UNIVERSIDADE FEDERAL DO PARANÁ

FERNANDO CREVOCULSKI
GUILHERME VOLPE DA CRUZ

MONITORAMENTO DA REDE LOCAL ATRAVÉS DO USO DE SERVIDOR PROXY EMBARCADO

CURITIBA
2019

FERNANDO CREVOCULSKI
GUILHERME VOLPE DA CRUZ

MONITORAMENTO DA REDE LOCAL ATRAVÉS DO USO DE SERVIDOR PROXY EMBARCADO

Projeto apresentado como requisito à obtenção de nota da disciplina de Trabalho de Conclusão de Curso B, do Curso de Engenharia Elétrica com Ênfase em Sistemas Eletrônicos Embarcados.
Orientador: Prof. Dr. Carlos Marcelo Pedroso

CURITIBA
2019

Em memória de Sinézio Belini Volpe.

Em memória de Nicolau Crevoculski (1948-2019).

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, Alice e Carlos, Por sempre terem me incentivado a concluir esta graduação em questão e me dado a educação para que me tornasse quem sou hoje como pessoa. Agradeço também a minha namorada, Yasmin, por ter me dado todo apoio durante as horas mais difíceis do Curso.

Agradeço aos meus colegas de trabalho na empresa Witt Empreendimentos médicos, onde trabalhei durante todo o período desta graduação, pessoas as quais eu convivo durante a maioria do meu dia, que com toda a certeza, ajudaram a me tornar o profissional que sou hoje.

Agradeço também a todos os meus colegas de classe ao longo destes 6 anos, onde muitos deles acabaram se tornando amigos importantes, seja nas necessárias horas de descontração e nos estudos para avaliações que aconteceram neste período. Um agradecimento especial para meu parceiro Fernando, que teve em mim a confiança de trabalharmos juntos neste trabalho.

Agradeço a todos os professores que participaram desta caminhada, me guiando e auxiliando a conseguir meus conhecimentos acadêmicos e pessoais, também posso dizer que muitos destes se tornaram amigos, que espero manter contato por bons tempos. Obrigado ao professor Dr. Carlos Marcelo Pedroso por ter orientado este TCC.

Finalmente, deixo os meus agradecimentos a todos que me auxiliaram a chegar até aqui, e que contribuíram de alguma forma para que meus objetivos fossem alcançados.

Agradeço a minha família por ter me suportado e me motivado ao longo do curso e em especial ao meu pai Nicolau que, infelizmente, não pôde ver seu filho se formar. Agradeço também ao meu irmão Claudinei pela ajuda durante a longa caminhada do curso e as minhas irmãs.

Expresso minha gratidão a todos os meus colegas de trabalho da Volvo e da Caterpillar por terem contribuído muito com o meu desenvolvimento profissional.

Agradeço a Universidade, por ter me proporcionado um ambiente de amizades e com colegas determinados que facilitaram toda a trajetória do curso, bem como aos professores que sempre focados, nunca desistiram da missão de ensinar em um mundo de constantes mudanças. Agradeço em especial ao nosso Orientador, o Dr. Carlos Marcelo Pedroso, pelos ensinamentos e também pela paciência.

Por fim, agradeço a todos as pessoas que estiveram nos bastidores desta caminhada, seja na Universidade ou na minha trajetória.

“To win big, you sometimes have to take big risks."

Bill Gates

RESUMO

A Internet tem uma ampla importância na sociedade mundial tendo alterado os padrões de comportamento da sociedade moderna. Porém a mesma tecnologia que deixa a informação ao alcance de maneira rápida e fácil, pode também prejudicar a performance das pessoas no ambiente de trabalho, que acabam gastando muito tempo em sites que não agregam aos objetivos corporativos. O objetivo deste projeto é desenvolver um sistema embarcado de *proxy* para monitorar a navegação dos funcionários de pequenas empresas. O projeto consiste utilizar em um servidor *proxy* instalado em um *Raspberry Pi*, onde uma base de dados estruturada é alimentada por *scripts* que fazem a leitura dos dados do *Proxy*, e finalmente um aplicativo para sistemas *android* que lê a base de dados e apresenta estatísticas de uso e relatórios para o gestor da empresa. Os resultados indicam todas as funcionalidades do sistema de *proxy* embarcado, aonde é possível o monitoramento de todo o tráfego de dados da rede pelo gestor na palma de sua mão, de maneira simples e descomplicada

Palavras-chave: Android. Proxy. Squid. Monitoramento. Internet.

ABSTRACT

The Internet is of wide importance in world society, through which can quickly and easily reach every information needed, but it can reducing employuu productivity, who end up spending a lot of time on sites that do not add to corporate goals. The objective of this project is to develop an embedded proxy system to monitor the navigation of employees of a small business. Applying a proxy server to a Raspberry Pi, where a structured database is fed by scripts that read Proxy data, and finally an android application that reads the database and presents statistics to the company manager. The results indicate all the features of the embedded proxy system, where it is possible to monitor network data traffic using a smartphone or tablet, greatly facilitate browsing and interpreting the data for managers.

Key-words: *Android. Proxy. Squid. Database. Monitoring. Internet.*

LISTA DE ILUSTRAÇÕES

LIST OF QUADROS

LISTA DE ABREVIATURAS E SIGLAS

IP	*Internet Protocol*: Protocolo de internet.
GB	Gigabyte: É uma unidade de medida de informação.
DNS	*Domain Name System*: Sistema de Nomes de Domínio.
DHCP	*Dynamic Host Configuration Protocol*: protocolo de configuração dinâmica de *host*.
RAM	*Random Access Memory*: Memória de acesso aleatório.
USB	*Universal Serial Bus*: Porta universal.
HDMI	*High-Definition Multimedia Interface*: Interface de multi-mídia de alta definição.
CSI	*Camera Serial Interface*: Interface serial para câmera.
DSI	*Display Serial Interface*: Interface serial de exibição.
SD	*Secure Digital Card*: Cartão digital seguro.
DB	*Database*: Banco de dados.
WiFi	*Wirelles Fidelity*: Rede sem fio.
SO	Sistema Operacional.
PWM	*Pulse Width Modulation*: Modulação por largura de pulso.
UART	*Universal asynchronous receiver-transmitter*: Receptor/transmissor assíncrono universal.
SATA	*Serial AT Attachment*: Conector que liga o dispositivo de armazenamento em massa aos barramentos do sistema.
MB	Megabyte: É uma unidade de medida de informação
ID	Identidade: É um identificador.

URL	*Uniform Resource Locator*: Localizador Uniforme de Recursos, endereço típico digitado no navegador, exemplo www.google.com.
LGPD	Lei geral de proteção de dados pessoais.
EXT3	*Third extended Filesytem*: Sistema de arquivos Linux.
EXT4	*Fourth extended Filesytem*: Sistema de arquivos Linux.
XFS	*Extended Filesytem*: Sistema de arquivos Linux.
JFS	*Journaled File System*: Sistema de arquivos Linux.

SUMÁRIO

1 INTRODUÇÃO

Não é de hoje que é notado o crescimento do uso da Internet no mundo. Desde sua criação, o site americano *We Are Social* apresenta relatórios mensais a respeito das estatísticas do uso da Internet. Os dados mostram crescimento em todos os sentidos, por exemplo, em 2019 o número de usuários cresceu 9,1% em relação à 2018. A Figura 1.1 mostra dados totais sobre a população mundial relacionada ao uso da rede mundial de computadores (CHAFFEY, 2019).

FIGURA 1.1: Uso de internet e redes sociais no mundo

Fonte: (CHAFFEY, 2019).

A população do nosso planeta já passa de 7 bilhões de pessoas em 2019, sendo que destes, 67% já possuem um celular pessoal. Outro dado interessante é o número de usuários de Internet, que alcança 57% da população. O numero de usuários ativos de redes sociais vem crescendo 9% ano a ano, estando com aproximadamente 3.484 bilhões de usuários mundialmente (CHAFFEY, 2019).

Trazendo o contexto um pouco mais para as redes sociais, outra empresa americana chamada *Global Web Index* analisou dados dos 45 maiores mercados de Internet do mundo, levando em conta o tempo médio que uma pessoa gasta nestas redes por dia, apresentando estes dados separados por países, mostrado na figura 1.2, e por regiões, mostrado na figura 1.3 (GLOBALWEBINDEX, 2019).

A figura 1.2 mostra que no Brasil, a média de tempo gasto por dia em redes sociais é de 225 minutos, ficando atrás apenas das Filipinas, que lidera o ranking com

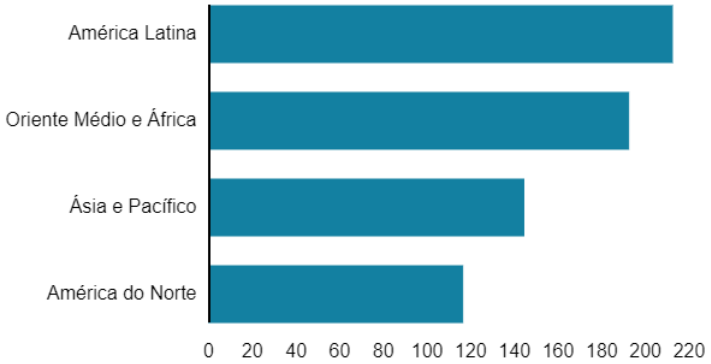
FIGURA 1.2: Uso de rede sociais por minutos

Ranking	País	Minutos (2019)	Minutos (2018)
1	Filipinas	241	248
2	Brasil	225	219
3	Colômbia	216	214
4	Nigéria	216	206
5	Argentina	207	197
6	Indonésia	195	203
7	Emirados Árabes	191	180
8	México	190	194
9	África do Sul	190	178
10	Egito	186	185
12	Arábia Saudita	186	172
13	Turquia	185	172
18	Rússia	148	141
19	Índia	145	148
22	China	139	120
25	EUA	117	125

Fonte: (GLOBALWEBINDEX, 2019).

241 minutos médios diários. Os Estados Unidos por exemplo, aparecem somente na 25ª posição, com apenas 117 minutos.

FIGURA 1.3: Gráfico de minutos gastos em redes sociais por região



Fonte: (GLOBALWEBINDEX, 2019).

Já a figura 1.3 mostra que mesmo ao dividir os dados entre as regiões, o Brasil ainda assim tem uma posição de destaque, por estar contido na região que mais utiliza destas redes

Mas o que pode acontecer se este tempo foi gasto dentro do ambiente de trabalho? Isso é o que a *Triadd Consulting*. Uma empresa especializada em produtividade, apresentou dados de como os trabalhadores utilizam a Internet no horário de trabalho.

De acordo com (RAKOSKI, 2018) 80% das pessoas gastam até 3 horas da

jornada de trabalho com atividades que não contribuem com seu serviço; para 35,6% dos entrevistados utilizam os próprios computadores para acessar redes sociais; 27,3% do tempo é gasto com e-mails e 21,4% com outras atividades online; 40,9% dos entrevistados usa o e-mail para repassar piadas, 26,1% deles trocam links do YouTube, 20,6% jogam em rede e 11,% veem pornografia.

É importante ressaltar que o uso excessivo de Internet e redes sociais pode até resultar em demissão por justa causa, de acordo com o Artigo 482 da CLT. os principais vilões da produtividade no trabalho são: (RAKOSKI, 2018)

- Redes Socias (Instagram, Facebook, YouTube);
- Email Pessoal;
- Serviços de Comunicação;
- Sites de Entretenimento .

1.1 PROBLEMA

Com estes altos índices de usuários ativos de mídias sociais, inclusive no horário de trabalho, o gestores das empresas devem se preocupar em reduzir o prejuízo com a quebra de produtividade causada pela utilização de redes sociais e outros aplicativos de Internet.

1.2 JUSTIFICATIVA

Uma nova lei chamada Lei Geral de Proteção de Dados Pessoais (LGPD) foi assinada no ano de 2018 pelo então presidente Michel Temer. Esta lei Regulamenta o tratamento de dados pessoais de clientes e usuários por empresas públicas e privadas. A Lei entrará em vigor em agosto de 2020 e as multas poderão chegar em até R$ 50 milhões. Entre as adequações desta lei, as empresas deverão conter 3 figuras centrais: (PLANALTO, 2018)

- Controlador;
- Operador;
- Encarregado. .

Um sistema *proxy* embarcado de monitoramento da rede e com análise de dados oferece alguns benefícios:

- transparência e conhecimento sobre os dados transmitidos e recebidos da internet na empresa;
- sistema de baixo custo, ideal para pequenas e médias empresas;
- monitoramento e fiscalização das atividades dos usuários da rede.

O desenvolvimento de um aplicativo mobile para o administrador permite:

- Trazer informações de rede para uma plataforma amigável e com alta disponibilidade;
- alertar o administrador sobre acessos suspeitos, inseguros e identificar os usuários infratores;
- dispor ao administrador dados que permitam tomada de decisões.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

O objetivo geral do presente trabalho é a concepção de um *proxy* embarcado de baixo custo, que registre e trate os dados da uma rede local e facilite o monitoramento e fiscalização das atividades dos usuários pelo administrador.

1.3.2 Objetivos específicos

Tem-se os seguintes objetivos específicos:

- Realizar a configuração do servidor proxy (Squid) na plataforma embarcada Raspberry Pi;
- Realizar configurações de DHCP, DNS, Gateway e de proxy automático nos hosts da rede;
- Leitura e inserção de arquivos de *log* para o banco de dados embarcado ao Raspberry Pi, com informações de horário e data de registro, URL acessada, mimetype, nome do usuário, IP, quantidade de bytes, métodos de request (GET, PUT...) e protocolo utilizado;
- Estruturação de banco de dados para aumento da velocidade das consultas feitas através de dispositivos móveis.
- Gerar relatórios de acessos por IP na rede, relatórios de redes sociais (Facebook, Twitter, Instagram e Youtube), relatórios de tráfego da rede ao longo das horas

do dia, relatório de tipos de dados trafegados por usuário (mimetype) e busca de palavra-chave nas URLs da base.

- Desenvolver aplicativo Android para a exibição de relatórios da rede, conectado ao banco de dados;

2 REVISÃO BIBLIOGRÁFICA

Para que a fundamentação possa englobar o projeto proposto de forma eficiente, foram levantados todos componentes do sistema para que posteriormente, o devido levantamento dos tópicos referentes a fundamentação teórica seja feito. Tem-se portanto:

- dispositivo servidor ou *hardware* do projeto;
- sistema operacional do servidor;
- programas e protocolos intermediários como Squid, DHCP, entre outros.

Agora, sabendo que o sistema de forma simplificada, é separado em *hardware*, sistema operacional e aplicações, é apropriado separar a fundamentação nesses mesmos tópicos para facilitar a estrutura e entendimento da fundamentação.

2.1 *HARDWARE* DO PROJETO

Primeiramente será discutido qual será as especificações e posteriormente as opções disponíveis.

2.1.1 Especificações

Para a escolha de um dispositivo para compor o servidor do projeto, deve-se levar em conta fatores como o tamanho e as especificações técnicas.

O objetivo é desenvolver para pequenas e médias empresas com no máximo 20 dispositivos simultâneos conectados, o servidor escolhido deve possuir memória RAM equivalente (entre 1 e 2 GB), memória disponível para *cache* (neste caso, quanto mais melhor), possuir uma aceitável capacidade de processamento (frequência de *clock* na casa dos gigahertz) para evitar lentidão e permitir acesso à rede desses, aproximadamente, 20 dispositivos.

Como muitos leigos entendem o servidor como paradigma de computadores fisicamente espaçosos e que ocupam cômodos inteiros, é importante clarificar que neste projeto a aplicação trata-se de uma pequena/média empresa, logo deve-se priorizar um dispositivo de tamanho pequeno.

2.1.2 Raspberry Pi

O Raspberry Pi é um mini computador desenvolvido pelo Fundação Raspberry Pi do Reino Unido, com o objetivo de auxiliar alunos na aprendizagem sobre sistemas operacionais e programação. Esse dispositivo possuí o mesmo tamanho de um cartão de crédito, ao mesmo tempo tem grande capacidade computacional, como será visto. Na figura 2.1 é possível ver o mini computador e ter noção de seu tamanho (FOUNDATION, 2018).

FIGURA 2.1: Mini Computador - Raspberry Pi 3 –

Fonte:(FOUNDATION, 2018)

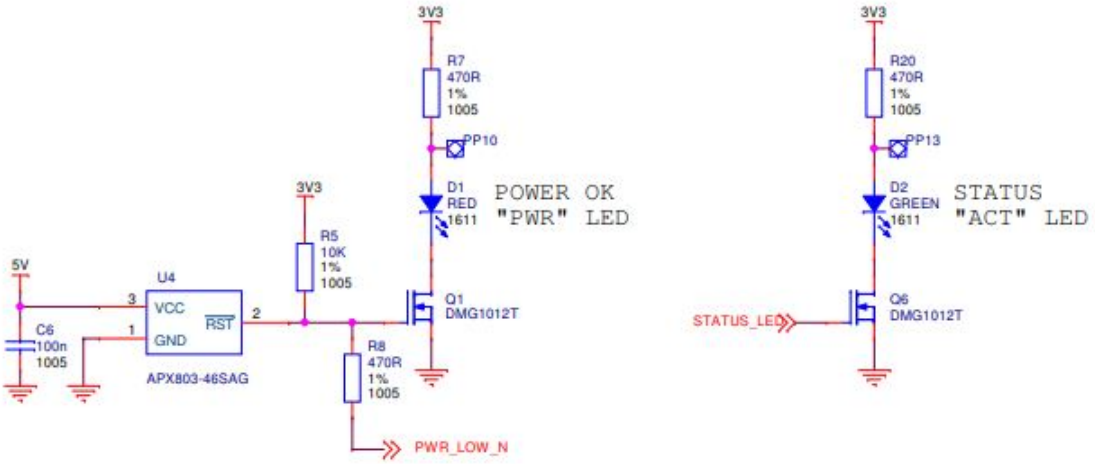
Este dispositivo possui um processador ARM Cortex A53 com arquitetura BCM2837 da fabricante Broadcom com quatro núcleos de 1,2 GHz de frequência. Possui também 1 GB de memória RAM, placa de rede BCM43438 com *bluetooth* de baixo consumo de energia, 40 pinos de entrada e saída e 4 portas USB. Além disso, ele possui uma porta HDMI, porta CSI para conexão de câmera, porta DSI para *touchscreen*, suporte para cartão SD e uma porta para conexão via cabo de Ethernet (FOUNDATION, 2018).

Quanto a alimentação, o Raspberry Pi é alimentado em 5,1V e 2,5A de corrente (FOUNDATION, 2018). Na figura 2.2 está o diagrama de alimentação do Raspberry.

2.1.3 Orange Pi

Outra opção de *hardware* para o projeto é o Orange Pi.

FIGURA 2.2: Esquemático da Alimentação - Raspberry Pi 3 –



Fonte:(FOUNDATION, 2018)

O Orange Pi é um mini computador open-source capaz de rodar, assim como o Rasberry, diversos sistemas operacionais, entre eles, Android 4.4, Ubuntu e o Debian. O objetivo da criação desse circuito, é desenvolver um dispositivo compacto e de alto desempenho para uso geral (ORANGEPI, 2018).

Quanto as especificações do mesmo, tem-se:

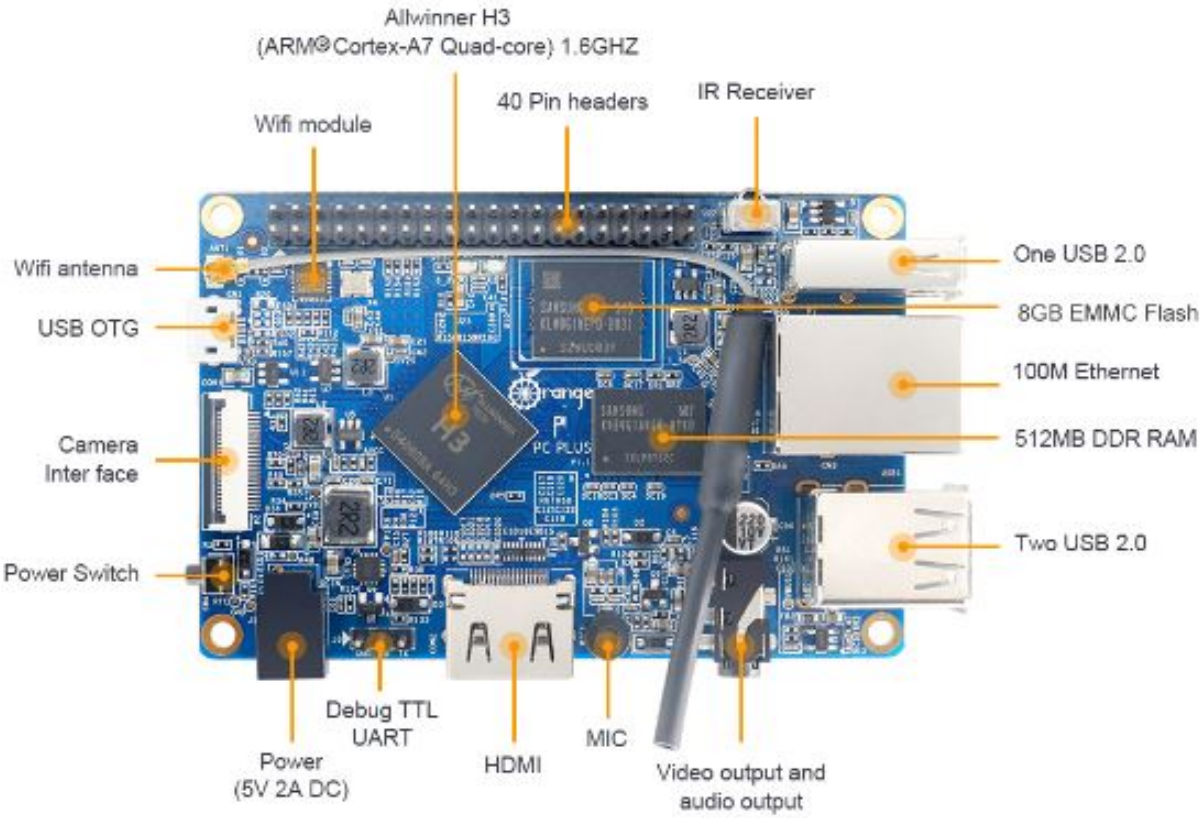
- processador H3 Quad-core Cortex-A7 de 1,6 GHz;
- memória RAM de 1 GB;
- suporte a cartão SD de até 32 GB;
- entrada para cabo de rede;
- conector CSI de alta resolução;
- suporte à microfone;
- suporte à HDMI;
- módulo WiFi;
- total de 3 entradas USB;
- 40 pinos de entrada/saída compatíveis ao Raspberry (ORANGEPI, 2018).

