvoipsip/pagina_2.asp\#:~:text=O\\%20SIP\\%20\\%C3\\%A9\\%20um\\%20protocolo\\%20da\\%20camada\\%20de\\%20sess\\%C3\\%A3o\\%20do,SCHULZRINNEQ\\%20e\\%20CAMARILLO\\%20et\\%20al\\%2C⟩. Acesso em: 20 set. 2022.

TELECO. ***Tecnologias de Celular: 5G***. 2022. Disponível em: ⟨https://www.teleco.com.br/5g_tecnologia.asp⟩. Acesso em: 21 mar. 2023.

TELECO. ***HSPA e WiMax Móvel I: Tecnologias***. 2023. Disponível em: ⟨https://www.teleco.com.br/tutoriais/tutorialhspawimax1/pagina_2.asp⟩. Acesso em: 11 mar. 2023.

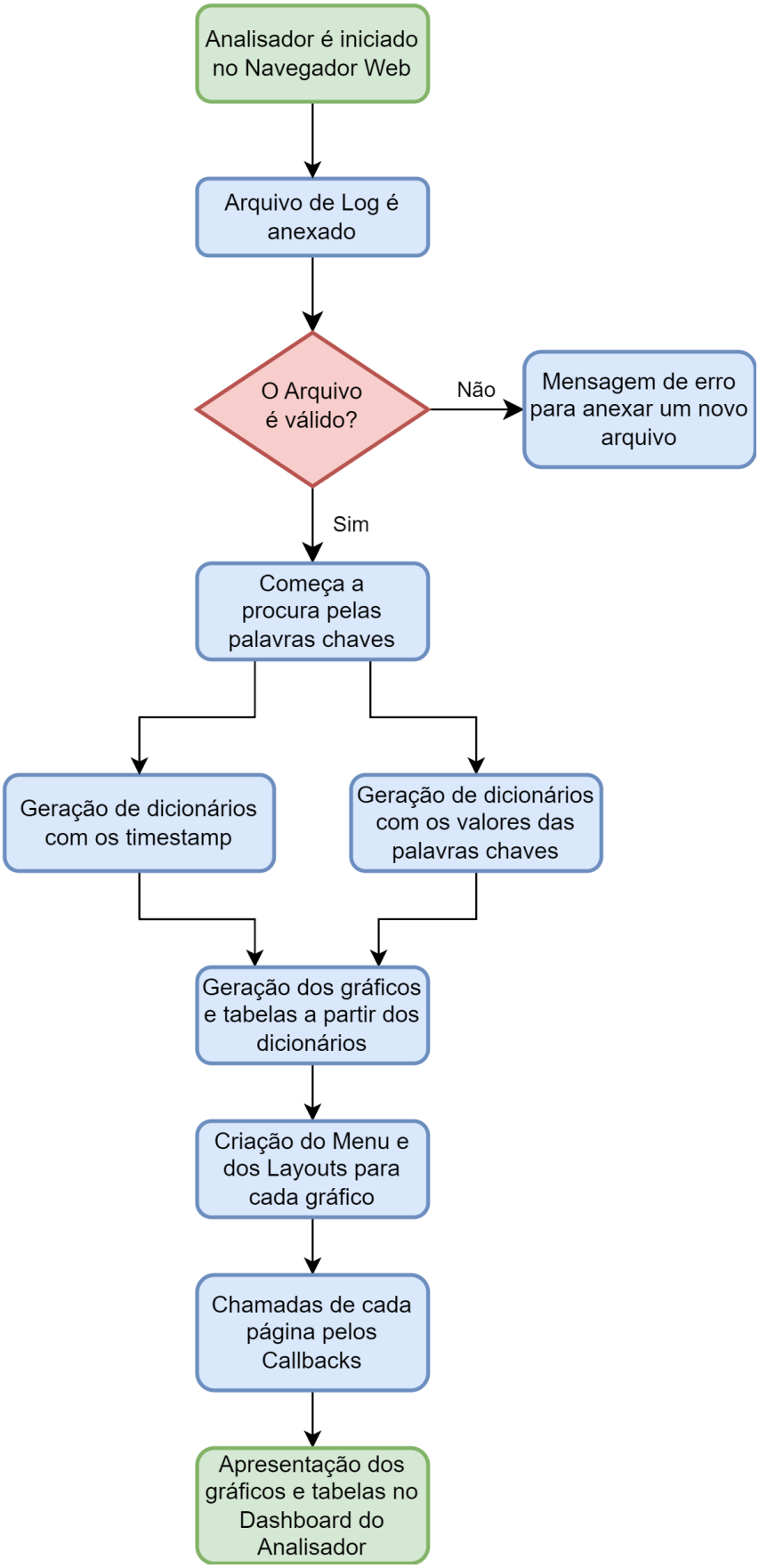
TELECO. ***Tecnologias de Celular: 3G***. 2023. Disponível em: ⟨https://www.teleco.com.br/3g_tecnologia.asp\#:~:text=Nesta\\%20p\\%C3\\%A1gina\\%3A\\%20Apresenta\\%20as\\%20tecnologias,com\\%20altas\\%20taxas\\%20de\\%20transmiss\\%C3\\%A3o.⟩ Acesso em: 18 mar. 2023.