Na figura 2.3 tem-se a vista superior do Orange Pi, assim como seus componentes.

Sua alimentação é um pouco superior ao Raspberry, sendo alimentado com 5V de tensão, mas 3A de corrente (ORANGEPI, 2018).

Ao mesmo tempo em que o Orange Pi é um pouco mais rápido em comparação

FIGURA 2.3: Vista superior do Orange Pi



Fonte:(ORANGEPI, 2018)

com Raspberry Pi, ele também é mais caro e por isso possui quantidade menor de usuários e por consequência menos material disponível.

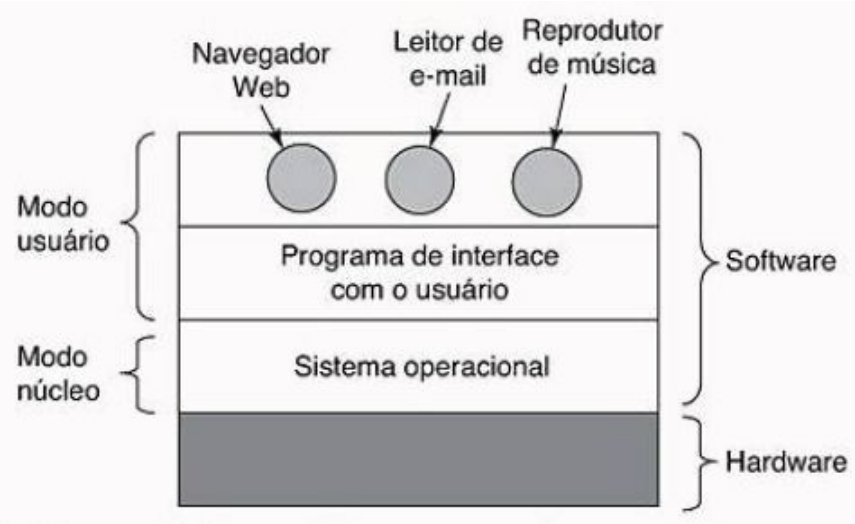
2.2 SISTEMA OPERACIONAL

Com o aumento da complexidade dos sistemas de computadores que possuem vários componentes como processadores, memória, discos, teclado e outros, seria um problema para um programador, por exemplo, ter que conhecer todos esses dispositivos apenas para desenvolver um aplicativo para uma máquina. Para evitar o problema do exemplo anterior, foi necessário implementar um programa (com uma média média de 5 milhões de linhas) capaz de organizar todas essas informações do computador e periféricos e disponibiliza-las de forma clara permitindo a abstração de hardware para as aplicações (TANENBAUM, 2010).

Essa diminuição na abstração ocorre a partir da capacidade de gerenciamento do sistema e por meio da padronização de comunicação entre dispositivos externos. Portanto, não importa qual aparelhagem é conectada ao computador, interessa apenas

o formato de *input* padronizado que é reconhecido pelo SO (TANENBAUM, 2010). Na figura 2.4 está uma ilustração da posição que o SO ocupa dentro do sistema computacional.

FIGURA 2.4: Diagrama básico de um sistema computacional



Fonte:(TANENBAUM, 2010)

A definição de sistema operacional é um conceito bem complicado, dizer simplesmente que ele é um *software* que executa de forma centralizada funções que coordenam o sistema não pode ser considerada completamente verdadeira. No geral, o SO executa duas funções principais, fornecer aos programadores de aplicativos os recursos claros e conhecidos pelo pessoal desse nível de aplicação, aumentando assim a abstração e gerenciar os recursos de *hardware* (TANENBAUM, 2010). Veja que o conceito de abstração está intrínseco dentro da própria definição de sistema operacional.

Embora existam várias arquiteturas diferentes para sistemas operacionais, a maior parte dos sistemas atuais, como Windows, Linux e Debian foram feitos segundo a arquitetura monolítica. Como esse projeto terá um SO baseado no Debian, durante a abordagem da estrutura do SO será dado um foco para entender mais sobre a arquitetura monolítica em específico (TANENBAUM, 2010).

2.2.0.1 Sistemas Monolíticos

Para essa arquitetura, o SO é executado inteiramente em um único bloco centralizado, possuindo assim uma série de rotinas conectadas a rotina principal (no centro) e deixando cada uma delas com dados de saída (*output*) bem definidos, seguindo as

boas práticas de programação e padrões documentados. Mesmo possuindo várias classes de rotinas como por exemplo, rotinas de serviço e rotinas de utilidades, essa arquitetura implementa uma estrutura mínima para executar a chamada ao sistema, para que seja possível manter a chama ao sistema simples, tem-se lugares definidos para a entrada de parâmetros para a chamada ao SO (chamada essa que ocorre durante interrupções, por exemplo) executando assim a instrução de desvio de controle que tira o controle do usuário e passa ao SO (TANENBAUM, 2010).

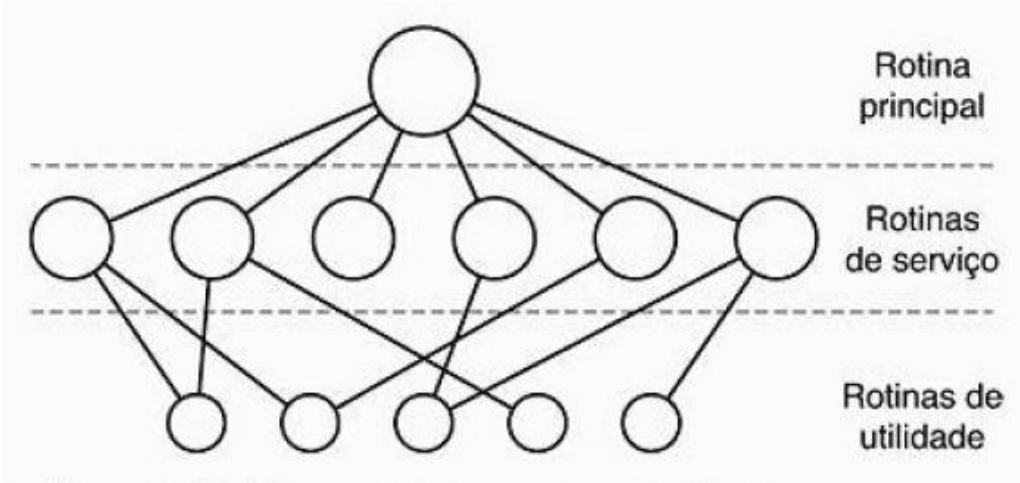
Quanto as vantagens dessa arquitetura tem-se o desempenho, já que ela possui elevada integração dos componentes internos, por consequência, os componentes podem interagir entre eles diretamente, sem a necessidade de utilizar a rotina principal, além disso, permite que sejam aproveitadas as informações à nível de *hardware*, isto é, os dados transmitidos por dispositivo do sistema, como teclado, monitor, entre outros, são passados diretamente para os componentes de destino, evitando assim intermediários que filtram e restringem as informações dos mesmos.(TANENBAUM, 2010; TEIXEIRA, 2018).

Na figura 2.5 está a representação de um sistema monolítico e suas funções integradas. Como pode ser visto, as rotinas estão todas interligadas entre elas e podem conversar entre si, sem a necessidade da rotina principal ou sistema operacional intervir no processo. Isso permite que o mesmo se ocupe de apenas com a sua própria rotina. No entanto, essa mesma integração pode causar uma parada no sistema já que se uma parar de responder, a cadeia toda entra em erro (TANENBAUM, 2010; TEIXEIRA, 2018).

2.2.1 Compartilhamento de Tempo: *Time Sharing*

O Processador de um sistema computacional pode ser considerado o 'cérebro' da máquina, sua função principal é a busca e execução de instrução onde, ele busca na memória a instrução e a executa. Embora existam diferentes arquiteturas de processadores para diversas aplicações que tentam executar tarefas em paralelo, no caso dos computadores isso normalmente ocorre em série. No entanto, uma série de medidas são tomadas para causar a impressão de paralelismo, entre elas está o uso de vários núcleos de processamento que fazem o ciclo de busca e execução para diferentes instruções da memória e o compartilhamento do recurso (*time sharing*) (TANENBAUM,

FIGURA 2.5: Diagrama da arquitetura monolítica



Fonte:(TANENBAUM, 2010)

2010).

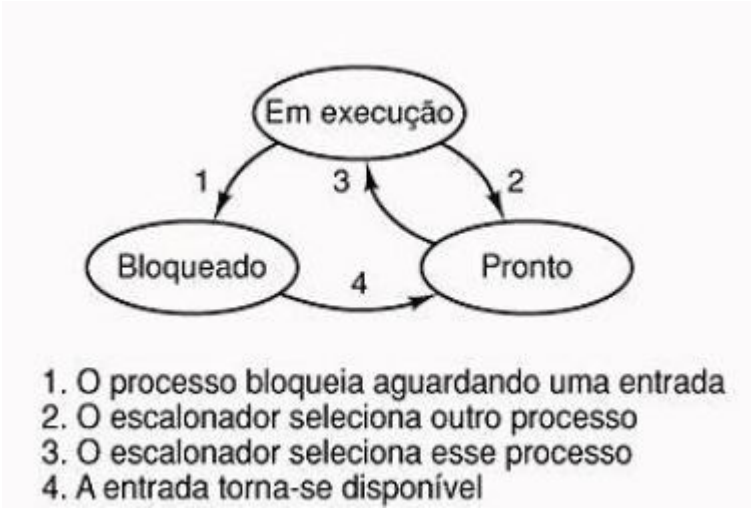
O *time sharing* consiste em dividir o recurso (no caso o processador) entre os devidos processos que estão competindo para serem executados. Os processos possuem níveis de prioridade, para simular o paralelismo, o processador executa o próximo processo da fila durante um curto período de tempo e depois passa para o próximo da fila. Toda vez que é feita a mudança entre um processo e outro é necessário fazer o salvamento de contexto, onde são armazenados os registradores que carregam as últimas informações sobre o processo para que assim seja possível saber qual deve ser a próxima instrução desse processo que deve ser executada num futuro escalonamento por parte do processador (MIDORIKAWA, 2018; TANENBAUM, 2010).

Como os processos não estão rodando ao mesmo tempo, eles possuem diferentes estados dentro de um computador. No geral, o processo pode estar em execução, bloqueado ou pronto, somente os processos que estão prontos podem ser escalonados e executados.Na figura 2.6 está a ilustração dos estados de processos (TANENBAUM, 2010).

2.2.2 Debian

O Debian é uma distribuição do sistema operacional Linux, que quando instalado, traz um conjunto de programas básicos e utilitários além de drivers de comunicação entre grande número de periféricos. Esse sistema foi desenvolvido através do Projeto Debian, formado por um conjunto de indivíduos cujo objetivo era criar um

FIGURA 2.6: Estados de processos



Fonte:(TANENBAUM, 2010)

sistema completamente livre (DEBIAN, 2018).

O Debian começou em 1993 com Ian Murdock em uma nova distribuição que seria feita abertamente, nos princípios do Linux e do projeto GNU. Ele cresceu de forma gradual para se tornar uma comunidade grande e bem organizada de desenvolvedores e usuários (DEBIAN, 2018).

Esse sistema utiliza o *kernel* Linux que faz as operações fundamentais, como por exemplo suporte a vários protocolos de comunicação. A reputação de estabilidade e segurança do Debian fez com que outras distribuições Linux se derivassem dele, entre eles o Ubuntu (NEGUS; BRESNAHAN, 2014).

2.2.2.1 Sistema de Arquivos

O Linux suporta uma lista de sistemas de arquivos, no entanto, o objetivo de todos eles é permitir o controle e acesso ao disco rígido, os mais utilizados são o EXT3 e EXT4, porém o Linux também suporta XFS e JFS (TANENBAUM, 2010).

2.2.3 Raspbian

O Raspbian é uma distribuição Linux derivada do Debian mas, ao mesmo tempo otimizado para o Raspberry Pi. Também é livre e possui mais de 35 mil pacotes otimizados para o *hardware* do Raspberry.

2.2.3.1 Configurações para *Hardware*

Com o Raspbian é possível editar uma série de parâmetros de baixo nível para configuração, todas voltadas para o sistema do Raspberry Pi. Entre elas tem-se:

- formato de arquivos;
- opções de memória como habilitar ou desabilitar acesso à memória *cache*, distribuição de uso de memória entre *cache* e RAM e habilitar ou desabilitar a taxa de atualização para medição de temperatura da RAM;
- opções para o módulo da câmera: habilitar ou desabilitar a câmera e memória miníma para uso da câmera;
- opções de rede: configuração do endereço IP da máquina;
- opções de áudio: alterações no algoritmo de PWM;
- opções de vídeo: opções de cores no HDMI, codificação de pixel, modos de resolução e frequência do HDMI, configurações de segurança, de rotação de imagem, entre outros;
- opções de partida do Raspberry (*Boot):* alterações nas configurações de carregamento do *kernel*, alteração da frequência de clock da UART e taxa de transmissão;
- opções para (*Overclock):* modo turbo, alterações na frequência de processamento, uso de memória *cache* para o processador e alterações na amplitude da tensão degrau do *clock*;
- opções de uso do cartão SD;
- opções para a alimentação USB (RASPBIAN, 2018).

2.3 PROGRAMAS E PROTOCOLOS DO PROJETO

Nesta seção serão abordados os programas e protocolos que serão utilizados do *hardware*/servidor do projeto.

2.3.1 Squid - *Optimising Web Delivery*

O Squid é um *proxy* capaz de efetuar salvamento em *cache* através da rede suportando os protocolos HTTP (*Hypertext Transfer Protocol*, protocolo de transferência de hipertexto) e o HTTPS (*Hyper Text Transfer Protocol Secure*, protocolo seguro de transferência de hipertexto). Ambos os protocolos são usados na base da comunicação

de dados da rede global de internet, funcionando como um protocolo de requisição/resposta no modelo cliente/servidor. Portanto, quando um cliente qualquer submete uma mensagem de requisição HTTP/HTTPS, o servidor que é responsável por fornecer os recursos, responde para o cliente com a informações requisitadas (SQUIDORG, 2018; COMER, 2007).

O Squid foi uma das linhas de pesquisa derivadas do Harvest Cache Daemon desenvolvido na década de 90. O Projeto Squid foi uma iniciativa de pesquisa na área de tecnologias para *caching* no entanto, o projeto foi encerrado e passou a ser retomado anos depois com o auxílio de voluntários e doadores. Atualmente, o Squid continua sendo desenvolvido por voluntários e sua história junto da segurança e funcionamento fez com que essa tecnologia tivesse a adoção por parte de muitas empresas. Essas mesma companhias encontraram no Squid a solução para economizar com tráfego de Internet, melhorias de performance, entrega rápida de serviços aos clientes no mundo a fora (SQUIDORG, 2018).

Ao mesmo tempo, o squid oferece grande controle e um ambiente seguro para o desenvolvimento de um *(*proxy web) que serve à outras aplicações. Entre as funções, tem-se opções otimização de tráfego de dados, opções de segurança e controle (SQUIDORG, 2018).

2.3.1.1 Configurações do Squid

A configuração do squid é realizada em três arquivos, que podem ser vistos no 2.1 (SAINI, 2011):

Quadro 2.1: Principais arquivos de configuração do Squid

```
1 /etc/squid/squid.conf
2 /usr/local/squid/etc/squid.conf
3 \${prefix}/etc/squid.conf
```

Este arquivo permite as principais configurações para o Squid, como a configuração da porta HTTP que será escutada, alterações nos critérios e parâmetros para *cache* como tamanho mínimo ou máximo do arquivo, taxa de *cache* e outros. Além disso, o direcionamento do fluxo da rede através do servidor *proxy* e muitas outras opções disponíveis para o gerente da rede (SAINI, 2011).

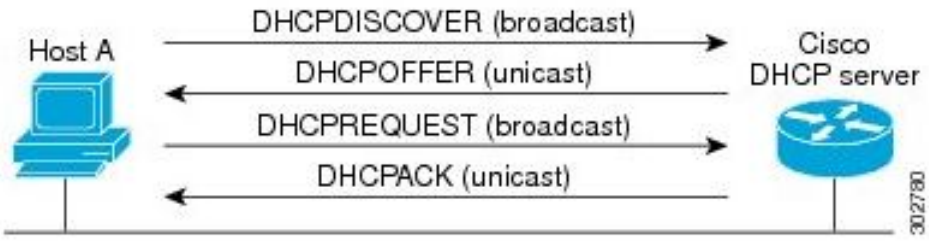
2.3.2 Protocolo DHCP para o Servidor Proxy - *Dynamic Host Configuration Protocol*

O DHCP é um protocolo que oferece o serviço de configuração dinâmica, como o fornecimento de IP de *host*, máscara de sub-redes e *gateway* padrão entre terminais (COMER, 2007).

Supondo um cliente requisitando acesso à uma rede qualquer através do DHCP, o funcionamento desse protocolo começa quando o cliente envia em *broadcast* (destinado à todas as máquinas) um pacote com requisição DHCP na porta 67. À partir disso, qualquer servidor DHCP irá responder a requisição com a informação para que o cliente possa se conectar, usualmente um endereço IP e uma máscara de rede, além de dados opcionais estão contidos nesse pacote de resposta. Com isso, o cliente passa a fazer parte da rede através do uso das informações recebidas pelo servidor DHCP (COMER, 2007; CISCO, 2018).

A figura 2.7 ilustra as etapas para o estabelecimento de conexão através do servidor DHCP.

FIGURA 2.7: Etapas do Processo de Conexão DHCP



Fonte:(CISCO, 2018)

O DHCP é amplamente utilizado para o estabelecimento de conexão sem fio, uma vez que os clientes não são conhecidos e eles necessitam ganhar acesso a rede, dessa forma esse protocolo permite que eles façam parte da rede.

2.3.3 Servidor DNS - *Domain Name System*

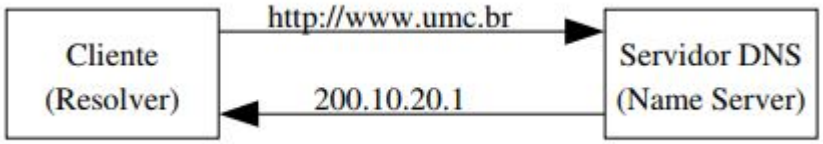
De forma geral, quando é feito o acesso à uma página da internet, tipicamente é inserido o formato (www.nomedapagina.com.br), mas de fato o endereço da página é completado pelo navegador resultando em (https://www.nomedapagina.com.br ou http://www.nomedapagina.com.br). O trecho (http:// e https://) serve para sinalizar qual protocolo de comunicação base é utilizado, conforme já discutido anteriormente, logo,

o resto do endereço é o que serve efetivamente para designar qual servidor deve ser acessado (COMER, 2007).

O próprio endereço do site na Internet, pode ser entendido como uma abstração para o usuário, na realidade este endereço é um IP, por exemplo, ao digitar o endereço do Google (www.google.com.br) na realidade está sendo acessado o IP (172.217.29.99), importante notar que o Google possui diversos servidores e por isso os IPs podem mudar. Convencionalmente, a tradução do endereço do *site* para IP é adotada e justificada pelo princípio pelo qual o cérebro humano tem maior facilidade de lembrar de nomes e textos em comparação com sequências numéricas, isto é, é muito mais fácil lembrar do nome Google do que do 172.217.29.99 (COMER, 2007; TANENBAUM, 2010).

Na figura 2.8 está o diagrama simplificado do funcionamento do DNS, que recebe um nome e traduz para endereço de IP.