TELECOM, V. ***How To Interpret Ec/Io, SINR, RSSI, RSCP, RSRP, RSRQ parameters in 2G/3G/LTE routers***. 2023. Disponível em: ⟨https://help.venntelecom.com/support/solutions/articles/44001930601-how-to-interpret-ec-io-sinr-rssi-rscp-rsrp-rsrq-parameters-in-2g-3g-lte-routers⟩. Acesso em: 15 mar. 2023.

TELTONIKA. ***RSRP and RSRQ***. 2018. Disponível em: ⟨https://wiki.teltonika-networks.com/view/RSRP_and_RSRQ⟩. Acesso em: 18 mar. 2023.

UFRJ, G. ***IMS IP Multimedia Subsystem***. 2010. Disponível em: ⟨https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2010_2/hugo/Arquitetura.html⟩. Acesso em: 9 set. 2022.

APÊNDICE A - FLUXOGRAMA DO FUNCIONAMENTO DO PROGRAMA



APÊNDICE B - INÍCIO DO PROGRAMA DASHBOARD

```
#!/usr/bin/env python3

from dash import Dash
import dash_bootstrap_components as dbc


external_stylesheets = [dbc.themes.QUARTZ]

app = Dash(
    __name__, suppress_callback_exceptions=True,
    external_stylesheets=external_stylesheets
)


app.title = 'Analisador de Logs'
```

LEITURA DO ARQUIVO DE LOG

```
#!/usr/bin/env python3

import base64
import datetime
import io
import os
import pandas as pd
from dash import dcc, html, dash_table
import dash_bootstrap_components as dbc

#Funcao para abrir o arquivo de Log
def save_file(filename, filecontent):
    f = open(
        os.path.join(os.path.abspath('.'),'data',filename),
        "w")
    f.write(filecontent)
    f.close()

#Funcao para leitura do Arquivo de Log
def parse_contents(contents, filename, date):
    content_type, content_string = contents.split(',')

    decoded = base64.b64decode(content_string)
    try:
        if 'log' in filename:
            # Confere se o arquivo e do tipo log
            df = decoded.decode('iso-8859-1')

        if 'txt' in filename:
            # Confere se o arquivo e do tipo txt
            df = decoded.decode('utf-8')

        save_file(filename="redes_moveis.log",filecontent=df)
    except Exception as e:
        print(e)
        return html.Div([
            'Houve um erro de processamento deste arquivo.'
        ])
```

```
39
40      return html.Div([
41
42          html.H5(filename),
43          html.H6(datetime.datetime.fromtimestamp(date))
44      ])
```

APÊNDICE C - BUSCA DE PALAVRAS E CRIAÇÃO DOS GRÁFICOS E TABELAS

```
#!/usr/bin/env python3

import os
import pandas as pd
from plotly import express as px
import plotly.figure_factory as ff

# Constantes da Documentacao do Android
SIGNAL_TYPE = 'getRilDataRadioTechnology'
SIGNAL_VOICE = 'getRilVoiceRadioTechnology'
SIGNAL_STRENGTH_LTE = 'mLte=CellSignalStrengthLte'
SIGNAL_STRENGTH_WCDMA = 'mWcdma=CellSignalStrengthWcdma'
RSRP = 'rsrp'
RSCP = 'rscp'
RSSI = 'rssi'
RSRQ = 'rsrq'
SIPMSG1 = 'SIPMSG[1]: ['
SIPMSG0 = 'SIPMSG[0]: ['
TECH_X = 'Tempo'
TECH_Y = 'Tecnologia'
SIGNAL_X = 'Tempo'
SIGNAL_Y = 'Pot ncia do Sinal (dBm)'
SIGNAL_Y_RSRQ = 'Qualidade do Sinal (dB)'
RADIO = 'radio'
PROCESS_1 = '10038'
PROCESS_2 = '1000'
MOBILE = 'MobileStatusTracker'
NETWORK_CONTROLLER = 'NetworkController.MobileSignalController'

#Valores para RSRP e RSCP
VERY_POOR = -95
POOR = -85
MODERATE = -75
GOOD = -60
#Valores para RSSI LTE
VERY_POOR_RSSI_LTE = -95
POOR_RSSI_LTE = -85
MODERATE_RSSI_LTE = -75
GOOD_RSSI_LTE = -65
#Valores para RSRQ LTE
VERY_POOR_RSRQ_LTE = -25
POOR_RSRQ_LTE = -20
MODERATE_RSRQ_LTE = -15
GOOD_RSRQ_LTE = -10

# Listas de Busca
key_words_list = []
usual_searchs = []


# Funcoes para encontrar palavras na linha
def find_index(splitted, searching):
    for word in splitted:
        if searching in word:
            return splitted.index(word)

# Funcoes para adicionar palavras encontradas na lista
def result_for_usual_searchs():
```

```
59      search_result = ''
60      for line in usual_searchs:
61          search_result = search_result + line
62      print(search_result)
63
64  #Listas Sinal LTE
65  def make_signal_lte(line):
66      words = (MOBILE, NETWORK_CONTROLLER)
67      for word in words:
68          if word in line:
69              splitted = line.split()
70              idx = find_index(splitted, RSRP)
71              if splitted[idx][5] == '-':
72                  signal_time_lte = splitted[0] + '#' + splitted[1]
73                  signal_lte = splitted[idx][5:]
74                  return signal_lte, signal_time_lte
75      return (None,None)
76
77  #Listas Sinal RSSI LTE
78  def make_signal_rssi_lte(line):
79      words = (MOBILE, NETWORK_CONTROLLER)
80      for word in words:
81          if word in line:
82              splitted = line.split()
83              idx = find_index(splitted, RSSI)
84              if splitted[idx][5] == '-':
85                  signal_rssi_time_lte = splitted[0] + '#' + splitted[1]
86                  signal_rssi_lte = splitted[idx][5:]
87                  return signal_rssi_lte, signal_rssi_time_lte
88      return (None,None)
89
90  #Listas Sinal RSRQ LTE
91  def make_signal_rsrq_lte(line):
92      words = (MOBILE, NETWORK_CONTROLLER)
93      for word in words:
94          if word in line:
95              splitted = line.split()
96              idx = find_index(splitted, RSRQ)
97              if splitted[idx][5] == '-':
98                  signal_rsrq_time_lte = splitted[0] + '#' + splitted[1]
99                  signal_rsrq_lte = splitted[idx][5:]
100                 return signal_rsrq_lte, signal_rsrq_time_lte
101     return (None,None)
102
103 #Listas Sinal WCDMA
104 def make_signal_wcdma(line):
105     words = (MOBILE, NETWORK_CONTROLLER)
106     signal_wcdma = '0'
107     signal_time_wcdma = '0'
108     for word in words:
109         if word in line:
110             splitted = line.split()
111             idx = find_index(splitted, RSCP)
112             if splitted[idx][5] == '-':
113                 signal_time_wcdma = splitted[0] + '#' + splitted[1]
114                 signal_wcdma = splitted[idx][5:]
115                 return signal_wcdma, signal_time_wcdma
116     return (None,None)
117
```

```
118 #Funcoes para leitura e busca de termos chaves
119 def search_in_log_file(user_input):
120     data = {
121         'tech': [],
122         'tech_time': [],
123         'tech_voice': [],
124         'tech_time_voice': [],
125         'signal_lte': [],
126         'signal_time_lte': [],