FIGURA 2.8: Entrada e Saída do DNS



Fonte:(JUNIOR, 2018)

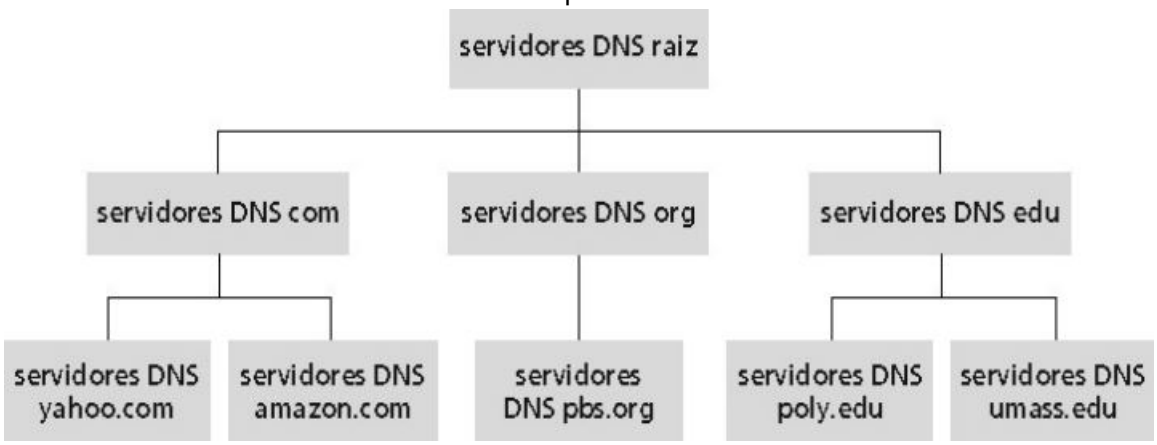
O funcionamento do servidor DNS se dá a partir do auxilio de outros servidores que armazenam os endereços IP e retornam o mapeamento quando o nome não é conhecido, esses servidores são os primeiros a serem consultados e por isso chamam-se de servidores raíz. Existem várias cópias do servidor raiz espalhadas pelo mundo, permitindo assim acessar o mais próximo e melhorando o desempenho. Na figura 2.9 estão os principais servidores, essa distribuição global de servidores ajuda a internet a se tornar descentralizada, ou seja, não há como desativar a rede global através de leis ou ações governamentais localizadas, pois através dessa variedade de servidores a Internet acaba existindo por si só (UEYAMA, 2014; JUNIOR, 2018).

Os servidores raiz conseguem informar para o DNS solicitante, qual é próximo endereço IP de servidos DNS que possui o nome que o requisitante está procurando, como o raiz encontra-se no topo do diagrama, convencionou-se o nome de *Root Server*, como pode ser visto na figura 2.10.

FIGURA 2.9: Principais Servidores Raiz

Fonte:(UEYAMA, 2014)

FIGURA 2.10: Hierarquia dos Servidores DNS



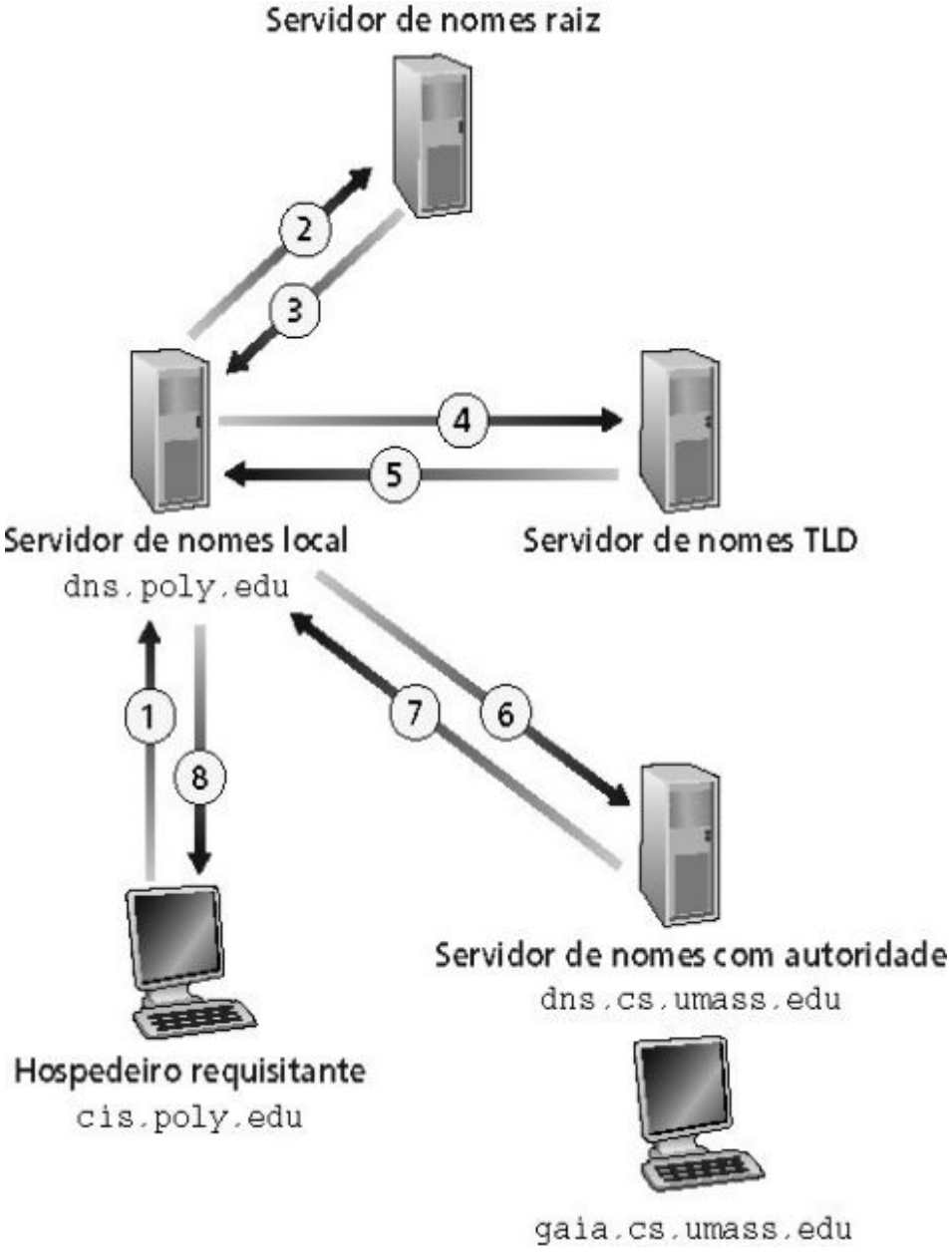
Fonte:(UEYAMA, 2014)

Para facilitar o entendimento do funcionamento da resolução de nomes através do emprego do conceito de servidor DNS, será utilizado o exemplo de acesso do hospedeiro em cis.poly.edu que quer o endereço IP para gaia.cs.umass.edu (UEYAMA, 2014).

Na figura 2.11 está a sequência de requisições para a resolução de nomes através do uso do DNS.

Pela figura 2.11, tem-se que primeiramente o Hospedeiro em cis.poly.edu, requisita ao servidor DNS local em dns.poly.edu, a resolução do nome gaia.cs.umass.edu

FIGURA 2.11: Exemplo da Resolução de Nomes para IP



Fonte:(UEYAMA, 2014)

(UEYAMA, 2014).

Em seguida, o servidor DNS local consulta o servidor raiz para saber qual é o servidor de nomes que consta a rede do endereço gaia.cs.umass.edu (UEYAMA, 2014).

Já no servidor de nomes, o DNS local consulta do servidor de nomes para saber qual o endereço do DNS do endereço gaia.cs.umass.edu (UEYAMA, 2014).

Por fim, ao consultar o servidor DNS local (dns.cs.umass.edu), ele obtem o IP do endereço gaia.cs.umass.edu (UEYAMA, 2014).

É importante salientar que esse método de resolução é conhecido como Consulta Encadeada que consiste em perguntar a um servidor que responde com o endereço de outro servidor mais próximo, assim sucessivamente, até que se chegue ao servidor de interesse. Além da Consulta Encadeada, existe a Consulta Recursiva que transfere a busca e resolução de nomes para o servidor de nomes consultando. No entanto, esse servidor de nomes que pode ser utilizado globalmente, acaba por sobrecarregar-se de solicitações tornando o sistema muito lento e por esse motivo pouco utilizado. Diferentemente do método de Consulta Recursiva, o método de Consulta Encadeada deixa a obrigação de processamento e busca de outros servidores nas mãos do servidor de DNS local, tornado assim a carga sobre os servidores globais muito mais leve (JUNIOR, 2018; UEYAMA, 2014).

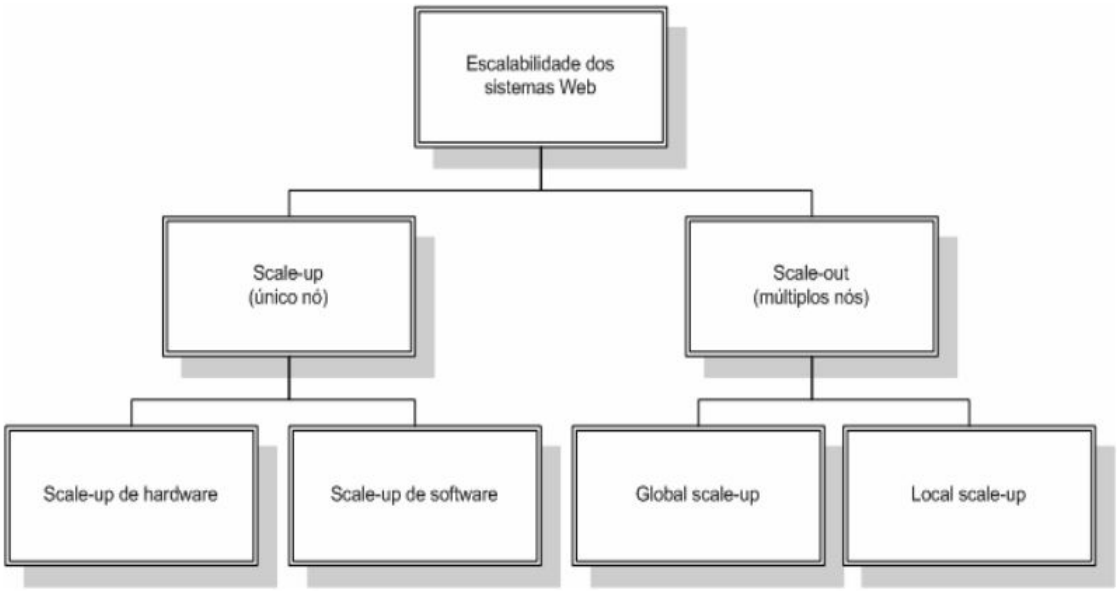
2.4 SERVIDOR *WEB*

Esses servidores são responsáveis por desempenhar uma das principais funções dentro da rede global de internet, junto com o crescimento da internet, foi necessário aumentar significativamente o número desses servidores para que fosse possível atender uma grande variedade se páginas de internet, com acesso rápido e eficiente. De fato, esses servidores estão incumbidos de fazer o armazenamento das páginas de *sites*, ao mesmo tempo que se comunicam com os navegadores dos usuários, como por exemplo o Google Chrome, e fornecem os dados necessários para que o próprio navegador sintetize a página em uma interface visual e de fácil uso (COMER, 2007; JUNIOR, 2008).

Para fazer a comunicação entre o usuário e o servidor, se faz necessário um protocolo para que não haja problemas de comunicação, segurança, entre outros. O protocolo utilizado é o HTTP, o número da porta HTTP.

Assumindo uma visão de alto nível de abstração, pode-se nomear três principais elementos para a composição de um servidor da rede de Internet, os navegadores ou *browsers*, a rede e os servidores. Todos esses elementos se comunicam através do modelo cliente-servidor, ou seja, a partir da demanda de um cliente, o servidor reponde retornando os pacotes solicitados. Esse modelo permite uma variedade de arquiteturas, na figura 2.12 tem-se as ramificações possíveis para o sistema (JUNIOR, 2008; TANENBAUM, 2010).

FIGURA 2.12: Possíveis Arquiteturas para Servidor *Web*



Fonte:(JUNIOR, 2008)

O método de nó único consiste em uma única máquina para atender os clientes, quando essa máquina passava a não suportar a demanda, então era feito o *scale-up* de *hardware* passando assim todos os processos, páginas e dados para uma máquina mais veloz e com maior capacidade de processamento. Uma outra forma de resolver o problema de demanda era o *scale-up* de *software*, onde tentava-se melhorar a performance do servidor através de um *software* mais eficiente. No entanto, com aumento exponencial da demanda por sites e páginas na internet, ambos os métodos não eram suficientes (JUNIOR, 2008).

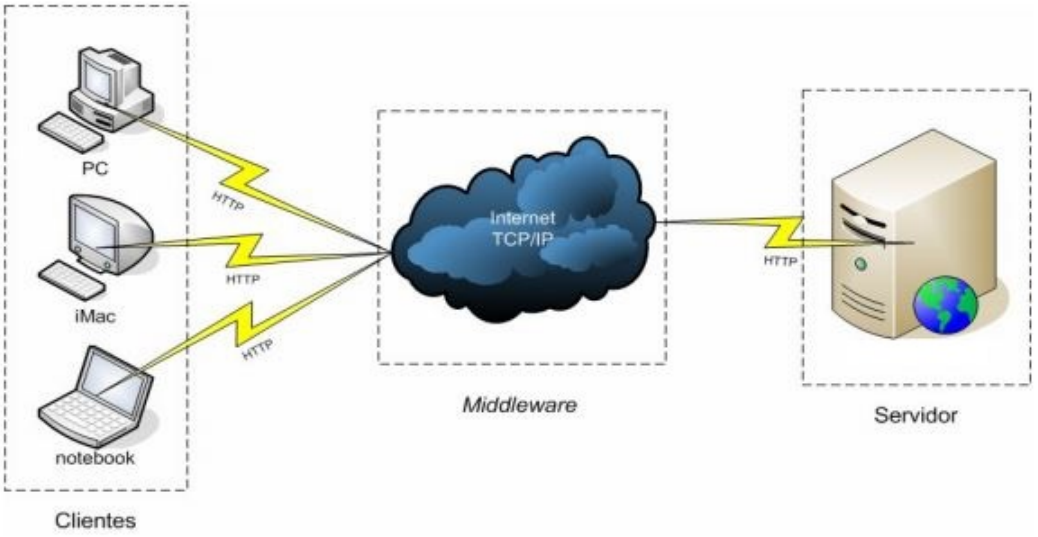
Já o método de múltiplos nós foi a saída encontrada para sanar o problema da demanda, neste caso, utiliza-se múltiplas máquinas poderosas para atender os clientes onde é possível fazer o *global scale-out* que consiste em espalhar os múltiplos servidores em várias localidades geográficas, deixando-os mais próximos do cliente e por consequência, aumentando a velocidade. O *local scale-out* é feito através do aumento de máquinas para atender em uma mesma rede que sofre sobrecarga (JUNIOR, 2008).

2.4.1 Requisições ao Servidor

O processo de requisição de uma página de internet é um fluxo complexo de transferência de dados entre vários protocolos e linguagens de programação. No geral, quanto mais funções o site apresentar, maior é o número de elementos envolvidos

entre a requisição da página ao servidor, o recebimento pelo navegador e a tradução da página. Um modelo simplificado de requisição pode ser visto na figura 2.13.

FIGURA 2.13: Fluxo de Requisição Cliente-Servidor



Fonte:(JUNIOR, 2008)

Na figura 2.13 está exemplificado um processo de três etapas. Primeiramente, o cliente faz a requisição da página através do navegador e normalmente por meio de funções chamadas *get* da linguagem HTML (JUNIOR, 2008; COMER, 2007).

Após isso, a requisição é enviada através do *middleware*, normalmente o *middleware* é a Internet ou até a rede local até a chegada ao servidor de destino, onde, por meio da resolução de nomes previamente discutida, é obtido o endereço do servidor (JUNIOR, 2008; COMER, 2007).

Finalmente a requisição chega ao local onde a página se encontra (servidor), o mesmo responde enviando os documentos requisitados. A requisição é feita através do protocolo HTTP e normalmente o servidor responde com um documento (código) em na linguagem de HTML, facilitando assim que o navegador transforme a informação recebida numa interface gráfica e de fácil uso para o usuário (JUNIOR, 2008; COMER, 2007).

2.5 DESENVOLVIMENTO DE API ANDROID

Android é um sistema operacional da google direcionado a dispositivos móveis, este sistema operacional é baseado em Linux, e assim como ele, se difere dos outros

sistemas operacionais por ser de desenvolvimento livre, sua logo pode ser vista na figura 2.14.

FIGURA 2.14: Logo Android

Fonte: (ANDROID, 2017).

Atualmente o sistema Android é o mais usado dentre os smartphones, desbancando concorrentes como IOS da Apple e o Windows Phone da Microsoft (BERGHER, 2017).

Para desenvolver aplicativos Android, a linguagem principal é java da Oracle, porém é necessário utilizar um kit de desenvolvimento de software, mais conhecido em inglês por SDK - software development kit. O SDK inclui um compreensivo conjunto de ferramentas de desenvolvimento, que incluem um depurador, uma biblioteca, um emulador, documentação, códigos de exemplos e tutoriais (ANDROID, 2017).

2.5.1 Android Studio

O software mais utilizado como ambiente de desenvolvimento de aplicativos é o Android Studio. Este ambiente foi especificamente desenvolvido para a criação de aplicativos Android e é mantido ativamente pelo Google (UDACITY, 2018).

Suas vantagens incluem o ambiente de programação com a utilização do java, ferramentas que auxiliam na criação dos leiautes visuais, bem como as funcionalidades do aplicativo, tudo isso juntamente com um emulador Android onde pode-se testar os aplicativos desenvolvidos (UDACITY, 2018).

2.5.2 Cursos para desenvolvimento de aplicativos

Existem diversos sites e empresas que oferecem cursos para desenvolvimento de aplicativos móveis, porém em sua maioria não são muito didáticos ou tem um preço de inscrição muito elevado.

A melhor opção provavelmente é o site Udacity que possui uma parceria com o Google para ofertar diversos cursos , entre eles os cursos focados em desenvolvimento

Alguns cursos ofertados pela Udacity:

• nivel: iniciante, curso: Android basics (Google);

• nivel: intermediário, curso: desenvolvedor Android (Google);

• nivel: avançado, curso: desenvolvimento de apps Android avançado (Google).

2.6 BANCO DE DADOS

Existem vários tipos de bancos de dados no mercado atual, porém um deles se destaca por ser de código aberto, assim como o sistema Android. O MySQL é um sistema gerenciado de banco de dados e é utilizado na maioria das aplicações de desenvolvimento gratuitas. Este sistema utiliza a linguagem SQL (Structure Query Language), que auxilia a inserir, gerir e acessar os dados armazenados (PISA, 2016).

Para poder utilizar o MySQL, é necessário que o usuário instale um servidor e uma aplicação cliente. O papel do servidor é fazer o armazenamento dos dados, e propiciar todas as ferramentas de utilização do mesmo, tais como: responder requisições; controlar a consistência; executar transações e operações

2.7 INTEGRAÇÃO: BANCO DE DADOS E APLICATIVO

Para que um aplicativo possa armazenar seus dados, que serão tratados de diversas formas para o usuário, é necessário que este utilize uma base de dados. As bases de dados podem estar armazenadas localmente, ou então em forma de servidores.

A forma de banco de dados externa é muito popular em aplicativos de dispositivos móveis, pois a capacidade local de armazenamento pode não ser tão robusta ao ponto de suportar um banco de dados. Tendo então a base de dados localizada externamente ao seu dispositivo, é necessário que algo auxilie na comunicação com estes dados.

Pode se fazer conexões diretas com as bases dados, porém isto implica numa dependência do modelo destes dados para fazer consultas, outro motivo de que não é recomendado fazer conexões diretas é a segurança, pois estes dados podem facilmente ser alterados ou manipulados.

O modo mais fácil e seguro de se fazer isto é utilizar uma API (application programming interface), que é uma aplicação que fará a interface entre o aplicativo e a base de dados. As tarefas executadas por essa API tem o objetivo de facilitar a aquisição e alimentação dos dados da base (CORDEIRO, 2016).

A figura 2.18 mostra o processo de funcionamento da API, conectando de forma fácil o dispositivo com sua base da dados. Para que algo seja gravado na base de dados, o dispositivo deve fazer uma requisição à API, que por sua vez fará a escrita desejada. O mesmo ocorre quando o dispositivo solicita algum dado, que lhe entrega é a API.

FIGURA 2.15: Funcionamento da API

Fonte: (CORDEIRO, 2016)

A API após criada e configurada, tem como função lidar com todas as conexões e manutenções da base de dados, e lidar com as requisições da aplicação que à utiliza, outra vantagem de utilizar uma API para controlar o banco de dados, é o fato de que o banco de dados pode ser facilmente alterado e recuperado (caso de backups).