127         'signal_wcdma': [],
128         'signal_time_wcdma': [],
129         'signal_rssi_lte': [],
130         'signal_rssi_time_lte': [],
131         'signal_rsrq_lte': [],
132         'signal_rsrq_time_lte': [],
133         'sipmsg_left': [],
134         'sipmsg_right': [],
135         'sipmsg_time': [],
136         'sipmsg_arrow': [],
137         'sipmsg_info': []
138     }
139     # Algoritimo para procurar palavras chaves
140     user_input = user_input.replace('\"', "")
141     assert os.path.exists(user_input), "N o foi poss vel encontrar o arquivo:" + str(user_input)
142     f = open(user_input, 'r', encoding="ISO-8859-1")
143     for line in f:
144         for key_word in (key_word for key_word in key_words_list if key_word in line):
145             usual_searchs.append(line)
146         if SIGNAL_TYPE in line:
147             if RADIO in line:
148                 splitted = line.split()
149                 if splitted[2] == RADIO:
150                     tech_time, tech = make_signal_radio_tech_line(line)
151                     data['tech'].append(tech)
152                     data['tech_time'].append(tech_time)
153             if PROCESS_1 in line:
154                 splitted = line.split()
155                 if splitted[2] == PROCESS_1:
156                     tech_time, tech = make_signal_radio_tech_line(line)
157                     data['tech'].append(tech)
158                     data['tech_time'].append(tech_time)
159             if PROCESS_2 in line:
160                 splitted = line.split()
161                 if splitted[2] == PROCESS_2:
162                     tech_time, tech = make_signal_radio_tech_line(line)
163                     data['tech'].append(tech)
164                     data['tech_time'].append(tech_time)
165         if SIGNAL_VOICE in line:
166             if RADIO in line:
167                 splitted_voice = line.split()
168                 if splitted_voice[2] == RADIO:
169                     tech_time_voice, tech_voice = make_signal_voice_tech_line(line)
170                     data['tech_voice'].append(tech_voice)
171                     data['tech_time_voice'].append(tech_time_voice)
172             if PROCESS_1 in line:
173                 splitted_voice = line.split()
174                 if splitted_voice[2] == PROCESS_1:
175                     tech_time_voice, tech_voice = make_signal_voice_tech_line(line)
176                     data['tech_voice'].append(tech_voice)
```

```
177                     data['tech_time_voice'].append(tech_time_voice)
178             if PROCESS_2 in line:
179                 splitted_voice = line.split()
180                 if splitted_voice[2] == PROCESS_2:
181                     tech_time_voice, tech_voice = make_signal_voice_tech_line(line)
182                     data['tech_voice'].append(tech_voice)
183                     data['tech_time_voice'].append(tech_time_voice)
184         if SIGNAL_STRENGTH_LTE in line:
185             signal_lte, signal_time_lte = make_signal_lte(line)
186             signal_rssi_lte, signal_rssi_time_lte = make_signal_rssi_lte(line)
187             signal_rsrq_lte, signal_rsrq_time_lte = make_signal_rsrq_lte(line)
188             if signal_lte:
189                 data['signal_lte'].append(signal_lte)
190                 data['signal_time_lte'].append(signal_time_lte)
191             if signal_rssi_lte:
192                 data['signal_rssi_lte'].append(signal_rssi_lte)
193                 data['signal_rssi_time_lte'].append(signal_rssi_time_lte)
194             if signal_rsrq_lte:
195                 data['signal_rsrq_lte'].append(signal_rsrq_lte)
196                 data['signal_rsrq_time_lte'].append(signal_rsrq_time_lte)
197         if SIGNAL_STRENGTH_WCDMA in line:
198             signal_wcdma, signal_time_wcdma = make_signal_wcdma(line)
199             if signal_wcdma:
200                 data['signal_wcdma'].append(signal_wcdma)
201                 data['signal_time_wcdma'].append(signal_time_wcdma)
202         if SIPMSG0 in line:
203             sipmsg_time, sipmsg_left, sipmsg_arrow, sipmsg_right, sipmsg_info = make_sipmsg(line)
204             data['sipmsg_left'].append(sipmsg_left)
205             data['sipmsg_right'].append(sipmsg_right)
206             data['sipmsg_time'].append(sipmsg_time)
207             data['sipmsg_arrow'].append(sipmsg_arrow)
208             data['sipmsg_info'].append(sipmsg_info)
209         if SIPMSG1 in line:
210             sipmsg_time, sipmsg_left, sipmsg_arrow, sipmsg_right, sipmsg_info = make_sipmsg(line)
211             data['sipmsg_left'].append(sipmsg_left)
212             data['sipmsg_right'].append(sipmsg_right)
213             data['sipmsg_time'].append(sipmsg_time)
214             data['sipmsg_arrow'].append(sipmsg_arrow)
215             data['sipmsg_info'].append(sipmsg_info)
216     f.close()
217     if os.path.exists(user_input):
218         os.remove(user_input)
219     else:
220         print("Arquivo n o existe")
221     return data
222
223 #Lista das Tecnologia de Dados
224 def make_signal_radio_tech_line(line: object) -> object:
225     splitted = line.split()
226     idx = find_index(splitted, SIGNAL_TYPE)
227     tech_time = splitted[0] + '#' + splitted[1]
228     tech = splitted[idx][26:].replace(',', '')
229     return tech_time, tech
230
231 #Lista das Tecnologia de Voz
232 def make_signal_voice_tech_line(line: object) -> object:
233     splitted_voice = line.split()
234     idx = find_index(splitted_voice, SIGNAL_VOICE)
235     tech_time_voice = splitted_voice[0] + '#' + splitted_voice[1]
```

```
236     tech_voice = splitted_voice[idx][27:].replace(',', '')
237     return tech_time_voice, tech_voice
238 
239 #Lista das Mensagens SIP
240 def make_sipmsg(line: object) -> object:
241     splitted = line.split()
242     sipmsg_time = splitted[0] + '#' + splitted[1]
243     if splitted[7] == '[-->]':
244         sipmsg_left = '   ' + splitted[8]
245         sipmsg_arrow = '------->'
246         sipmsg_right = '        | '
247         sipmsg_info = splitted[9] + ' ' + splitted[10]
248     if splitted[7] == '[<--]':
249         sipmsg_right = '   ' + splitted[8]
250         sipmsg_arrow = '<-------'
251         sipmsg_left = '         | '
252         sipmsg_info = splitted[9] + ' ' + splitted[10]
253     return sipmsg_time, sipmsg_left, sipmsg_arrow, sipmsg_right, sipmsg_info
254 
255 #Plotar Tabela das Mensagens SIP
256 def df_table_sipmsg(sipmsg_time, sipmsg_left, sipmsg_arrow, sipmsg_right, sipmsg_info):
257     sipmsg_table = [sipmsg_time, sipmsg_left, sipmsg_arrow, sipmsg_right, sipmsg_info]
258     df_sipmsg = pd.DataFrame(sipmsg_table).transpose()
259     df_sipmsg.columns = ['Tempo', 'UE', 'D i r e   o', 'Rede', 'Info']
260     df_sipmsg['  '] = '  '
261     df_sipmsg['   '] = '   '
262     df_sipmsg['    '] = '    '
263     df_sipmsg['     '] = '     '
264     df_sipmsg['      '] = '      '
265     df_sipmsg['       '] = '       '
266     fig = ff.create_table(df_sipmsg, colorscale=[[0, '#800080'], [.5, '#d8b1d4'], [1, '#ecd8e9']])
267     return fig
268 
269 #Plotar Grafico da Tecnologia de Dados
270 def df_chart_radio_techlogy(tech_time, tech):
271     tech_table = [tech_time, tech]
272     tech_dataframe = pd.DataFrame(tech_table).transpose()
273     tech_dataframe.columns = [TECH_X, TECH_Y]
274     tech_dataframe['Tecnologia de Dados'] = 1
275     tech_chart = px.histogram(tech_dataframe, x=TECH_X, y='Tecnologia de Dados', color=TECH_Y,
276                               title='Tempo x Tecnologia de Dados')
277     tech_chart.update_xaxes(categoryorder='category ascending')
278     tech_chart.update_yaxes(dtick=1, range=[0, 1])
279     return tech_chart
280 
281 #Plotar Grafico da Tecnologia de Voz
282 def df_chart_voice_techlogy(tech_time_voice, tech_voice):
283     tech_table = [tech_time_voice, tech_voice]
284     tech_dataframe = pd.DataFrame(tech_table).transpose()
285     tech_dataframe.columns = [TECH_X, TECH_Y]
286     tech_dataframe['Tecnologia de Voz'] = 1
287     tech_chart = px.histogram(tech_dataframe, x=TECH_X, y='Tecnologia de Voz', color=TECH_Y,
288                               title='Tempo x Tecnologia de Voz')
289     tech_chart.update_xaxes(categoryorder='category ascending')
290     tech_chart.update_yaxes(dtick=1, range=[0, 1])
291     return tech_chart
292 