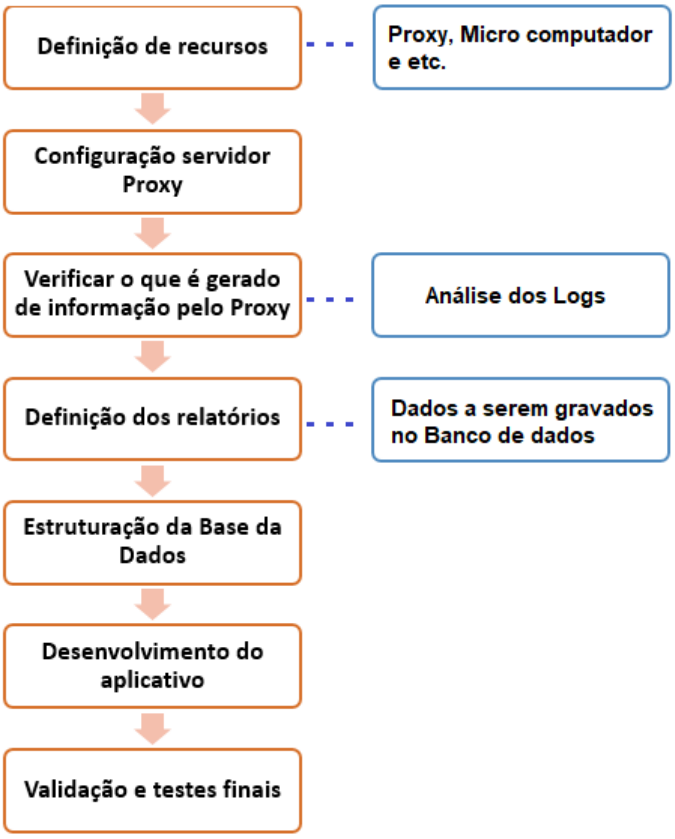
3 DESENVOLVIMENTO

Nesta seção, serão apresentados todos os objetos de desenvolvimento do projeto e que estão atrelados aos objetivos.

3.1 METODOLOGIA

A figura 3.1 apresenta um fluxograma que define a metodologia que será utilizada neste projeto.

FIGURA 3.1: Fluxograma com a metodologia do Projeto.

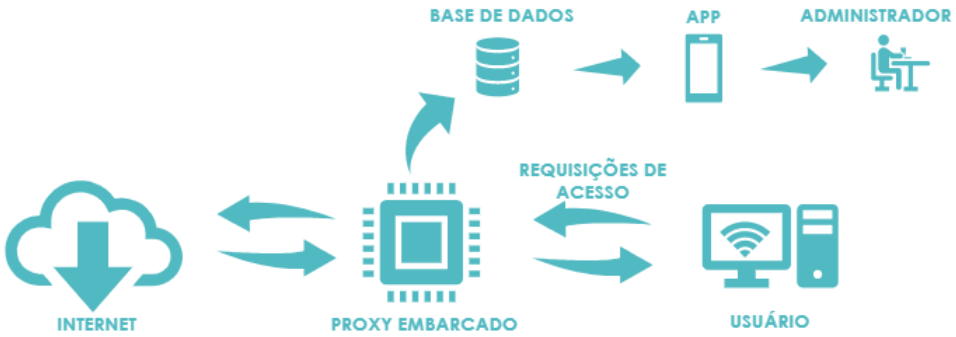


Já a figura 3.2 mostra um fluxograma que contém os itens do projetos, e os relacionamentos entre cada parte do projeto.

3.2 ESTRUTURA DA REDE DO SERVIDOR *PROXY*

Para que seja possível controlar uma rede é necessário que a fonte de controle tenha acesso às máquinas da rede e as informações que são recebidas ou enviadas

FIGURA 3.2: Fluxograma de itens do Projeto.



entre os clientes e servidores.

Essa abordagem requer uma arquitetura de rede diferenciada uma vez que os *hosts* não podem ter acesso direto à rede externa sem passar por um mecanismo de controle, segurança e gerenciamento de dados. Para tudo isso, deve-se utilizar o elemento servidor *proxy*.

Na figura 3.3, pode ser visto o modelo da rede proposto para o projeto. Esse modelo é baseado numa rede local de uma casa onde tem-se computadores e celulares conectados tanto por cabos quanto por WiFi.

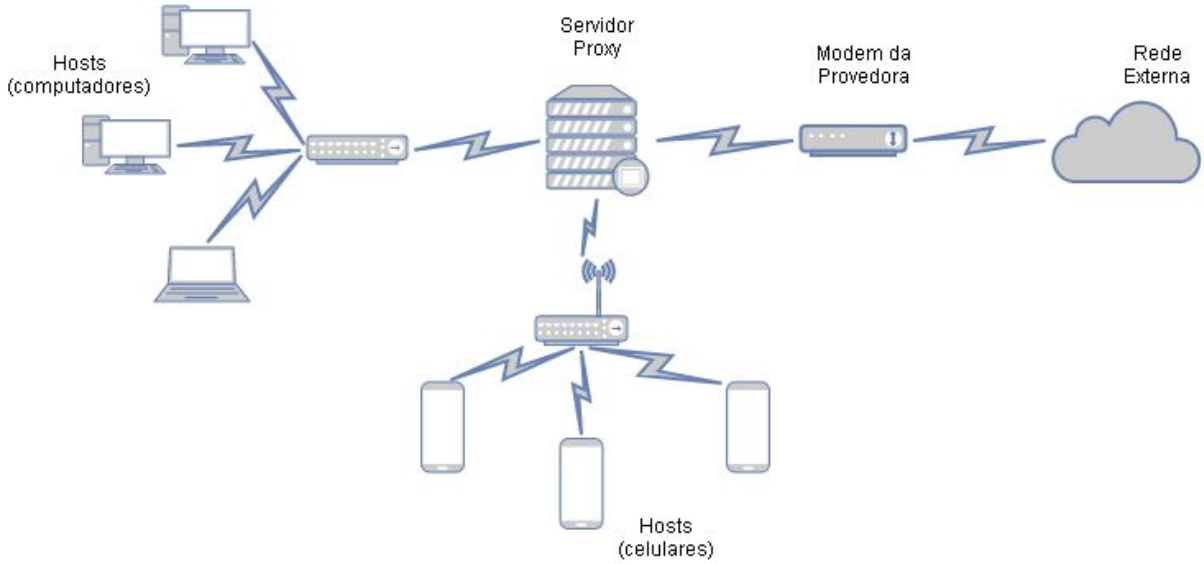


FIGURA 3.3: Arquitetura da rede do projeto.

Nota-se, na figura 3.3, que o Raspberry Pi permite que ele se comporte como servidor e como roteador.

Outro aspecto importante dessa arquitetura é que uma vez implantada, coloca o servidor *proxy* numa posição centralizadora na rede impedindo que os *hosts* se

comuniquem com agentes externos sem enviarem primeiro requisições ao Raspberry Pi. Quando o servidor recebe o pedido de um dos hosts, o mesmo pode decidir entre repassar a solicitação para a rede externa ou bloquear.

3.3 CONFIGURAÇÃO DO SQUID

Nesta seção serão apresentadas todas as configurações feitas no programa do Squid e o objetivo dessas alterações.

3.3.1 Direcionamento de Pacotes

Para que seja possível fazer o direcionamento dos pacotes para o Squid, é necessário primeiramente que os pacotes e requisições da rede sejam mandados para o Raspberry Pi que no caso trabalha como um servidor.

Assim, deve existir uma configuração para o DCHP que informa a todos os *hosts* da rede que devem se comunicar com o Raspberry Pi. O arquivo de configuração do DHCP é o /etc/dhcp/dhcpd.conf.

Dentro do arquivo dhcpd.conf está a configuração do quadro 3.1.

Quadro 3.1: Código de configuração do DHCP – Fonte: (RASPBIAN, 2018) alterado

```
1  subnet 10.1.1.0 netmask 255.255.255.0 {
2          range 10.1.1.10 10.1.1.240;
3          option domain-name-servers 10.1.1.249;
4          option domain-name "wpad.";
5          option routers 10.1.1.249;
6          option broadcast-address 10.1.1.255; # Broadcast
7          default-lease-time 600;
8  # Tempo para verificar se IP esta valido na rede
9          max-lease-time 7200;
10 # Tempo para o servidor verificar se ip esta sendo usado
11         host tp_repeter {
12                 hardware ethernet F4:F2:6D:9F:D1:53;
13                 fixed-address 10.1.1.250;
14         }
15         host printer {
16                 hardware ethernet 28:80:23:F5:8B:B8;
17                 fixed-address 10.1.1.9;
18         }
19 }
```

No quadro 3.1 nota-se que nas linhas 3 e 5 estão sendo direcionados o DNS e o *gateway*, respectivamente para o Raspberry Pi. Já *lease-time* define que se um determinado endereço MAC recebe um endereço IP, este endereço não será dado

para outro em mo mínimo 7200 seg, ou 2 horas. Se o computador ficar duas horas desligado, o endereço pode ser distribuído para outro.

No quadro 3.1, tem-se também o IP do *gateway* que consiste no endereço padrão que fará a comunicação entre a rede local e outra rede, ou seja, ele é o endereço utilizado quando deseja-se fazer a comunicação com o exterior. A configuração feita no DHCP faz assim, o direcionamento de todos os pacotes para o Raspberry (COMER, 2007).

No entanto, ainda é preciso que esses pacotes cheguem até o Squid. Para que seja direcionado todo o trafego para o Squid que está rodando dentro do Raspberry Pi (10.1.1.249) é utilizado o comando do quadro 3.2.

Quadro 3.2: Comando para Direcionamento de Portas – Fonte: (RASPBIAN, 2018) alterado

```
1 iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT
2 --to-port 3128
```

O comando do quadro 3.2, é rodado dentro de um arquivo iptables em shell script localizado no endereço /etc/rc.local, por esse motivo ele é executado ao iniciar o SO do Raspberry Pi. O iptables permite a configuração do filtro de pacotes do sistema Linux (NEGUS; BRESNAHAN, 2014).

3.4 BANCO DE DADOS EMBARCADO

Uma vez instalado no hardware do projeto, o MySQL já é capaz de receber comandos em SQL para acesso e configuração do banco. Nesta seção serão apresentados os principais comandos utilizados para a criação e estruturação do banco do projeto.

3.4.1 Ferramenta para Gestão de Banco de Dados - DbVisualizer

O DbVisualizer trata-se de uma rica ferramenta para desenvolvedores e administradores de banco de dados, provendo uma interface amigável e que suporta uma série de bancos de dados, incluindo o MySQL. Esse software permite a conexão com o banco de dados externo e a sua configuração, ao passo que facilita testes de *Queries*, construção de bancos, organização relacionamentos entre tabelas e todas as outras funções do SQL. Além disso, o DbVisualizer fornece uma ferramenta de *debug* tornando assim muito mais fácil a prototipagem (DBVISUALIZER, 2019).

A importância deste aplicativo no projeto vai muito além da facilidade na prototipagem. Como entrega do projeto, é preciso mostrar de forma clara como o trabalho está estruturado, neste quesito, o DbVisualizer disponibiliza uma visão compreensível e clara de como o banco está operando, desta forma, observou-se a necessidade de abordagem do mesmo no presente documento.

3.5 CONFIGURAÇÃO E ESTRUTURAÇÃO DO BANCO

3.5.1 Criação de Bancos de Dados

Neste projeto, optou-se pela construção de dois bancos de dados. O primeiro sendo o banco geral, contendo todas as colunas do *log* do Squid e que recebe o *Insert* das novas linhas de *log*. O segundo que é o resultado dos dados que serão mostrados no aplicativo móvel e que herdam dados do bando geral.

É importante salientar que a separação dos bancos é feita como forma de garantir o dinamismo da estrutura, pois, no futuro, caso seja necessário a criação de novos indicadores no aplicativo, não é necessário alterar o algoritmo que insere dados no banco. No quadro 3.3 está o comando utilizado para o DB do projeto.

Quadro 3.3: Comando de Criação de Banco – Fonte: (GROFF; WEINBERG, 2002) alterado

```
1 CREATE DATABASE <nome do banco>;
```

3.5.2 Criação de Tabelas

As tabelas de um banco são as fontes de informação, isto é, trata-se do lugar onde os dados estão efetivamente armazenados. É nesta etapa onde a definição dos dados que pretende-se mostrar se faz importante. Como as tabelas são acessadas pelos dispositivos externos que colocam ou solicitam informações da tabela, é de suma relevância a estruturação das tabelas para evitar o máximo consultas (*queries*) longas e que gerarão perda de performance do banco e por consequência lentidão para o usuário final da aplicação (GROFF; WEINBERG, 2002).

No quadro 3.4 está o comando utilizado para a criação das tabelas do projeto.

Quadro 3.4: Comando de Criação de Tabela – Fonte: (GROFF; WEINBERG, 2002) alterado

```
1 CREATE TABLE IF NOT EXISTS <nome da tabela> (
2     <nome da coluna primaria> INT AUTO_INCREMENT PRIMARY KEY,
3     <nome da coluna> <tipo>,
4     ...
5 )  ENGINE=INNODB;
```

3.5.3 Modelo Relacional

O modelo relacional de tabelas foi apresentado pelo Dr. Codd em seu artigo de 1970 como uma forma de simplificação da estrutura de DB, eliminando disposição explicita de herança e longas linhas de dados. O modelo de Codd explica que um banco de dados relacional é onde todos os dados visíveis ao usuário são organizados estritamente como tabelas de valores e onde todas as operações do usuário com o banco ocorrem nessas tabelas (GROFF; WEINBERG, 2002).

A definição deste modelo é entendida através do conceito de ordenação de valores em comum entre as mais diferentes tabelas de um modelo relacionado, isto né, um ponteiro que sinaliza que um valor da tabela A também está presente na Tabela B (GROFF; WEINBERG, 2002).

Na figura 3.4 está um exemplo simples de tabela relacional para uma aplicação de processamento de ordens.

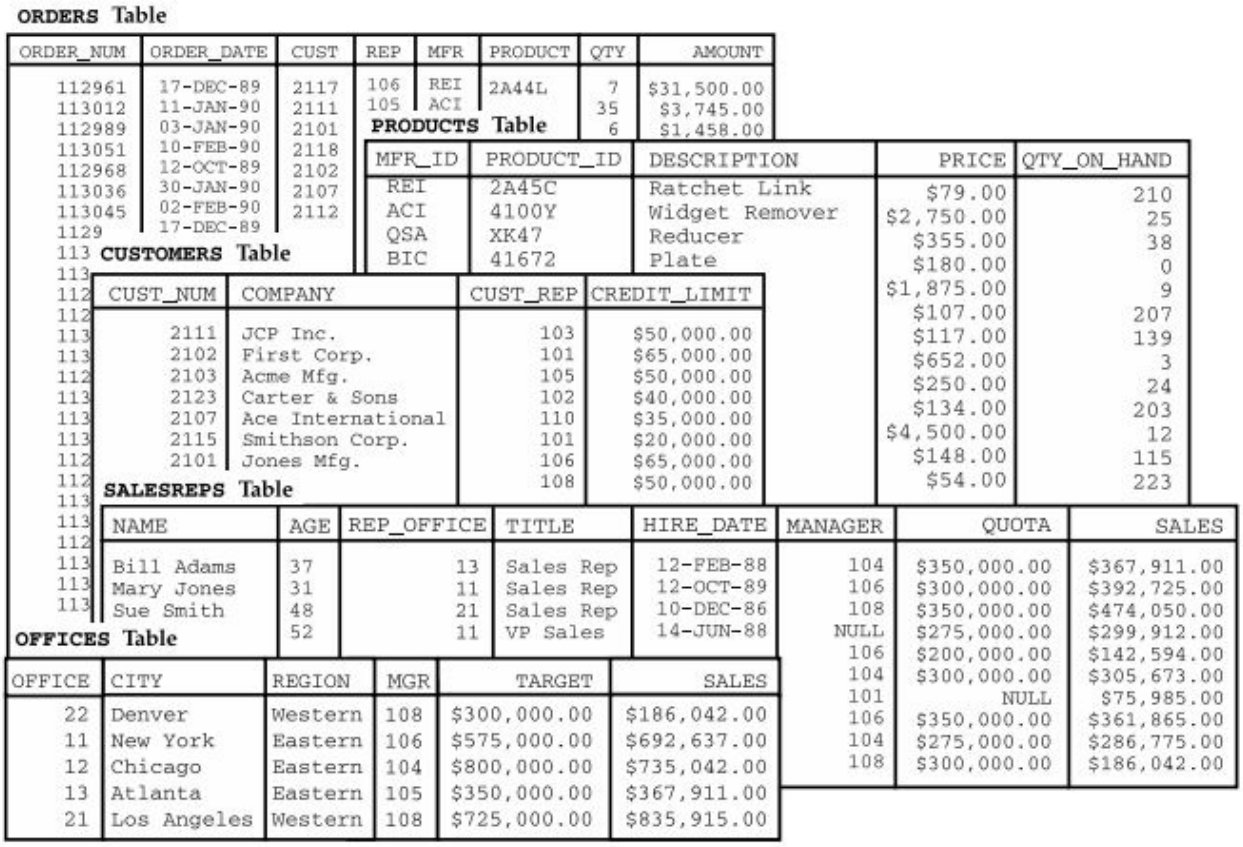
No modelo da figura 3.4 tem-se 5 tabelas onde a tabela chamada *CUSTOMERS*, armazena dados de clientes como nomes, limite de credito entre ouros. A tabela *SALESREPS* armazena dados dos funcionários de vendas. A tabela *OFFICES* armazena dados de cada escritório da empresa. A tabela *ORDERS* mantém o controle de cada de cada pedido feito pelos clientes e finalmente a tabela *PRODUCTS* apresenta os dados sobre os produtos disponíveis para vendas (GROFF; WEINBERG, 2002).

Neste estrutura, é visível a relação entre as tabelas através das primeiras colunas de cada tabela. Por exemplo, partindo de uma ordem, é possível saber qual foi a empresa que requisitou o produto por meio da tabela *CUSTOMERS* e da chave CUST da tabela *ORDERS*. O mesmo exemplo vale para as demais tabelas.

3.5.3.1 Chave Primária e Chave Estrangeira

Como as linhas de uma relação não estão necessariamente em ordem, não é possível selecionais a linha 1 de diferentes tabelas e esperar que elas estejam

FIGURA 3.4: Exemplo de Tabela Relacional



ORDERS Table

ORDER_NUM	ORDER_DATE	CUST	REP	MFR	PRODUCT	QTY	AMOUNT
112961	17-DEC-89	2117	106	REI	2A44L	7	$31,500.00
113012	11-JAN-90	2111	105	ACI		35	$3,745.00
112989	03-JAN-90	2101				6	$1,458.00
113051	10-FEB-90	2118					
112968	12-OCT-89	2102					
113036	30-JAN-90	2107					
113045	02-FEB-90	2112					
1129	17-DEC-89						

PRODUCTS Table

MFR_ID	PRODUCT_ID	DESCRIPTION	PRICE	QTY_ON_HAND
REI	2A45C	Ratchet Link	$79.00	210
ACI	4100Y	Widget Remover	$2,750.00	25
QSA	XK47	Reducer	$355.00	38
BIC	41672	Plate	$180.00	0
			$1,875.00	9
			$107.00	207
			$117.00	139
			$652.00	3
			$250.00	24
			$134.00	203
			$4,500.00	12
			$148.00	115
			$54.00	223

CUSTOMERS Table

CUST_NUM	COMPANY	CUST_REP	CREDIT_LIMIT
2111	JCP Inc.	103	$50,000.00
2102	First Corp.	101	$65,000.00
2103	Acme Mfg.	105	$50,000.00
2123	Carter & Sons	102	$40,000.00
2107	Ace International	110	$35,000.00
2115	Smithson Corp.	101	$20,000.00
2101	Jones Mfg.	106	$65,000.00
		108	$50,000.00

SALESREPS Table

NAME	AGE	REP_OFFICE	TITLE	HIRE_DATE	MANAGER	QUOTA	SALES
Bill Adams	37	13	Sales Rep	12-FEB-88	104	$350,000.00	$367,911.00
Mary Jones	31	11	Sales Rep	12-OCT-89	106	$300,000.00	$392,725.00
Sue Smith	48	21	Sales Rep	10-DEC-86	108	$350,000.00	$474,050.00
	52	11	VP Sales	14-JUN-88	NULL	$275,000.00	$299,912.00
					106	$200,000.00	$142,594.00
					104	$300,000.00	$305,673.00
					101	NULL	$75,985.00
					106	$350,000.00	$361,865.00
					104	$275,000.00	$286,775.00
					108	$300,000.00	$186,042.00

OFFICES Table

OFFICE	CITY	REGION	MGR	TARGET	SALES
22	Denver	Western	108	$300,000.00	$186,042.00
11	New York	Eastern	106	$575,000.00	$692,637.00
12	Chicago	Eastern	104	$800,000.00	$735,042.00
13	Atlanta	Eastern	105	$350,000.00	$367,911.00
21	Los Angeles	Western	108	$725,000.00	$835,915.00

Fonte: (GROFF; WEINBERG, 2002)

relacionadas, isso seria um erro grave, pois não existe primeira, décima ou última linha de uma tabela, existe apenas linhas quaisquer que ocupam essas posições nas mais variadas ordens e tabelas (GROFF; WEINBERG, 2002).

Para solucionar o problema mencionado, existem as chaves primarias (*primary keys*) e as chaves estrangeiras (*foreign keys*). As chaves primarias tratam-se de uma coluna ou combinação de colunas com valores únicos que identificam cada linha de uma tabela, como eplo tem-se a tabela *OFFICES* da figura 3.4, onde a coluna *OFFICE* possui valores únicos para cada linha. Já as chaves estrangeiras é uma coluna de uma tabela com valores que casam com as chaves primarias de outra tabela, assim, forma-se a relação entre tabelas (GROFF; WEINBERG, 2002).

Na linha 2 do quadro 3.4 está o comando utilizado criação de uma chave primária na tabela. Para a criação de uma chave estrangeira, tem-se o quadro 3.5.