293 #Plotar Grafico do RSRP
294 def df_chart_signal_lte(signal_lte, signal_time_lte):
```

```
295     signal = list(map(int, signal_lte))
296     if signal != []:
297         signal_max = max(signal)
298         signal_min = min(signal)
299 
300     signal_table = [signal_time_lte, signal]
301     signal_dataframe = pd.DataFrame(signal_table).transpose()
302     signal_dataframe.columns = [SIGNAL_X, SIGNAL_Y]
303     signal_dataframe = signal_dataframe.sort_values(by=SIGNAL_X)
304 
305     if signal != []:
306         signal_chart = px.line(signal_dataframe, x=SIGNAL_X, y=SIGNAL_Y,
307                                title='Tempo x Pot ncia do Sinal 4G (RSRP)')
308         signal_chart.add_hrect(y0=GOOD, y1=GOOD + 60,
309                                line_width=0, fillcolor="green", opacity=0.5)
310         signal_chart.add_hrect(y0=MODERATE, y1=GOOD,
311                                line_width=0, fillcolor="green", opacity=0.4)
312         signal_chart.add_hrect(y0=POOR, y1=MODERATE,
313                                line_width=0, fillcolor="yellow", opacity=0.4)
314         signal_chart.add_hrect(y0=VERY_POOR, y1=POOR,
315                                line_width=0, fillcolor="orange", opacity=0.4)
316         signal_chart.add_hrect(y0=VERY_POOR - 30, y1=VERY_POOR,
317                                line_width=0, fillcolor="red", opacity=0.4)
318         signal_chart.update_yaxes(tick0=MODERATE, dtick=5,
319                                   range=[signal_min - 1, signal_max + 1])
320         return signal_chart
321 
322 #Plotar Grafico do RSSI LTE
323 def df_chart_signal_rssi_lte(signal_rssi_lte, signal_rssi_time_lte):
324     signal = list(map(int, signal_rssi_lte))
325     if signal != []:
326         signal_max = max(signal)
327         signal_min = min(signal)
328 
329     signal_table = [signal_rssi_time_lte, signal]
330     signal_dataframe = pd.DataFrame(signal_table).transpose()
331     signal_dataframe.columns = [SIGNAL_X, SIGNAL_Y]
332     signal_dataframe = signal_dataframe.sort_values(by=SIGNAL_X)
333 
334     if signal != []:
335         signal_chart = px.line(signal_dataframe, x=SIGNAL_X, y=SIGNAL_Y,
336                                title='Tempo x Intensidade 4G (RSSI)')
337         signal_chart.add_hrect(y0=GOOD_RSSI_LTE, y1=GOOD_RSSI_LTE + 60,
338                                line_width=0, fillcolor="green", opacity=0.5)
339         signal_chart.add_hrect(y0=MODERATE_RSSI_LTE, y1=GOOD_RSSI_LTE,
340                                line_width=0, fillcolor="green", opacity=0.4)
341         signal_chart.add_hrect(y0=POOR_RSSI_LTE, y1=MODERATE_RSSI_LTE,
342                                line_width=0, fillcolor="yellow", opacity=0.4)
343         signal_chart.add_hrect(y0=VERY_POOR_RSSI_LTE, y1=POOR_RSSI_LTE,
344                                line_width=0, fillcolor="orange", opacity=0.4)
345         signal_chart.add_hrect(y0=VERY_POOR_RSSI_LTE - 30, y1=VERY_POOR_RSSI_LTE,
346                                line_width=0, fillcolor="red", opacity=0.4)
347         signal_chart.update_yaxes(tick0=MODERATE_RSSI_LTE, dtick=5,
348                                   range=[signal_min - 1, signal_max + 1])
349         return signal_chart
350 
351 #Plotar Grafico do RSRQ LTE
352 def df_chart_signal_rsrq_lte(signal_rsrq_lte, signal_rsrq_time_lte):
353     signal = list(map(int, signal_rsrq_lte))
```

```
354     if signal != []:
355         signal_max = max(signal)
356         signal_min = min(signal)
357 
358     signal_table = [signal_rsrq_time_lte, signal]
359     signal_dataframe = pd.DataFrame(signal_table).transpose()
360     signal_dataframe.columns = [SIGNAL_X, SIGNAL_Y_RSRQ]
361     signal_dataframe = signal_dataframe.sort_values(by=SIGNAL_X)
362 
363     if signal != []:
364         signal_chart = px.line(signal_dataframe, x=SIGNAL_X, y=SIGNAL_Y_RSRQ,
365                                title='Tempo x Qualidade do Sinal 4G (RSRQ)')
366         signal_chart.add_hrect(y0=GOOD_RSRQ_LTE, y1=GOOD_RSRQ_LTE + 60,
367                                line_width=0, fillcolor="green", opacity=0.5)
368         signal_chart.add_hrect(y0=MODERATE_RSRQ_LTE, y1=GOOD_RSRQ_LTE,