Quadro 3.5: Tabela com Chave Estrangeira – Fonte: (GROFF; WEINBERG, 2002) alterado

```
1 CREATE TABLE IF NOT EXISTS <nome da tabela> (
2     id INT AUTO_INCREMENT PRIMARY KEY,
3     <nome da coluna> <tipo>,
4       ...
5     <nome da chave estrangeira> <tipo>,
6      FOREIGN KEY (<nome da chave estrangeira>)
7         REFERENCES <nome da tabela com chave primaria> (<nome da
8                                                 coluna com chave primaria>)
9 )  ENGINE=INNODB;
```

3.5.4 Inserindo Dados Relacionados na Tabela

Criar as tabelas com as relações devidamente configuradas é um passo importante e necessário, no entanto existe outra atividade igualmente importante, a inserção de dados. A inserção dos dados é relevante pois caso seja feita de forma incorreta, resultara em dados não confiáveis sendo exibidos ao usuário final. Além do mais, quanto mais estruturado o banco mais complexa é a inserção das informações, logo é essencial levar em conta as chaves primarias e estrangeiras (GROFF; WEINBERG, 2002).

Como no presente trabalho o banco é constituído de tabelas relacionadas que herdam dados de uma tabela geral, as inserções de dados consistem em consultas na base principal seguidas por a inclusão nos demais índices.

No quadro 3.6 está o comando em SQL para inserção relacional. Nota-se que a chave estrangeira é uma das colunas da tabela, pois é utilizada em consultas, além disso existe a verificação onde a chave estrangeira é igual a chave primaria, bem como o agrupamento. Este comando trata-se de um comando genérico, existem outras formas de obter-se resultados iguais (GROFF; WEINBERG, 2002).

Quadro 3.6: Inserção Relacional – Fonte: (GROFF; WEINBERG, 2002) alterado

```
1 INSERT INTO <tab. destino> (<col. destino>, <col. destino...>)
2 SELECT <col. destino>, <col. destino...>, <chave estrangeira>
3 FROM <tabela primaria>, <tab. origem dos dados>
4 WHERE <chave estrangeira> = <chave primaria> ...
5 GROUP BY <chave estrangeira>;
6 COMMIT;
```

Os comando utilizados neste projeto serão discutidos no capítulo de resultados.

3.6 ROTINAS DE ATUALIZAÇÃO DO BANCO DE DADOS

Nesta seção será apresentado todo o trajeto dos dados, partindo do *log* em txt do Squid, passando pelo programa que estrutura os dados e insere no MySql, bem como as rotinas de atualização que são configuradas no SO do Raspberry Pi.

3.6.1 Leitura do *Log*

O próprio Raspibian (SO do Raspberry Pi) apresenta uma solução para pivotação de arquivo de *log*, ou seja, arquivo contendo os dados da rede pode ser renomeado e salvado, ao passo que um novo é criado e começa a ser populado do zero. Essa função é chamada de *Log Rotate* e pode ser configurada no arquivo /etc/logrotate.d (GROFF; WEINBERG, 2002; RASPBIAN, 2018). No quadro 3.7 está a configuração de rotação para os dados da rede.

Quadro 3.7: Pivotação de Arquivo *Log* – Fonte: (RASPBIAN, 2018) alterado

```
1  #       Logrotate fragment for squid.
2  #
3  /var/log/squid/*.log {
4          daily
5          rotate 2
6          missingok
7          nocreate
8          sharedscripts
9          prerotate
10                 test ! -x /usr/sbin/sarg-reports || /usr/sbin/sarg-
                       reports daily
11         endscript
12         postrotate
13                 test ! -e /var/run/squid.pid || test ! -x /usr/sbin/
                       squid || /usr/sbin/squid -k rotate
14         endscript
15 }
```

O quadro 3.7 apresenta a configuração diária para rotação de dois *logs* que encontram-se no endereço /var/log/squid/. No presente trabalho, esta função é usada para contornar o problema de leitura de um arquivo que estaria sendo modificado pelo próprio Squid, com a pivotação é possível ler um arquivo que não está mais sendo usado, mas ao custo de não ter-se dados em tempo real.

Com a garantia de que o arquivo de *log* pode ser lido, tem-se a necessidade de organizar e quebrar as linhas de uma maneira que o MySql possa entender e manipular

os dados. No quadro 3.8, tem-se um trecho do arquivo accesslog2 que foi rotacionado e refere-se ao dia 17/11.

Quadro 3.8: Trecho do *Log*

```
1 1573983625.258      32 10.1.1.115 TCP_MISS/204 199 HEAD http://
    connectivitycheck.gstatic.com/generate_204 - ORIGINAL_DST
    /172.217.28.67 -
2 1573983660.454      71 10.1.1.159 TCP_MISS/204 199 GET http://clients3
    .google.com/generate_204 - ORIGINAL_DST/172.217.29.238 -
3 1573984030.964 75 10.1.1.86 TCP_TUNNEL/200 4149 CONNECT app-b03.lp1.
    npns.srv.nintendo.net:443 - HIER_DIRECT/3.85.76.222 -
```

O algoritmo desenvolvido em java capaz de ler este texto e inserir no banco de dados está disponível no Apêndice A. Sua estrutura foi construída seguindo o método do MVC (*Model-View-Controller*), onde o *Model* é a classe linha do *log* do Apêndice A.5 e as demais classes são responsáveis pelo *Controller*, ou seja, leitura e quebra dos textos para o formato do *Model* e finalmente a inserção no banco disponível no Apêndice B.1.

3.6.2 Rotinas de Atualização das Bases

Para agendar as rotinas de atualização de base, tem-se o Cron. O Cron é um programa Linux que roda em segundo plano e é responsável por efetuar a varredura pelo *script* de processos agendados pelos usuários, o *script* com as instruções é a Crontab (NEGUS; BRESNAHAN, 2014).

No quadro 3.9, está a Crontab do servidor *proxy* do projeto.

Quadro 3.9: Crontab do Servidor *Proxy*

```
1 00 7 * * * /root/logMysqlUpdate.sh
2 00 4 1 * * reboot
3 00 2 * * 0 /root/scriptShellTCC.sh
```

Pelo quadro 3.9, na linha 1, verifica-se que o programa em Shell, logMysqlUpdate.sh, roda todos os dias a 07:00 da manhã, fazendo a inserção dos dados, conforme o Apêndice A. Na linha 2 está configurado a reinicialização do Raspberry Pi toda segunda-feira, as 04:00 da manhã. Na linha 3, o Cron roda o programa em shell, scriptShellTCC.sh, todos os domingos às 02:00 da manhã. Este programa atualiza as tabelas relacionadas que são utilizadas no aplicativo móvel.

O programa logMysqlUpdate.sh pode ser visto no quadro 3.10, neste quadro está presente a chamada do programa LogMiner em java, que pode ser visto no Apêndice A.3.

Quadro 3.10: Programa logMysqlUpdate.sh

```
1 date>>/media/usb/Insert.log
2 java -jar /media/usb/squid/LogMiner.jar /var/log/squid/access.log
      .2>>/media/usb/Insert.log
3 date>>/media/usb/Insert.log
```

3.6.3 Atualização das Tabelas Relacionadas

No quadro 3.9, tem-se o agendamento do programa scriptShellTCC.sh. Este programa é responsável por executar o arquivo em SQL do Apêndice B. Portanto, uma vez por semana a Cron roda o programa em SQL que deleta os últimos 3 meses de dados das tabelas e insere novamente os 3 meses atualizados através das *queries* da base principal e respeitando as relações entre chaves primarias e chaves estrangeiras.

3.7 DESENVOLVIMENTO DO APLICATIVO MÓVEL E RELATÓRIOS

O desenvolvimento do aplicativo passou primeiro pela definição dos dados que seriam mostrados. Tendo como base o *log* do Squid que disponibiliza as seguintes informações, conforme as colunas da figura 4.3:

3.7.1 Desenvolvimento dos Relatórios

- Data e hora da requisição (squidtimestamp);
- Quantidade de tempo em milissegundos para a requisição (tempo);
- Endereço local de IP (ipadress);
- Resultados da transação (resultCode);
- Tamanho de dados trafegados (bytes);
- Método de requisição (requestMethod);
- URL acessada (fulUrl);
- Identidade do requisitante, caso tenha (user);
- Código que mostra como a requisição foi trabalhada (hierarchyCode),
- Mimetype que identifica o tipo do documento ou dado (typeReq).

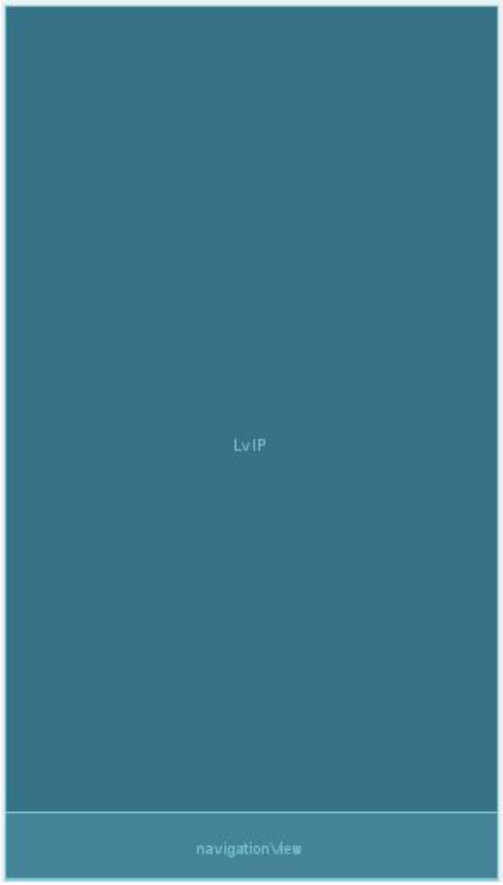
Observou-se que os relatórios teriam informações de contagem de acessos, quantidade de volume de dados trafegados, filtragem de endereços de sites (redes sociais), o tipo dos dados e as falhas de requisições na rede identificadas pelo request-Method igual a NONE. Essas informações são categorizadas pelo IP, mês e ano de acesso, em alguns casos, pelo horário em que foi requisitado. Desta forma é possível reduzir o volume de dados das bases e a velocidade das consultas feitas pelo aplicativo.

3.7.2 Criação das Telas e Classes do Aplicativo

3.7.2.1 Layout de Objetos da *Main Activity* - Tela Inicial

O código desta tela pode ser visto no Apêndice C.1. Como tratando-se da tela inicial, ela é a responsável botão central da aba no rodapé. Foi utilizado uma tabela que contêm a lista dos IPs que algum dia passaram pela rede. O arranjo desta tela pode ser visto na figura 3.5.

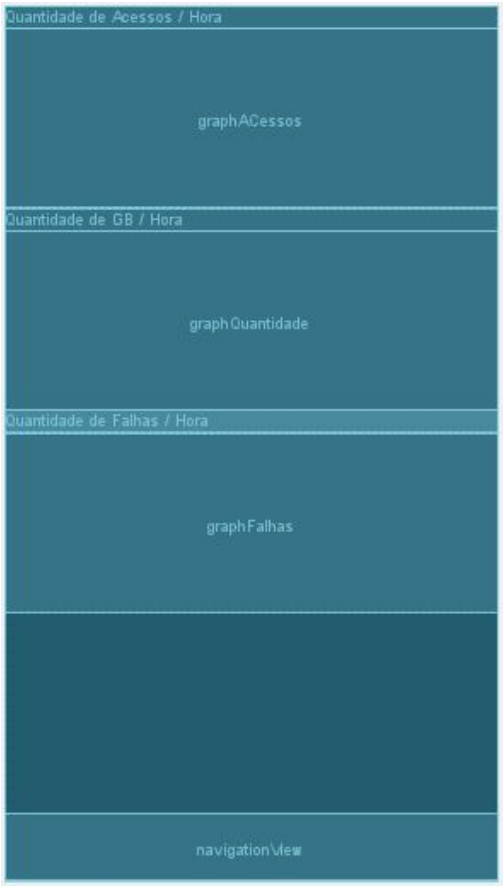
FIGURA 3.5: Layout de Objetos da Tela Inicial



3.7.2.2 Layout de Objetos da Tela de Relatório de Tráfego da Rede

Esta tela é chamada ao clicar em dados na aba do rodapé. O *layout* da tela responsável pelo relatório da rede pode ser visto na figura 3.6. Esta tela apresenta três gráficos, cada um mostrando informações de quantidade de acessos ao longo das horas do dia em um mês, bem como o volume de dados e a quantidade de falhas. Além disso, através do código que está no Apêndice C.5, nota-se que existe um item para seleção de calendário (*picker view*).

FIGURA 3.6: Layout de Objetos da Tela de Tráfego da Rede



3.7.2.3 Layout de Objetos da Tela de Pesquisa em URL

Esta tela é acionada através do botão de busca presente na aba do rodapé. Seu arranjo pode ser visto na figura 3.7 e nota-se que a mesma contêm apenas uma tabela, no entanto, ao analisar o código do Apêndice C.6, constata-se que a barra do cabeçalho possui um campo de busca que é inserido junto á consulta que é enviada ao banco de dados, está consulta é feita em outra classe.

FIGURA 3.7: Layout de Objetos da Tela de Pesquisa por URL

Fonte: Dos autores

Nesta *activity* foi preciso a consulta direta ao banco de dados principal, gerando um tempo de resposta elevado. Para contornar o problema, foi implementado um *progress dialog* de carregamento com uma *thread*, resultando numa tela de *loading*.

3.7.2.4 Classe de Conexão ao Banco de Dados

Para facilitar as consultas ao banco Mysql, foi implementada a classe DBConnection do Apêndice C.2, ela contem as credenciais e endereços para acesso ao banco e efetivamente cria uma conexão com o DB.

3.7.2.5 Classe de Consulta ao Banco de Dados

A classe que consulta o banco de dados pode ser vista no Apêndice C.3, além de utilizar a a classe de conexão, está classe faz a busca com base nos parâmetros enviados no DB e popula listas e vetores que são utilizados em outras classes. Ao final, a conexão é encerrada.

3.7.2.6 Layout de Objetos da Tela de Dados do IP

Para esta tela, tem-se informações sobre a quantidade de acessos e o volume de dados que determinado IP, selecionado na tela inicial, apresenta. Por meio do código do Apêndice C.7 evidencia-se que esta tela apresenta também um botão para seleção do mês e ano que faz a busca dos três sites ais acessados pelo usuário em determinado mês e atualiza a tabela na região inferior da tela disponível na figura 3.8.

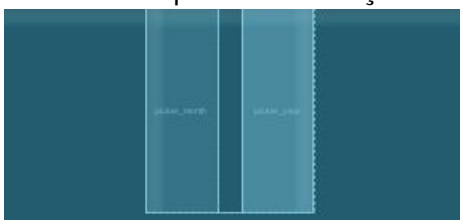
FIGURA 3.8: Layout de Objetos da Tela de Dados do IP

Nesta *activity* foi necessária também uma consulta direta ao banco de dados principal para gerar os três sites mais acessados, gerando um tempo de resposta mais elevado. Para contornar o problema foi implementado uma tela de *loading*.

3.7.2.7 Layout de Objetos do Adaptador de Seleção de Mês e Ano

Para permitir a escolha do mês e do ano em várias telas do aplicativo, foi construído esse adaptador para exibir as opções ao usuário. O código deste adaptador está no Apêndice C.4 e seu arranjo pode ser visto na figura 3.9.

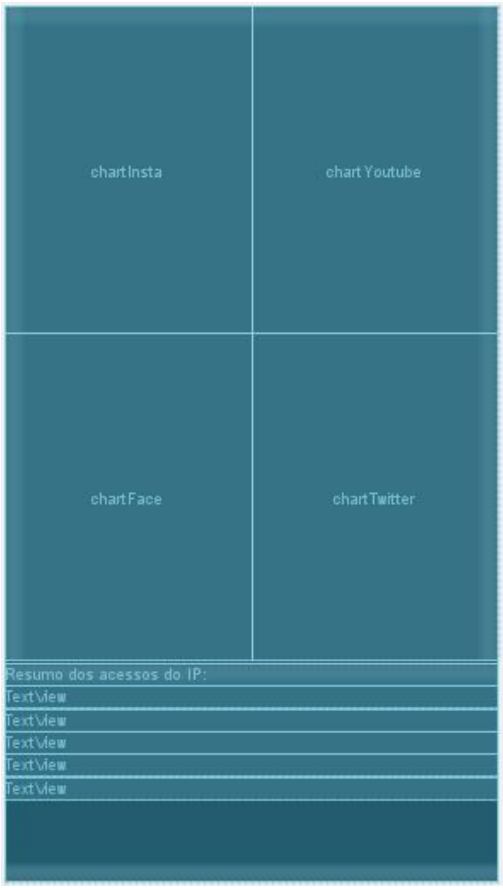
FIGURA 3.9: Adaptador de Seleção Mês/Ano



3.7.2.8 Layout de Objetos da Tela de Redes Sociais

A tela de redes sociais é obtida também através do IP, ela apresenta quatro gráficos de pizza, cada um representando uma das redes sociais (Facebook, Instagram, Youtube e Twitter). Os textos na parte inferior da tela mostram todos os dados obtidos nas consultas feitas por essa *activity*. O arranjo dos objetos desta tela podem ser vistos na figura 3.10.

FIGURA 3.10: Layout de Objetos da Tela de Redes Sociais



Essa tela também implementa a seleção de data/ano através da um item na barra superior. Já seu código está no Apêndice C.8.

3.7.2.9 Layout de Objetos da Tela de Tipos de Dados Trafegados

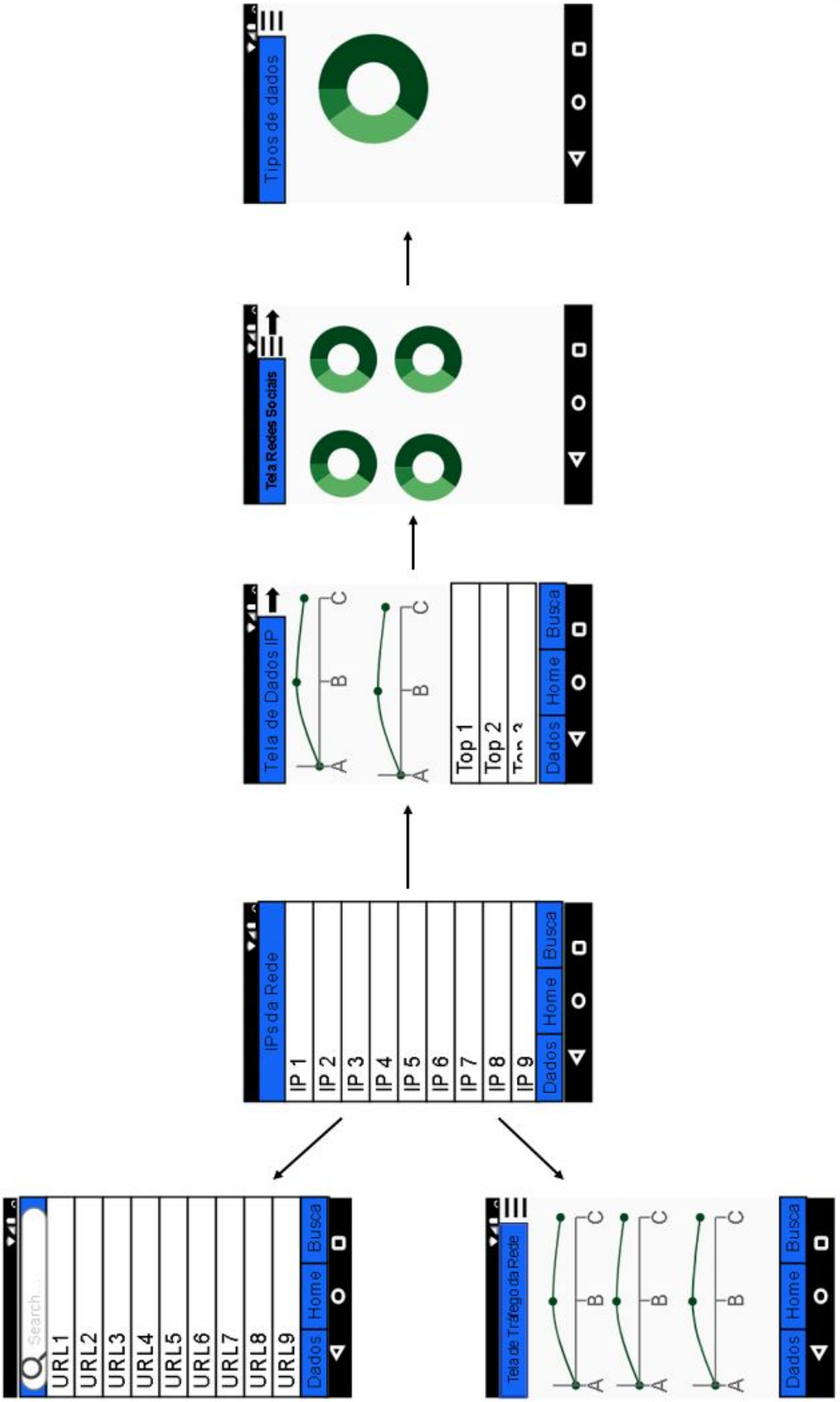
Para os tipos de dados trafegados, tem-se figura 3.11. Nota-se que existe um gráfico de pizza que mostra todos os tipos de dados que o IP selecionado na tela inicial acessou no período (mês/ano) escolhido também através da *picker view*. O código desta *activity* está no Apêndice C.9.

FIGURA 3.11: Layout de Objetos da Tela de Tipos de Dados Trafegados

3.7.3 *Mockup* e Fluxo das Telas do Aplicativo

Para auxiliar no desenvolvimento e o entendimento do fluxo das telas, foi montado o *Mockup* da figura 3.12. Nota-se a centralização da tela contendo os IPs, uma vez que esse IP é utilizado numa cadeia de telas que passam como parâmetro para a execução de consulta ao banco de dados.

FIGURA 3.12: Mockup do Aplicativo do Projeto



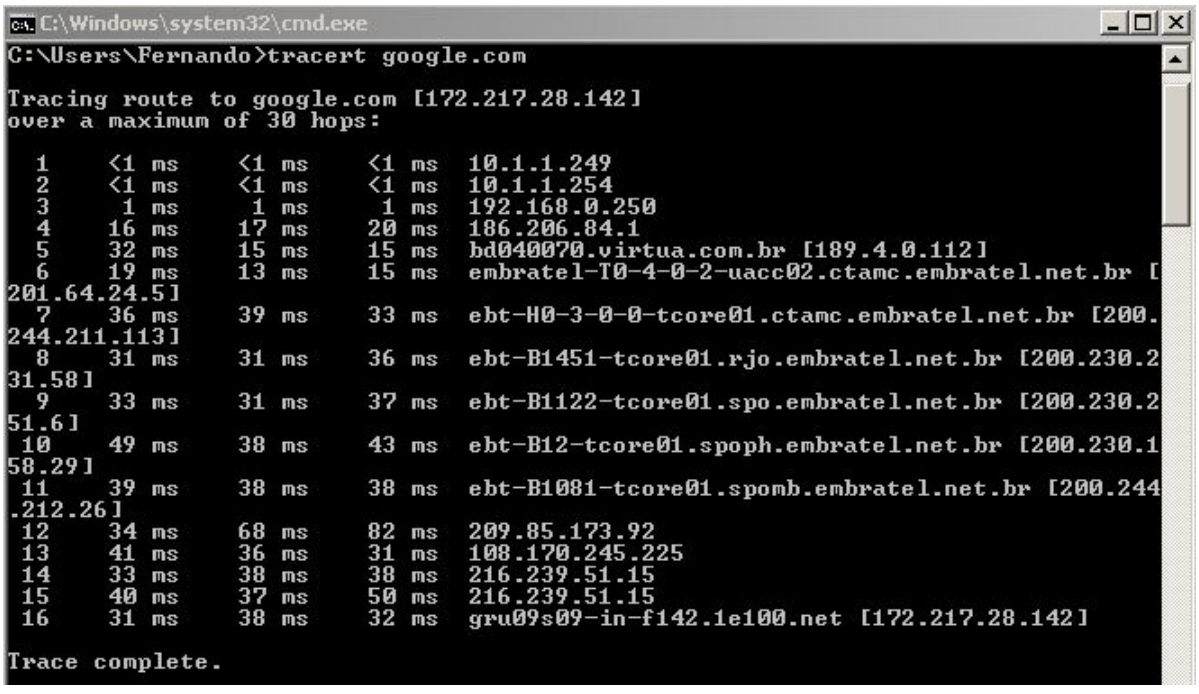
4 RESULTADOS

4.1 RESULTADOS DA CONFIGURAÇÃO DE DIRECIONAMENTO DO SERVIDOR *PROXY*, DNS E DCHP

Para mostrar como o direcionamento funciona, pode ser utilizado um teste em um dos *hosts* da rede. Através do comando tracert é possível mapear, de forma simples, quais são os roteadores por onde os pacotes passam. O teste proposto aqui é tracert google.com, isto é, verificar para onde é direcionado os pacotes até chegar no endereço do Google.

O resultado do teste feito em um computador que está conectado num *switch* e que este encontra-se ligado ao servidor *proxy* (conforme figura 3.3) pode ser visto na figura 4.1.

FIGURA 4.1: Resultado do comando TRACERT



Nota-se na figura 4.1 que acontecem três *pings* dentro da própria rede (pois o tempo para resposta foi de 1 ms).

O primeiro lugar para onde o pacote vai é o 10.1.1.249, ou seja, diretamente para o Raspberry Pi, mostrando assim o direcionamento.

O segundo lugar para onde o pacote é direcionado é o modem da provedora

(vide figura 3.3).

Em terceiro lugar, os pacotes vão para a provedora da internet. Os outros passos de fato são os que se encontram fora da rede e que leva o pacote até o servidor da Google.

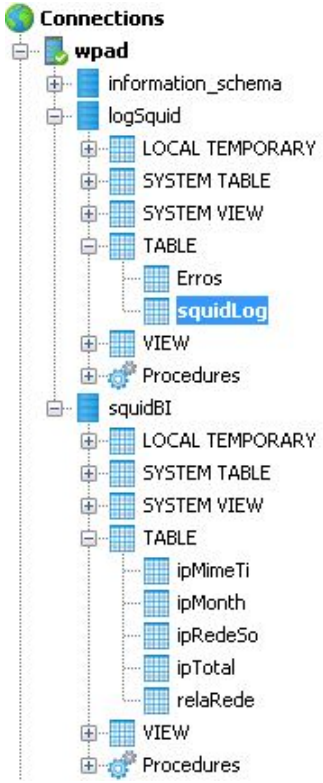
Com esse teste, é possível ver que todo acesso a rede externa está sendo passado para o Raspberry Pi, permitindo assim que sejam feitos dispositivos de controle e segurança dentro do próprio servidor *proxy*.

4.2 RESULTADOS DA LEITURA E INSERÇÃO DE *LOG* PARA BANCO DE DADOS

O código e java desenvolvido para a leitura dos arquivos de texto gerados pelo Squid e inserção no banco de dados podem ser visto do Apêndice A. Conforme explicado no desenvolvimento, este programa é rodado todos os dias pelo Cron ao passo que o LogRotate do Raspibian atualiza o arquivo diariamente que será inserido no banco.

Na figura 4.2 tem-se a imagem do DbVisualizer com o banco de dados principal (logSquid) bem como o banco com as tabelas relacionais (squidBI).

FIGURA 4.2: Resultado dos Bancos Criados



Já na figura 4.3 está um trecho com as 10 primeiras linhas da tabela principal chamada squidLog que é populada através do programa em java.

FIGURA 4.3: Resultado da Tabela Principal

Table: squidLog

wpad/logSquid/TABLE/squidLog

Info | Columns | Data | Row Count | Primary Key | Indexes | Grants | Row Id | References

In **DbVisualizer Pro** this feature includes data editing functionality and management of binary/BLOB data.

*	id	squidtimestamp	tempo	ipadress	resultCode	bytes	requestM...	fullUrl	user	hierarchyCode	typeReq
1	303...	2019-01-01 00:00:00	23	10.1.1.37	TCP_MISS/204	199.0	GET	http://clients3.goo...	-	ORIGINAL_D...	-
2	303...	2019-01-01 00:00:00	24	10.1.1.37	TCP_MISS/204	199.0	GET	http://clients3.goo...	-	ORIGINAL_D...	-
3	303...	2019-01-01 00:00:02	351	10.1.1.163	TCP_MISS/200	20985.0	GET	http://www.bing.c...	-	ORIGINAL_D...	text/html
4	303...	2019-01-01 00:00:53	374	10.1.1.153	TCP_REFRESH...	198.0	GET	http://connectivity...	-	ORIGINAL_D...	-
5	303...	2019-01-01 00:00:58	108	10.1.1.97	TCP_MISS/200	1732.0	GET	http://finance.serv...	-	ORIGINAL_D...	application/xml
6	303...	2019-01-01 00:00:58	420	10.1.1.97	TCP_MISS/200	1583.0	GET	http://pt-br.appex...	-	ORIGINAL_D...	application/xml
7	303...	2019-01-01 00:00:58	369	10.1.1.97	TCP_MISS/200	1647.0	GET	http://pt-br.appex...	-	ORIGINAL_D...	application/xml
8	303...	2019-01-01 00:01:06	240473	10.1.1.225	TCP_TUNNEL/200	1014.0	CONNECT	picasaweb.google....	-	HIER_DIRECT...	-
9	303...	2019-01-01 00:01:10	126	10.1.1.163	TCP_MISS/302	894.0	GET	http://www.youtu...	-	ORIGINAL_D...	text/html
10	303...	2019-01-01 00:01:17	440	10.1.1.153	TCP_MISS/200	1115.0	POST	http://198.11.136....	-	ORIGINAL_D...	application/json

Em relação aos parâmetros de criação desta tabela tem-se, na figura 4.4, estão os tipos que definem a tabela principal.

FIGURA 4.4: Diagrama da Tabela Principal

4.3 RESULTADOS DA ESTRUTURAÇÃO E ATUALIZAÇÃO DAS TABELAS RELACIONADAS

Seguindo a o processo de criação de tabelas relacionadas descrito no capitulo de desenvolvimento, observou-se que seria necessário a construção de cinco tabelas

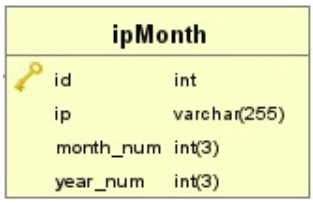
para atender os relatórios. Entre essas tabelas, três estariam relacionadas através de uma outra tabela e a última teria índices próprios.

4.3.1 Tabela de Chave Primária: ipMonth

Esta tabela consiste no agrupamento dos IPs, meses e anos dos registros de *log* que estão na tabela principal da figura 4.3

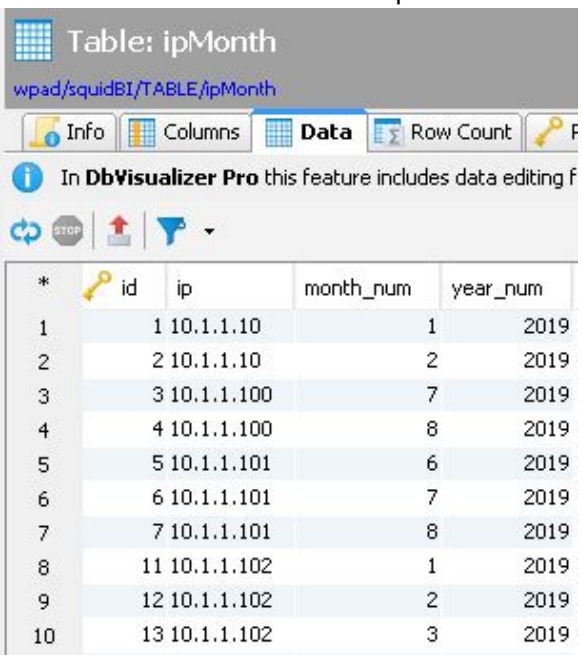
Tratando-se de uma tabela de índices, tem-se poucas colunas, como pode ser visto no diagrama da figura 4.5.

FIGURA 4.5: Diagrama da Tabela ipMonth



A as primeiras linhas da ipMonth já populada podem ser vistas na figura 4.6.

FIGURA 4.6: Tabela ipTotal



*	id	ip	month_num	year_num
1	1	10.1.1.10	1	2019
2	2	10.1.1.10	2	2019
3	3	10.1.1.100	7	2019
4	4	10.1.1.100	8	2019
5	5	10.1.1.101	6	2019
6	6	10.1.1.101	7	2019
7	7	10.1.1.101	8	2019
8	11	10.1.1.102	1	2019
9	12	10.1.1.102	2	2019
10	13	10.1.1.102	3	2019

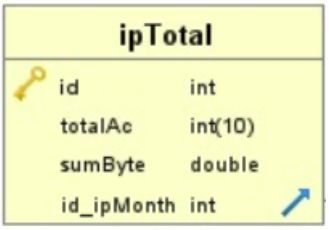
Neste formato, é possível verificar a existência de um ID único, para uma combinação única de IP, mês e ano. E esta portanto será a configuração que existe nas tabelas que se relacionam com ela.

4.3.2 Tabela de Quantidade de Acessos e Bytes: ipTotal

Esta tabela é responsável por carregar os valores que são utilizados nos relatórios de acessos por IP na rede e também em um dos campos do relatório de redes sociais.

O diagrama desta tabela pode ser visto na figura 4.7.

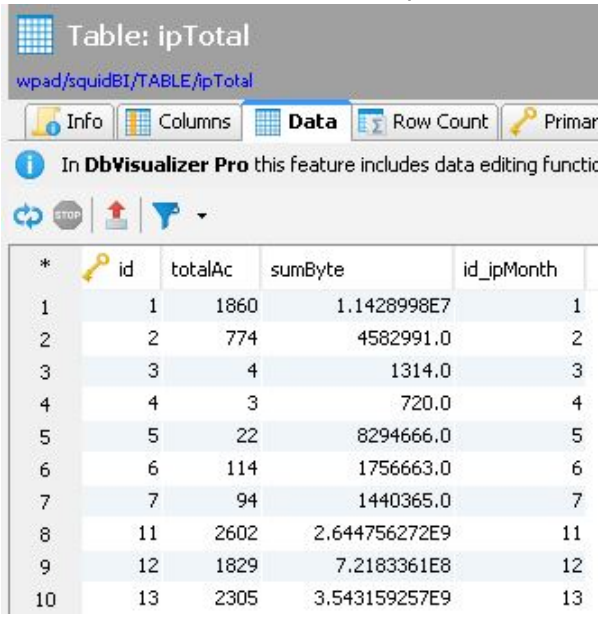
FIGURA 4.7: Diagrama da Tabela ipTotal



Na figura 4.7 é possível verificar que a coluna id_ipMonth está a chave estrangeira que vem da tabela de índices ipMonth.

Já a tabela propriamente dita pode ser vista na figura 4.8.

FIGURA 4.8: Tabela ipTotal



*	id	totalAc	sumByte	id_ipMonth
1	1	1860	1.1428998E7	1
2	2	774	4582991.0	2
3	3	4	1314.0	3
4	4	3	720.0	4
5	5	22	8294666.0	5
6	6	114	1756663.0	6
7	7	94	1440365.0	7
8	11	2602	2.644756272E9	11
9	12	1829	7.2183361E8	12
10	13	2305	3.543159257E9	13

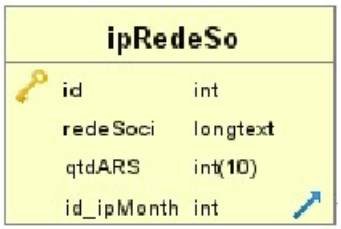
Constata-se que existem duas colunas principais com dados, a coluna totalAc que contem a quantidade total de acessos feitos por determinado IP, em certo mês e ano. Da mesma forma, a coluna sumByte possui a soma total dos bytes para determinado IP, mês e ano.

4.3.3 Tabela de Acessos as Redes Sociais: ipRedeSo

Esta tabela é utilizada no relatório de redes sociais e mostra, dentro da rede os a quantidade de acesso ao Facebook, Youtube, Instagram e Twitter para determinado IP, mês e ano (conforme chave primária).

O diagrama desta tabela pode ser visto na figura 4.9.

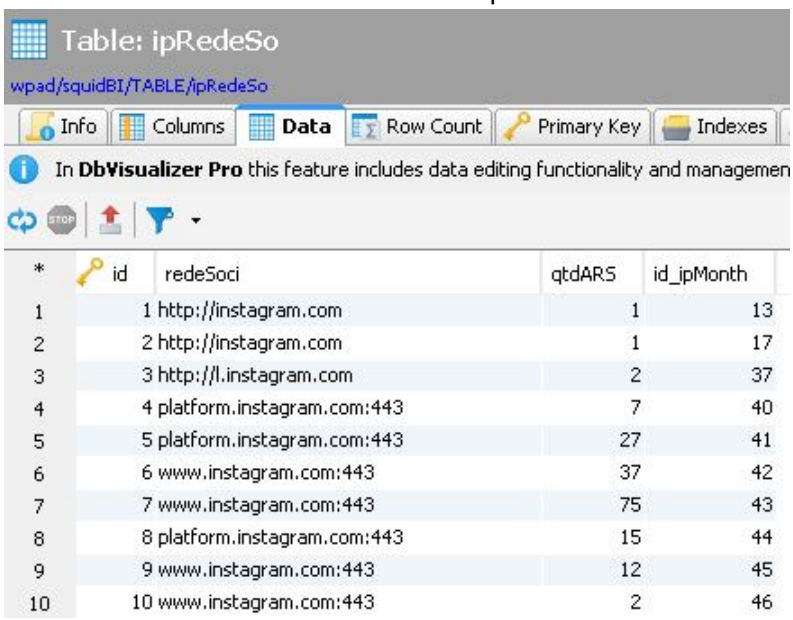
FIGURA 4.9: Diagrama da Tabela ipRedeSo



Verifica-se também a existência da chave estrangeira e consequentemente a relação entre ela e a ipMonth.

Com relação aos dados da tabela, os mesmos podem ser visto na figura 4.10. Percebe-se que a tabela possui a URL, com o texto Facebook, Youtube, Instagram ou Twitter e ao lado a quantidade de acessos. Para evitar os mais variados tipos de URL que cada página pode ter, optou-se pela quebra da URL na terceira barra, e através do agrupamento dessas URLs chegou-se no resultado da figura.

FIGURA 4.10: Tabela ipRedeSo



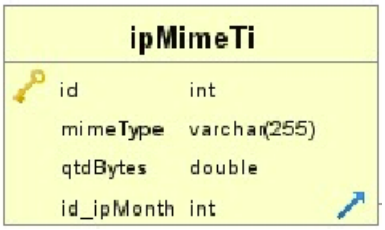
*	id	redeSoci	qtdARS	id_ipMonth
1	1	http://instagram.com	1	13
2	2	http://instagram.com	1	17
3	3	http://l.instagram.com	2	37
4	4	platform.instagram.com:443	7	40
5	5	platform.instagram.com:443	27	41
6	6	www.instagram.com:443	37	42
7	7	www.instagram.com:443	75	43
8	8	platform.instagram.com:443	15	44
9	9	www.instagram.com:443	12	45
10	10	www.instagram.com:443	2	46

4.3.4 Tabela de Tipos: ipMimeTi

A tabela ipMimeTi foi criada para o fornecimento de dados pré preparados para o relatório de tipos de dados trafegados por usuário (mimetype). Seu diagrama pode ser visto na figura 4.11.

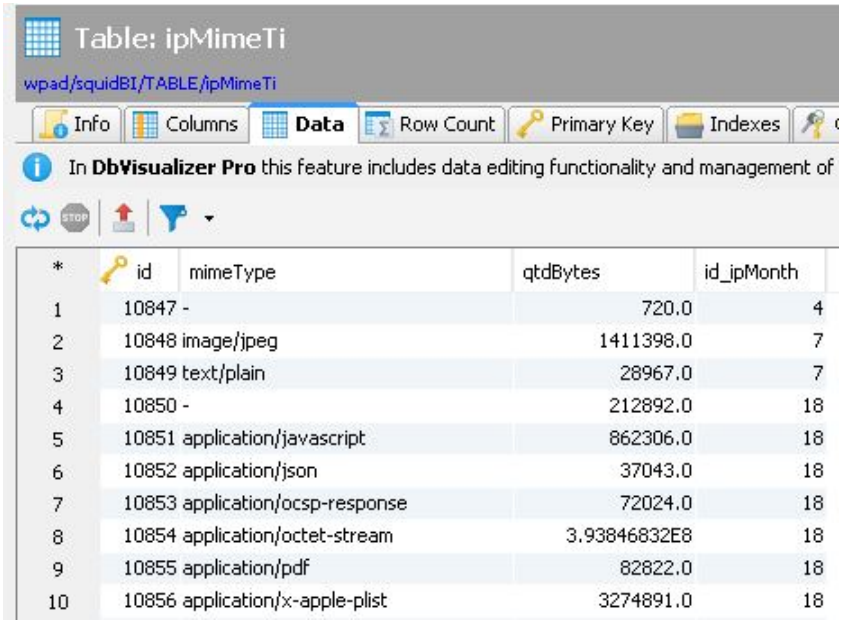
Nesta tabela tem-se o Mimetype, isto é, um identificador de tipo de dado. Outra coluna importante é a quantidade de Bytes, portanto, neste item tem-se a relação entre IP, mês e ano para cada Mimetype trafegado e a quantidade total de Bytes trafegado.

FIGURA 4.11: Diagrama da Tabela ipMimeTi



A tabela populada pode ser vista na figura 4.12. É importante salientar que a chave estrangeira também está presente, coluna id_ipMonth.

FIGURA 4.12: Tabela ipMimeTi



*	id	mimeType	qtdBytes	id_ipMonth
1	10847	-	720.0	4
2	10848	image/jpeg	1411398.0	7
3	10849	text/plain	28967.0	7
4	10850	-	212892.0	18
5	10851	application/javascript	862306.0	18
6	10852	application/json	37043.0	18
7	10853	application/ocsp-response	72024.0	18
8	10854	application/octet-stream	3.93846832E8	18
9	10855	application/pdf	82822.0	18
10	10856	application/x-apple-plist	3274891.0	18

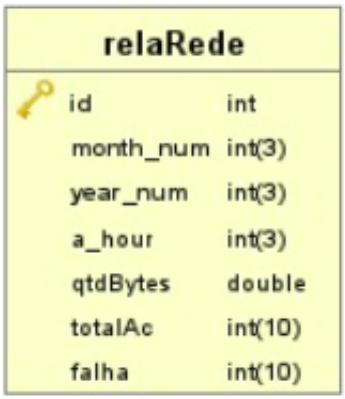
4.3.5 Tabela de Tráfego da Rede: relaRede

A tabela relaRede é utilizada para gerar o relatório de trafego na rede ao longos das horas do dia dos meses e anos. Neste caso específico, como o objetivo desta

tabela não é mostrar dados por usuários mas sim dados da rede como um todo, a inserção de uma coluna com a hora do dia e a exclusão da coluna IP não permite a relação entre ela e a ipMonth.