369                                line_width=0, fillcolor="green", opacity=0.4)
370         signal_chart.add_hrect(y0=POOR_RSRQ_LTE, y1=MODERATE_RSRQ_LTE,
371                                line_width=0, fillcolor="yellow", opacity=0.4)
372         signal_chart.add_hrect(y0=VERY_POOR_RSRQ_LTE, y1=POOR_RSRQ_LTE,
373                                line_width=0, fillcolor="orange", opacity=0.4)
374         signal_chart.add_hrect(y0=VERY_POOR_RSRQ_LTE - 30, y1=VERY_POOR_RSRQ_LTE,
375                                line_width=0, fillcolor="red", opacity=0.4)
376         signal_chart.update_yaxes(tick0=MODERATE_RSRQ_LTE, dtick=5,
377                                   range=[signal_min - 1, signal_max + 1])
378         return signal_chart
379 
380 #Plotar Grafico do RSCP
381 def df_chart_signal_wcdma(signal_wcdma, signal_time_wcdma):
382     signal = list(map(int, signal_wcdma))
383     if signal != []:
384         signal_max = max(signal)
385         signal_min = min(signal)
386 
387     signal_table = [signal_time_wcdma, signal]
388     signal_dataframe = pd.DataFrame(signal_table).transpose()
389     signal_dataframe.columns = [SIGNAL_X, SIGNAL_Y]
390     signal_dataframe = signal_dataframe.sort_values(by=SIGNAL_X)
391 
392     if signal != []:
393         signal_chart = px.line(signal_dataframe, x=SIGNAL_X, y=SIGNAL_Y,
394                                title='Tempo x Pot ncia do Sinal 3G (RSCP)')
395         signal_chart.add_hrect(y0=GOOD, y1=GOOD + 60,
396                                line_width=0, fillcolor="green", opacity=0.5)
397         signal_chart.add_hrect(y0=MODERATE, y1=GOOD,
398                                line_width=0, fillcolor="green", opacity=0.4)
399         signal_chart.add_hrect(y0=POOR, y1=MODERATE,
400                                line_width=0, fillcolor="yellow", opacity=0.4)
401         signal_chart.add_hrect(y0=VERY_POOR, y1=POOR,
402                                line_width=0, fillcolor="orange", opacity=0.4)
403         signal_chart.add_hrect(y0=VERY_POOR - 30, y1=VERY_POOR,
404                                line_width=0, fillcolor="red", opacity=0.4)
405         signal_chart.update_yaxes(tick0=MODERATE, dtick=5,
406                                   range=[signal_min - 1, signal_max + 1])
407         return signal_chart
408 
409 #Funcao para Geracao dos graficos
410 def generate_graphics():
411 
412 
```

```
413     graphics = {
414         'Tecnologia de Dados': {},
415         'Tecnologia de Voz': {},
416         'Pot ncia do Sinal 4G': {},
417         'Intensidade do Sinal 4G': {},
418         'Qualidade 4G': {},
419         'Pot ncia do Sinal 3G': {},
420         'Mensagem SIP': {}
421     }
422     path = os.path.join(os.path.abspath('.'),'data','redes_moveis.log')
423 
424     if os.path.exists(path):
425         data = search_in_log_file(os.path.join(os.path.abspath('.'),'data','redes_moveis.log'))
426 
427         graphics['Tecnologia de Dados'] = df_chart_radio_techlogy(data['tech_time'],
428                                                                   data['tech'])
429         graphics['Tecnologia de Voz'] = df_chart_voice_techlogy(data['tech_time_voice'],
430                                                                 data['tech_voice'])
431         graphics['Pot ncia do Sinal 4G'] = df_chart_signal_lte(data['signal_lte'],
432                                                               data['signal_time_lte'])
433         graphics['Intensidade do Sinal 4G'] = df_chart_signal_rssi_lte(data['signal_rssi_lte'],
434                                                                        data['signal_rssi_time_lte'])
435         graphics['Qualidade 4G'] = df_chart_signal_rsrq_lte(data['signal_rsrq_lte'],
436                                                             data['signal_rsrq_time_lte'])
437         graphics['Pot ncia do Sinal 3G'] = df_chart_signal_wcdma(data['signal_wcdma'],
438                                                                  data['signal_time_wcdma'])
439         graphics['Mensagem SIP'] = df_table_sipmsg(data['sipmsg_time'], data['sipmsg_left'],
440                                                    data['sipmsg_arrow'], data['sipmsg_right'],