O diagrama contendo pode ser visto na figura 4.13. Verifica-se que existem as colunas de mês, ano, hora, quantidade de Bytes, quantidade de acessos e quantidade de falhas. A quantidade de falhas é contada através da tabela principal (squidLog), coluna requestMethod igual a NONE (conforme Apêndice B linha 105) que sinaliza a falha numa requisição.

FIGURA 4.13: Diagrama da Tabela relaRede



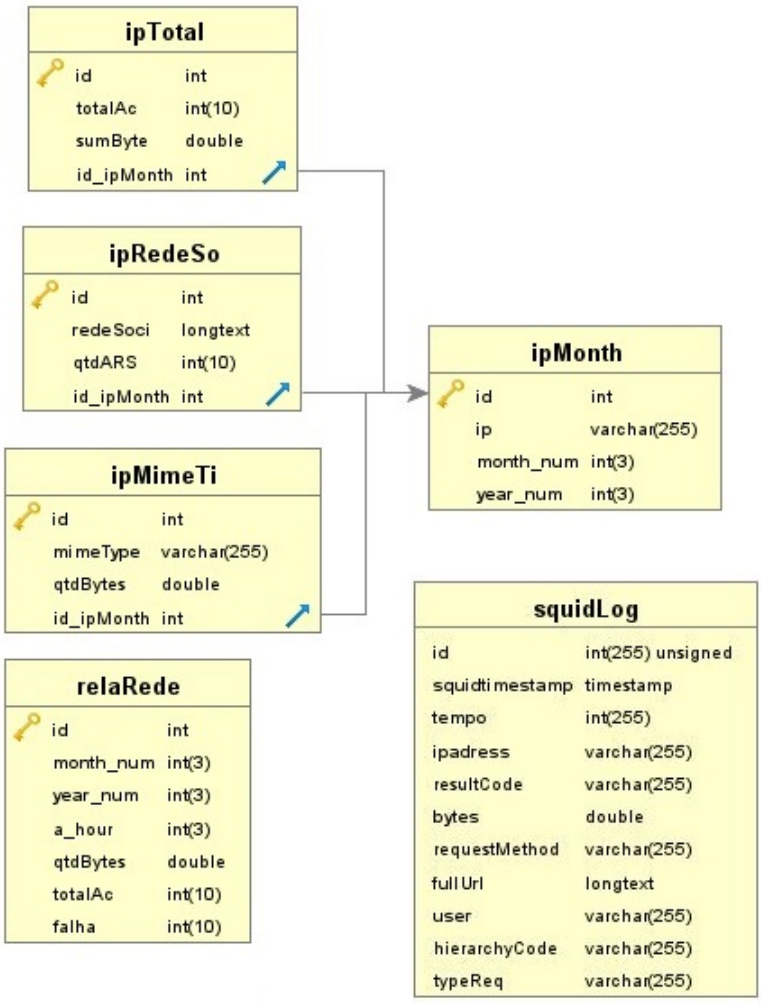
4.3.6 Resultado da Estruturação das Tabelas

Com as tabelas já definidas, criou-se a relação entre elas através das chaves primarias e estrangeiras. O resultado diagrama de relações do banco de dados pode ser visto na figura 4.14.

O funcionamento completo do banco se inicia na rotina de inserção de dados de *log* do Squid para a tabela squidLog, em seguida entra a rotina de inserção de dados nas demais tabelas (ipMonth, ipTotal, ipRedeSo, ipMimeTi e relaRede) por meio de consultas na tabela squidLog. Finalmente, tem-se as *queries* feitas no banco através do aplicativo de celular.

A razão pela qual foi decidido não inserir o *log* diretamente para nas tabelas relacionadas é que em uma futura alteração de relatórios, seria necessário alterar o programa de inserção.

FIGURA 4.14: Diagrama da Estrutura Completa do Banco de Dados MySQL



4.4 RESULTADOS DO APLICATIVO CRIADO

Após utilizar os aprendizados adquiridos na fundamentação teórica, e após desenvolver o que está relatado no capitulo anterior, teve-se como resultado o aplicativo desejado, com as telas principais mostradas a seguir.

A primeira tela do aplicativo é a lista de usuários que pode ser vista na figura 4.15, o resultado foi uma tela simples e funcional, que mostra todos os IP's dos usuários da rede.

Já ao selecionar um IP de usuário, a próxima tela apresentada é o relatório de dados da rede, mostrando a quantidade de acesso e o tráfego de dados, esta tela pode ser vista na figura 4.16.

É importante mostrar que foi utilizado um *fragment DatePicker* para que o

FIGURA 4.15: Tela 1 APP - Lista Usuários

gestor que utilizará o aplicativo possa filtrar os dados por uma data desejada, a figura 4.17 mostra essa função.

Ao tocar no botão avançar no canto superior direito da tela, é apresentado a figura 4.18, que mostra o relatório de uso de redes sociais, que apresentam dados sobre o use de Youtube, Facebook e Twitter.

Avançando de tela mais uma vez, tem-se o relatório de tipos de dados, que mostra um gráfico com todos os tipos de dados que foram acessados pelo usuário, a figura 4.19 mostra este relatório.

A tela que apresenta dados gerais sobre a rede, unificando todos os usuários é mostrada na figura 4.20, apresentando os gráficos de quantidades de acessos por hora, quantidade de GB por hora, e também a quantidade de falhas de conexão por hora.

Já as figuras 4.21 e figura 4.22 dizem respeito á tela de busca por URL acessada, na figura 4.21 é possível digitar uma URL ou um fragmento URL, após a resposta da base de dados, tem-se o que é mostrado na figura 4.22, que apresenta todos os acesso à URL em questão.

FIGURA 4.16: Tela 2 APP - Relatório de dados

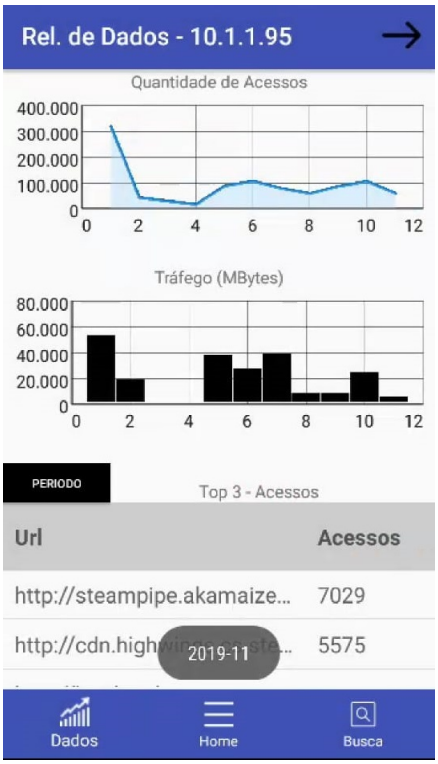


FIGURA 4.17: Tela 3 APP - Fragment (DatePick)

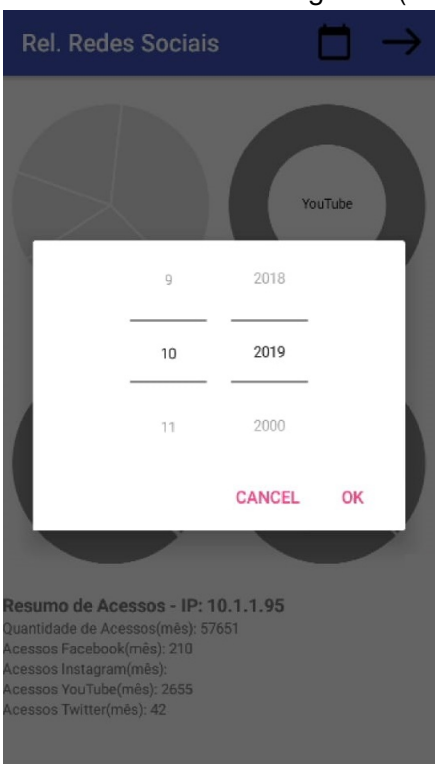


FIGURA 4.18: Tela 4 APP - Relatório de redes sociais

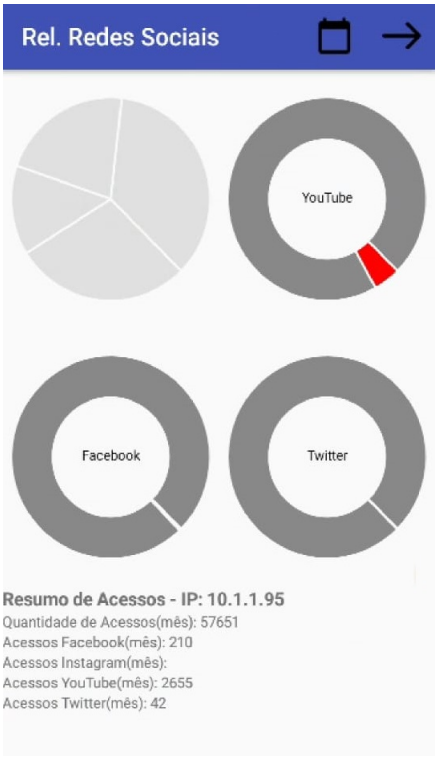


FIGURA 4.19: Tela 5 APP - Relatório de tipos de dados

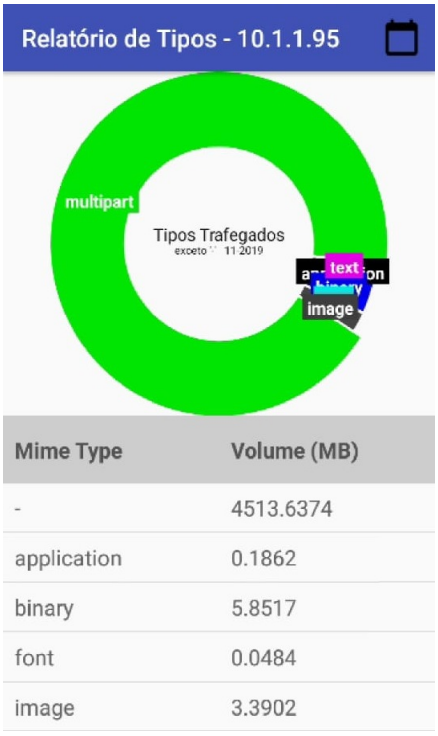


FIGURA 4.20: Tela 6 APP - Relatório de dados da rede

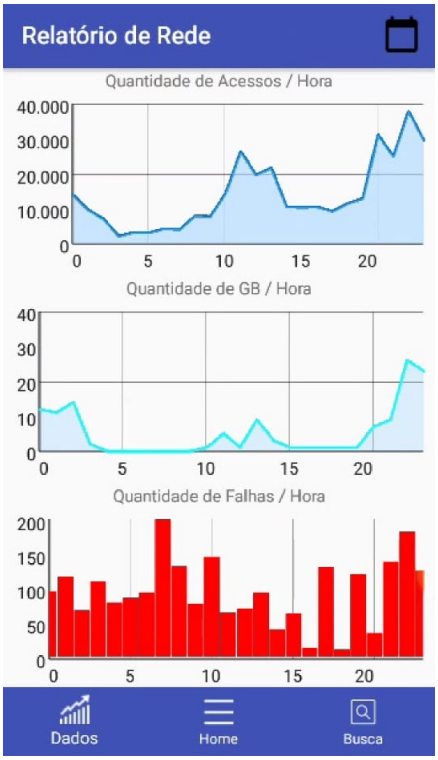


FIGURA 4.21: Tela 7 APP - Tela de busca de URL

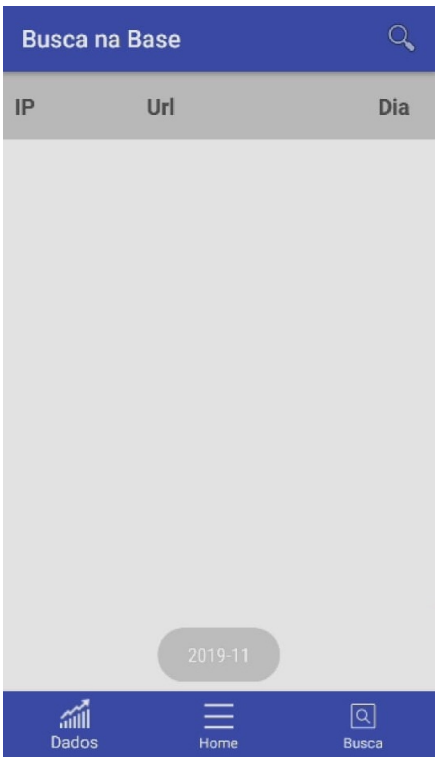
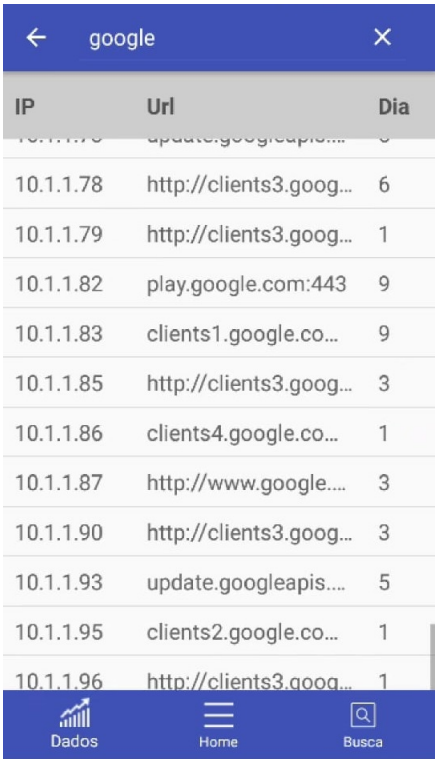


FIGURA 4.22: Tela 8 APP - Resultado busca URL



5 CONCLUSÃO

A Internet é uma ferramenta que facilita o acesso à informação e que tem revolucionado a maneira que as empresas se relacionam com seu público. Porém, se usada da maneira incorreta pode prejudicar o desempenho dos funcionários destas empresas. Neste projeto foi apresentado um dispositivo embarcado capaz de monitorar o tráfego de Internet de uma pequena empresa, permitindo a visualização de relatórios em tempo real utilizando smartphones ou tablets.

A configuração do servidor de *proxy* Squid foi bem sucedida, utilizando-se do que foi apresentado no capítulo de desenvolvimento, e cujo resultado foi mostrado através de imagens comprovando que todo o fluxo de rede passa única e exclusivamente pelo sistema embarcado projetado. Também foram realizadas as configurações de itens importantes para o funcionamento do projeto como o DNS, o DHCP e o *Gateway* padrão.

Os arquivos de *log* do projeto de *proxy* embarcado foram gerados com sucesso, mostrando toda e qualquer informação referente às requisições de acesso recebidas. Dentro deste *log* foi possível verificar que todas as informações necessárias estavam presentes, como por exemplo a data de acesso, a URL completa, o usuário que fez a requisição, bem como muitas outras informações deste acesso

A estruturação do banco de dados foi necessária, pois a velocidade de acesso é maior quando a base utilizada é dividida conforme a necessidade, após ter sido escolhido os dados que seriam apresentados, a base de dados foi projetada para que estes mesmo dados pudessem ser acessados de maneira rápida. No entanto, pesquisas como a busca por texto nas URLs e a seleção dos três sites mais acessados precisaram ser feitas diretamente na base, gerando assim um tempo maior de resposta agravado pela falta e memória RAM do Raspberry Pi. Um fator que atenuou esse problema é o *cache* do MySql.

Com todos os dados separados conforme a necessidade de cada relatório, foi desenvolvido então o aplicativo baseado na plataforma Android, todas as telas e seus conteúdos foram mostrados no capítulo de resultados. Foram adicionados também rotinas de carregamento (*loading*) nas telas que apresentam *queries* diretamente na base. Agora os gestores de pequenas empresas poderão ter na palma da mão todos

os dados referentes aos acessos executados por seus funcionários no ambiente de trabalho.

Este mesmo gestor poderá ter acesso aos relatórios baseados pelos IP's dos usuários, de forma a analisar o comportamento referente ao uso total de internet, o tipo de arquivo que eles acessam, quantidade de acessos às redes sociais, e um relatório geral da rede.

Por fim, o resultado total do projeto foi satisfatório, pois após seguir a metodologia definida no desenvolvimento, todos os objetivos foram cumpridos, e todos os itens envolvidos no projeto funcionaram e comunicaram-se entre si para que o sistema de monitoramento da rede local através do uso de um servidor *proxy* embarcado fosse concluído de forma funcional.

REFERÊNCIAS

ANDROID. *Android*. 2017. <https://developer.android.com/studio/command-line/index.html>. Acessado em: 2018-17-03. Citado na página 39.

BERGHER. *Celular Android*. 2017. <https://www.zoom.com.br/celular/deumzoom/o-que-e-um-celular-android>. Acessado em: 2018-17-03. Citado na página 39.

CHAFFEY, D. *Smartinsights*. 2019. <https://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/>. Acessado em: 2019-18-11. Citado na página 17.

CISCO. *IP Addressing: DHCP Configuration Guide*. [S.l.], 2018. Disponível em: <https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipaddr_dhcp/configuration/15-mt/dhcp-15-mt-book/config-dhcp-client.html>. Acesso em: 24.03.2018. Citado na página 32.

COMER, D. E. *Redes de Computador e Internet 4ª Edição*. New Jersey: Bookman, 2007. Citado 6 vezes nas páginas 31, 32, 33, 36, 38 e 45.

CORDEIRO. *banco de dados no android*. 2016. <http://www.techtudo.com.br/artigos/noticia/2012/ 04/o-que-e-e-como-usar-o-mysql.htmlhttps://www.androidpro.com.br/ usando-banco-de-dados-externo-no-android/>. Acessado em: 2018-10-03. Citado na página 41.

DBVISUALIZER. *DbVisualizer 10.0: User Guide*. Estados Unidos: [s.n.], 2019. Citado na página 45.

DEBIAN. *Debian*. [S.l.], 2018. Disponível em: <https://www.debian.org>. Acesso em: 18.03.2018. Citado na página 29.

FOUNDATION, R. *Raspberry Foundation*. [S.l.], 2018. Disponível em: <https://www.raspberrypi.org/>. Acesso em: 18.03.2018. Citado 2 vezes nas páginas 23 e 24.

GLOBALWEBINDEX. *Time spend on social media*. 2019. <https://www.bbc.com/portuguese/geral-49602237>. Acessado em: 2019-18-11. Citado 2 vezes nas páginas 17 e 18.

GROFF, J. R.; WEINBERG, P. N. *SQL: The Complete Reference, Second Edition*. Estados Unidos: McGraw-Hill/Osborne, 2002. Citado 6 vezes nas páginas 10, 46, 47, 48, 49 e 50.

JUNIOR, C. P. *DNS – Domain Name System*. [S.l.], 2018. Disponível em: <http://www.lsi.usp.br/~penasio/cursos/adm_redes/aula_DNS.pdf>. Acesso em: 08.04.2018. Citado 2 vezes nas páginas 33 e 36.

JUNIOR, G. G. *Modelo de Servidor Web com Quatro Módulos de Atendimento de Requisições*. [S.l.], 2008. Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-02072008-144125/publico/geraldo.pdf>. Acesso em: 22.04.2018. Citado 3 vezes nas páginas 36, 37 e 38.

MIDORIKAWA, E. T. *Introdução aos Sistemas de Computação*. [S.l.], 2018. Disponível em: <http://www.lsi.usp.br/~volnys/courses/linux/pdf-col/sist-col.pdf>. Acesso em: 18.03.2018. Citado na página 28.

NEGUS, C.; BRESNAHAN, C. *Linux A Bíblia*. Rio de Janeiro, Brasil: Alta Books, 2014. Citado 3 vezes nas páginas 29, 45 e 51.

ORANGEPI. *OrangePi*. [S.l.], 2018. Disponível em: <http://www.orangepi.org/>. Acesso em: 18.03.2018. Citado 2 vezes nas páginas 24 e 25.

PISA. *Mysql*. 2016. <http://www.techtudo.com.br/artigos/noticia/2012/04/ o-que-e-e-como-usar-o-mysql.html>. Acessado em: 2019-17-03. Citado na página 40.

PLANALTO. *Lei geral de Proteção de Dados Pessoais*. 2018. <http://www.planalto.gov.br/ccivil$_0 3/_a to 2015 - 2018/2018/lei/L13709.htm >$ $.Acessadoem : 2019 - 10 - 07.Citadonap\acute{a}gina 19.$

RAKOSKI, L. *Redes sociais no trabalho*. 2018. <https://campoeste.org.br/redes-sociais-no-trabalho/>. Acessado em: 2019-18-11. Citado 2 vezes nas páginas 18 e 19.

RASPBIAN. *Raspibian*. [S.l.], 2018. Disponível em: <https://www.raspbian.org/>. Acesso em: 22.03.2018. Citado 5 vezes nas páginas 10, 30, 44, 45 e 50.

SAINI, K. *Squid Proxy Server 3.1 Beginner's Guide*. United Kingdom: Packt, 2011. Citado na página 31.

SQUIDORG. *Squid-Cache*. [S.l.], 2018. Disponível em: <http://www.squid-cache.org/>. Acesso em: 24.03.2018. Citado na página 31.