441                                                    data['sipmsg_info'])
442     return graphics
```

APÊNDICE D - CRIAÇÃO DO DASHBOARD

LAYOUTS

```python
#!/usr/bin/env python3

from dash import html, dcc
from layouts.pages import menu, header, footer
#from utils.parser_utils import generate_graphics


layout = html.Div([
    menu.layout,
    header.layout,
    html.Div([
        html.Div([
            html.H2("Pot ncia do Sinal 4G",style={'text-align':'center'})
        ]),
    ], className='form-control container-fluid'),
    html.Br(),
    html.Div([
        dcc.Graph(
            id='signal-lte',
            config={"displayModeBar": True},
        )
    ]),
    footer.layout
],className="container-fluid")
```

CALLBACKS

```python
#!/usr/bin/env python3

from app import app
from dash.dependencies import Output, Input


@app.callback(
    Output('technology-data-graphic', 'figure'),
    Input('store', 'data'))
def update_data_technology(data):
    if not data:
        return {}
    return data['Tecnologia de Dados']

@app.callback(
    Output('technology-data-voice', 'figure'),
    Input('store', 'data'))
def update_voice_technology(data):
    if not data:
        return {}
    return data['Tecnologia de Voz']

@app.callback(
    Output('signal-lte', 'figure'),
    Input('store', 'data'))
def update_signal_lte(data):
    if not data:
        return {}
    return data['Pot ncia do Sinal 4G']
```

```
30
31 @app.callback(
32     Output('signal-rssi-lte', 'figure'),
33     Input('store', 'data'))
34 def update_signal_rssi_lte(data):
35     if not data:
36         return {}
37     return data['Intensidade do Sinal 4G']
38
39 @app.callback(
40     Output('rsrq', 'figure'),
41     Input('store', 'data'))
42 def update_signal_rsrq_lte(data):
43     if not data:
44         return {}
45     return data['Qualidade 4G']
46
47 @app.callback(
48     Output('signal-wcdma', 'figure'),
49     Input('store', 'data'))
50 def update_signal_wcdma(data):
51     if not data:
52         return {}
53     return data['Pot ncia do Sinal 3G']
54
55 @app.callback(
56     Output('sip', 'figure'),
57     Input('store', 'data'))
58 def update_sip(data):
59     if not data:
60         return {}
61     return data['Mensagem SIP']
```

MENUS

```
1  from dash import html, dcc
2
3  layout = html.Div([
4
5      html.Ul([
6          html.Li([
7              dcc.Link('Home',
8                       href='/',
9                       className='nav-link')
10         ], className='nav-item'),
11         html.Li([
12             dcc.Link('Tecnologia de Dados',
13                      href='/technology-data',
14                      className='nav-link')
15         ], className='nav-item'),
16         html.Li([
17             dcc.Link('Tecnologia de Voz',
18                      href='/technology-voice',
19                      className='nav-link')
20         ], className='nav-item'),
21         html.Li([
22             dcc.Link('Pot ncia do Sinal 4G',
23                      href='signal-lte',
24                      className='nav-link')
```

```
25          ], className='nav-item'),
26          html.Li([
27              dcc.Link('Intensidade do Sinal 4G',
28                       href='signal-rssi-lte',
29                       className='nav-link')
30          ], className='nav-item'),
31          html.Li([
32              dcc.Link('Qualidade do Sinal 4G',
33                       href='rsrq',
34                       className='nav-link')
35          ], className='nav-item'),
36          html.Li([
37              dcc.Link('Pot ncia do Sinal 3G',
38                       href='signal-wcdma',
39                       className='nav-link')
40          ], className='nav-item'),
41          html.Li([
42              dcc.Link('Mensagens SIP',
43                       href='sip',
44                       className='nav-link')
45          ], className='nav-item'),
46          html.Li([
47              dcc.Link('Sobre',
48                       href='/about',
49                       className='nav-link')
50          ], className='nav-item'),
51      ],className='nav nav-pills')
52  ], className='container-fluid',)
```