TANENBAUM, A. S. *Sistemas Operacionais Modernos 3ª Edição*. São Paulo: Pearson, 2010. Citado 7 vezes nas páginas 25, 26, 27, 28, 29, 33 e 36.

TEIXEIRA, M. A. *Sistemas Operacionais*. [S.l.], 2018. Disponível em: <http://ctd.ifsp.edu.br/~marcio.andrey/images/estrutura_so-ifsp-catanduva.pdf>. Acesso em: 18.03.2018. Citado na página 27.

UDACITY. *Cursos de android online*. 2018. <https://classroom.udacity.com/courses/ud851/lessons/93affc67-3f0b-4f9b-b3a4-a7a26f241a86/concepts/ 98b2f686-aca1-4547-9632-31e84ccca355>. Acessado em: 2018-05-05. Citado na página 39.

UEYAMA, J. *Redes de Computadores*. [S.l.], 2014. Disponível em: <http://wiki.icmc.usp.br/images/7/7e/Rc04-dns.pdf>. Acesso em: 08.04.2018. Citado 4 vezes nas páginas 33, 34, 35 e 36.

APÊNDICE A – LEITURA E INSERÇÃO DO LOG

A.1 ARQUIVO LOGMINER.JAVA – CLASSE DE LEITURA DO *LOG* FONTE: AUTORIA PRÓPRIA

```
1  /*
2   * To change this license header, choose License Headers in Project
        Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package controler;
7
8  import java.io.File;
9
10 public class LogMiner {
11
12     /**
13      * @param args the command line arguments
14      */
15     public static void main(String[] args) throws Exception {
16
17         File file = new File(args[0]);
18         ReadFile.read(file);
19
20     }
21
22 }
```

A.2 ARQUIVO MYSQLCON.JAVA – INSERÇÃO DE LINHAS FONTE: DOS AUTORES

```
1  package controler;
2
3  import java.sql.*;
4  import java.util.concurrent.TimeUnit;
5  import java.util.logging.Level;
6  import java.util.logging.Logger;
7  import model.Logline;
```

```
8
9
10 public class MysqlCon {
11     private static String user = "squid";
12     private static String pwd = "198@gorA";
13     private static String dbName = "logSquid";
14     private static int port = 3306;
15
16     public static Connection conecta () throws SQLException,
           ClassNotFoundException{
17
18         Class.forName("com.mysql.cj.jdbc.Driver");
19         return DriverManager.getConnection("jdbc:mysql://wpad:"+port+"/"+
               dbName,user,pwd);
20         //Teste
21         //return DriverManager.getConnection("jdbc:mysql://mysql:"+port
               +"/"+dbName,"root",pwd);
22     }
23
24     public static void CloseCon (Connection con) throws SQLException{
25         con.close();
26     }
27
28     public static int insertLogRow(Logline line, Connection con) throws
           SQLException{
29         try {
30             Class.forName("com.mysql.cj.jdbc.Driver");
31
32
33             PreparedStatement stmt = con.prepareStatement(" INSERT INTO
                   squidLog\n" +
34             "(squidtimestamp, tempo, ipadress, resultCode, bytes,
                   requestMethod, fullUrl, user, hierarchyCode, typeReq)\n" +
35             "VALUES\n" +
36                "(FROM_UNIXTIME("+line.getTimeStamp()+"),?,?,?,?,?,?,?,?,?)
                       ;");
37             stmt.setLong(1, line.getTempo());
38             stmt.setString(2, line.getIpadress());
39             stmt.setString(3, line.getResultCode());
```

```
40              stmt.setLong(4, line.getBytes());
41              stmt.setString(5, line.getRequestMethod());
42              stmt.setString(6, line.getFullUrl());
43              stmt.setString(7, line.getUser());
44              stmt.setString(8, line.getHierarchyCode());
45              stmt.setString(9, line.getTypeReq());
46
47              //System.out.println(stmt.toString());
48
49
50                              /*+line.getTempo()+", '"+line.getIpadress()
                                    +"', '"+line.getResultCode()
51                      +"',"+line.getBytes()+",'"+line.getRequestMethod()
                            +"','"+line.getFullUrl()+"','"+line.getUser()
                            +"','"+line.getHierarchyCode()
52                      +"','"+line.getTypeReq()+"');");*/
53              int rs= stmt.executeUpdate();
54              //con.close();
55              return rs;
56
57          } catch (ClassNotFoundException ex) {
58              Logger.getLogger(MysqlCon.class.getName()).log(Level.SEVERE,
                    null, ex);
59          }
60          return -1;
61
62      }
63
64      public static int insertError (Exception e, String line, Connection
            con) throws SQLException, InterruptedException{
65          try {
66              Class.forName("com.mysql.cj.jdbc.Driver");
67
68              PreparedStatement stmt = con.prepareStatement(" INSERT INTO
                    Erros\n" +
69              "(Linetxt, Error)\n" +
70              "VALUES (?,?);");
71              stmt.setString(1, line);
72              stmt.setString(2, e.getMessage().toString());
```

```
73              //System.out.println(stmt.toString());
74              int rs=stmt.executeUpdate();
75
76              //con.close();
77              return rs;
78
79           } catch (ClassNotFoundException ex) {
80              Logger.getLogger(MysqlCon.class.getName()).log(Level.SEVERE,
                    null, ex);
81          }
82          catch (SQLSyntaxErrorException ex){
83              System.out.println("Erro erro insert");
84              System.out.println(e.getMessage().toString());
85              System.out.println(line);
86              TimeUnit.MINUTES.sleep(1);
87          }
88
89          return -1;
90
91      }
92
93      public static int checkBeforeInsert (long FirstTimeStamp, long
            lastTimeStamp, Connection con){
94          try {
95              //Class.forName("com.mysql.cj.jdbc.Driver");
96              //Connection con=DriverManager.getConnection("jdbc:mysql://
                    mysql:"+port+"/"+dbName,user,pwd);
97              Statement stmt=con.createStatement();
98              ResultSet result=stmt.executeQuery("select squidtimestamp from
                     squidLog where"
99                      + " squidtimestamp >= FROM_UNIXTIME("+FirstTimeStamp+"
                            ) and squidtimestamp <= FROM_UNIXTIME("+
                            lastTimeStamp+");");
100
101             result.last();
102             return result.getRow();
103
104         } catch (SQLException ex) {
105             Logger.getLogger(MysqlCon.class.getName()).log(Level.SEVERE,
```

```
                null, ex);
106         }
107
108         return -1;
109     }
110
111     public static int deleteRows (long FirstTimeStamp, long lastTimeStamp,
            Connection con){
112         try {
113             //Class.forName("com.mysql.cj.jdbc.Driver");
114             //Connection con=DriverManager.getConnection("jdbc:mysql://
                    mysql:"+port+"/"+dbName,user,pwd);
115             Statement stmt=con.createStatement();
116             int rs=stmt.executeUpdate("delete from squidLog where"
117                     + " squidtimestamp >= FROM_UNIXTIME("+FirstTimeStamp+"
                            ) and squidtimestamp <= FROM_UNIXTIME("+
                            lastTimeStamp+");");
118
119             return rs;
120
121         } catch (SQLException ex) {
122             Logger.getLogger(MysqlCon.class.getName()).log(Level.SEVERE,
                    null, ex);
123         }
124
125         return -1;
126     }
127
128 }
```

A.3 ARQUIVO READFILE.JAVA – LEITURA ESTRUTURADA FONTE: DOS AUTORES

```
1 package controler;
2
3 // Java Program to illustrate reading from FileReader
4 // using BufferedReader
5 import java.io.*;
6 import static java.lang.System.exit;
7 import java.sql.Connection;
```

```
8  import java.sql.SQLException;
9  import java.util.concurrent.TimeUnit;
10 import static jdk.nashorn.internal.objects.NativeMath.abs;
11 import model.Logline;
12 public class ReadFile
13 {
14 public static void read (File file) throws Exception
15     {
16     // We need to provide file path as the parameter:
17     // double backquote is to avoid compiler interpret words
18     // like \test as \t (ie. as a escape sequence)
19
20     BufferedReader br = new BufferedReader(new FileReader(file));
21
22     int erroCount = 0;
23     int okCount = 0;
24     int dif = 0;
25     int telerancy = 10;
26     String st;
27
28     Connection con = MysqlCon.conecta();
29
30     while ((st = br.readLine()) != null){
31         String linetxt = StringCuter.removeTwoSpaces(st);
32         // System.out.println(StringCuter.TimeStamp(StringCuter.
               removeTwoSpaces(st)).toString());
33         //System.out.println(StringCuter.Tempo(StringCuter.removeTwoSpaces(
               st)));
34         //System.out.println(StringCuter.typeReq(StringCuter.
               removeTwoSpaces(st)));
35             try {
36                 Logline logrow = null;
37                 logrow = StringCuter.loadline(linetxt);
38                 if (okCount == 0) {
39                     dif = (MysqlCon.checkBeforeInsert(ReadFile.firstLine(
                           file), ReadFile.lastLine(file),con)-ReadFile.
                           countLines(file));
40                     if (dif>telerancy || telerancy<(-1*dif)) {
41                         if (MysqlCon.checkBeforeInsert(ReadFile.firstLine(
```

```
                                file), ReadFile.lastLine(file),con)>0){
42                              System.out.println("Check count wrong
                                    deleteing dbRows:"+MysqlCon.
                                    checkBeforeInsert(ReadFile.firstLine(file),
                                     ReadFile.lastLine(file),con)+" fileRows: "
                                    +ReadFile.countLines(file) +" dif "+dif);
43                              MysqlCon.deleteRows(ReadFile.firstLine(file),
                                    ReadFile.lastLine(file),con);
44                          }
45                      }
46                      else {
47                          System.out.println("Check count rows ok dbRows:"+
                                MysqlCon.checkBeforeInsert(ReadFile.firstLine(
                                file), ReadFile.lastLine(file),con)+" fileRows:
                                 "+ReadFile.countLines(file) +" dif "+dif);
48                          break;
49                      }
50                      //TimeUnit.MINUTES.sleep(1);
51                  }
52                  MysqlCon.insertLogRow(logrow,con);
53                  okCount += 1;
54              }
55              catch (Exception e){
56                  MysqlCon.insertError(e, st,con);
57                  erroCount += 1;
58              }
59
60          }
61          System.out.println("Insert ok: "+okCount);
62          System.out.println("Insert errors: "+erroCount);
63          MysqlCon.CloseCon(con);
64      }
65
66  public static long lastLine (File file) throws FileNotFoundException,
        IOException, SQLException {
67          BufferedReader br = new BufferedReader(new FileReader(file));
68          Logline logrow = null;
69          String st;
70
```

```
71          while ((st = br.readLine()) != null){
72              String linetxt = StringCuter.removeTwoSpaces(st);
73              try {
74                  logrow = StringCuter.loadline(linetxt);
75              }
76              catch (Exception e){
77              }
78          }
79          return logrow.getTimeStamp();
80      }
81
82  public static long firstLine (File file) throws FileNotFoundException,
        IOException, SQLException {
83          BufferedReader br = new BufferedReader(new FileReader(file));
84          Logline logrow = null;
85          String st;
86
87          while ((st = br.readLine()) != null){
88              String linetxt = StringCuter.removeTwoSpaces(st);
89              try {
90                  logrow = StringCuter.loadline(linetxt);
91                  return logrow.getTimeStamp();
92              }
93              catch (Exception e){
94              }
95          }
96          return logrow.getTimeStamp();
97      }
98
99  public static int countLines (File file) throws FileNotFoundException,
        IOException{
100     /*BufferedReader br = new BufferedReader(new FileReader(file));
101     int count = 0;
102     String st = null;
103     while ((st = br.readLine()) != null){
104             count = count +1;
105         }
106     return count;
107     }*/
```

```
108     int result = 0;
109
110     FileReader       input = new FileReader(file);
111     LineNumberReader count = new LineNumberReader(input);
112     {
113         while (count.skip(Long.MAX_VALUE) > 0)
114         {
115       // Loop just in case the file is > Long.MAX_VALUE or skip() decides
            to not read the entire file
116         }
117             result = count.getLineNumber() + 1;
                                                    // +1 because line index
                    starts at 0
118         }
119         return result;
120     }
121 }
```

A.4 ARQUIVO STRINGCUTER.JAVA – QUEBRA DAS STRINGS FONTE: DOS AUTORES

```
1
2  package controler;
3
4  import java.sql.Timestamp;
5  import model.Logline;
6
7  public class StringCuter {
8      public static String removeTwoSpaces (String text){
9          String tempTxt = new String();
10         tempTxt = text.replaceAll("  ", " ");
11         while (tempTxt.indexOf("  ")!=-1){
12            tempTxt = tempTxt.replaceAll("  ", " ");
13         }
14         return tempTxt;
15     }
16
17     public static long timeStamp (String text){
18
19         String temptxt = new String();
```

```
20          long number = 0;
21          //temptxt =   text.substring(text.indexOf(" ")).trim();
22          temptxt = text.substring(0, text.indexOf(" ")).trim();
23          //System.out.println(temptxt);
24          return (long)Double.parseDouble(temptxt);
25          //number = (long)Double.parseDouble(temptxt);
26          //return new Timestamp(number);
27      }
28
29      public static long tempo (String text){
30          String temptxt = new String();
31          long number = 0;
32          //temptxt =   text.substring(text.indexOf(" ")).trim();
33          temptxt = text.substring(text.indexOf(" ",text.indexOf(" "))).trim
                ();
34          temptxt = temptxt.substring(0,temptxt.indexOf(" ")).trim();
35
36          return (long)Double.parseDouble(temptxt);
37      }
38
39      public static String ipadress (String text){
40          String temptxt = new String();
41          temptxt = text.substring(text.indexOf(" ",text.indexOf(" "))).trim
                ();
42          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
43          temptxt = temptxt.substring(0,temptxt.indexOf(" "));
44          return temptxt;
45      }
46
47      public static String resultCode (String text){
48          String temptxt = new String();
49          temptxt = text.substring(text.indexOf(" ",text.indexOf(" "))).trim
                ();
50          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
51          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
52          temptxt = temptxt.substring(0,temptxt.indexOf(" "));
53          return temptxt;
54      }
55
```

```
56      public static long bytes (String text){
57          String temptxt = new String();
58          double numero = 0;
59
60          temptxt = text.substring(text.indexOf(" ",text.indexOf(" "))).trim
                ();
61          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
62          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
63          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
64          temptxt = temptxt.substring(0,temptxt.indexOf(" "));
65
66          numero = Double.parseDouble(temptxt);
67
68          return (long) numero;
69      }
70
71      public static String requestMethod (String text) {
72          String temptxt = new String();
73          temptxt = text.substring(text.indexOf(" ",text.indexOf(" "))).trim
                ();
74          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
75          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
76          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
77          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
78          temptxt = temptxt.substring(0,temptxt.indexOf(" "));
79          return temptxt;
80      }
81
82      public static String fullUrl (String text) {
83         String temptxt = new String();
84          temptxt = text.substring(text.indexOf(" ",text.indexOf(" "))).trim
                ();
85          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
86          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
87          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
88          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
89          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
90          temptxt = temptxt.substring(0,temptxt.indexOf(" "));
91          return temptxt;
```

```
92      }
93
94      public static String user (String text) {
95         String temptxt = new String();
96          temptxt = text.substring(text.indexOf(" ",text.indexOf(" "))).trim
               ();
97          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
98          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
99          temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
100         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
101         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
102         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
103         temptxt = temptxt.substring(0,temptxt.indexOf(" "));
104         return temptxt;
105     }
106
107
108     public static String hierarchyCode (String text) {
109        String temptxt = new String();
110         temptxt = text.substring(text.indexOf(" ",text.indexOf(" "))).trim
               ();
111         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
112         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
113         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
114         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
115         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
116         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
117         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
118         temptxt = temptxt.substring(0,temptxt.indexOf(" "));
119         return temptxt;
120     }
121
122     public static String typeReq (String text) {
123        String temptxt = new String();
124         temptxt = text.substring(text.indexOf(" ",text.indexOf(" "))).trim
               ();
125         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
126         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
127         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
```

```
128         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
129         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
130         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
131         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
132         temptxt = temptxt.substring(temptxt.indexOf(" ")).trim();
133         //temptxt = temptxt.substring(0,temptxt.indexOf(" "));
134         return temptxt;
135     }
136
137     public static Logline loadline (String line){
138         String line2 = StringCuter.removeTwoSpaces(line);
139         Logline logrow = new Logline(StringCuter.timeStamp(line2),
                StringCuter.tempo(line2),StringCuter.ipadress(line2),
140         StringCuter.resultCode(line2),StringCuter.bytes(line2),StringCuter
                .requestMethod(line2),
141         StringCuter.fullUrl(line2),StringCuter.user(line2),StringCuter.
                hierarchyCode(line2),StringCuter.typeReq(line2));
142
143         return logrow;
144
145     }
146 }
```

A.5 ARQUIVO LOGLINE.JAVA – CLASSE LINHA FONTE: DOS AUTORES

```
1  /*
2   * To change this license header, choose License Headers in Project
        Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package model;
7
8  import java.sql.Timestamp;
9
10
11 public class Logline {
12     private long timeStamp;
13     private long tempo;
14     private String ipadress;
```

```
15      private String resultCode;
16      private long bytes;
17      private String requestMethod;
18      private String fullUrl;
19      private String user;
20      private String hierarchyCode;
21      private String typeReq;
22
23      public Logline (long tS, long tempol, String ipadressl,
24      String resultCodel, long bytesl, String requestMethodl, String
            fullUrll, String userl,
25      String hierarchyCodel, String typeReql){
26          this.setTimeStamp(tS);
27          this.setTempo(tempol);
28          this.setIpadress(ipadressl);
29          this.setResultCode(resultCodel);
30          this.setBytes(bytesl);
31          this.setRequestMethod(requestMethodl);
32          this.setFullUrl(fullUrll);
33          this.setUser(userl);
34          this.setHierarchyCode(hierarchyCodel);
35          this.setTypeReq(typeReql);
36      }
37
38      public long getTimeStamp() {
39          return timeStamp;
40      }
41
42      public void setTimeStamp(long timeStamp) {
43          this.timeStamp = timeStamp;
44      }
45
46      public long getTempo() {
47          return tempo;
48      }
49
50      public void setTempo(long tempo) {
51          this.tempo = tempo;
52      }
```

```
53
54      public String getIpadress() {
55          return ipadress;
56      }
57
58      public void setIpadress(String ipadress) {
59          this.ipadress = ipadress;
60      }
61
62      public String getResultCode() {
63          return resultCode;
64      }
65
66      public void setResultCode(String resultCode) {
67          this.resultCode = resultCode;
68      }
69
70      public long getBytes() {
71          return bytes;
72      }
73
74      public void setBytes(long bytes) {
75          this.bytes = bytes;
76      }
77
78      public String getRequestMethod() {
79          return requestMethod;
80      }
81
82      public void setRequestMethod(String requestMethod) {
83          this.requestMethod = requestMethod;
84      }
85
86      public String getFullUrl() {
87          return fullUrl;
88      }
89
90      public void setFullUrl(String fullUrl) {
91          this.fullUrl = fullUrl;
```

```
92      }
93
94      public String getUser() {
95          return user;
96      }
97
98      public void setUser(String user) {
99          this.user = user;
100     }
101
102     public String getHierarchyCode() {
103         return hierarchyCode;
104     }
105
106     public void setHierarchyCode(String hierarchyCode) {
107         this.hierarchyCode = hierarchyCode;
108     }
109
110     public String getTypeReq() {
111         return typeReq;
112     }
113
114     public void setTypeReq(String typeReq) {
115         this.typeReq = typeReq;
116     }
117
118
119 }
```

APÊNDICE B – ATUALIZAÇÃO DAS TABELAS

B.1 ARQUIVO SCRIPTTCC.SQL – ATUALIZAÇÃO DAS TABELAS COM RELAÇÕES FONTE: DOS AUTORES

```
1
2  use squidBI;
3
4  start transaction;
5
6  DELETE FROM squidBI.ipTotal WHERE id_ipMonth in (SELECT id FROM squidBI.
       ipMonth WHERE month_num in(MONTH(CURDATE()), MONTH(CURDATE())-1, MONTH(
       CURDATE())-2));
7
8  DELETE FROM squidBI.ipRedeSo WHERE id_ipMonth in (SELECT id FROM squidBI.
       ipMonth WHERE month_num in(MONTH(CURDATE()), MONTH(CURDATE())-1, MONTH(
       CURDATE())-2));
9
10 DELETE FROM squidBI.ipMimeTi WHERE id_ipMonth in (SELECT id FROM squidBI.
       ipMonth WHERE month_num in(MONTH(CURDATE()), MONTH(CURDATE())-1, MONTH(
       CURDATE())-2));
11
12 DELETE FROM squidBI.ipMonth WHERE month_num in (MONTH(CURDATE()), MONTH(
       CURDATE())-1, MONTH(CURDATE())-2);
13
14 DELETE FROM squidBI.relaRede WHERE month_num in (MONTH(CURDATE()), MONTH(
       CURDATE())-1, MONTH(CURDATE())-2);
15
16
17
18
19
20 INSERT INTO squidBI.ipMonth (ip, month_num, year_num)
21 select ipadress, month(squidtimestamp), year(squidtimestamp) from logSquid
       .squidLog
22 WHERE month(squidtimestamp) in (MONTH(CURDATE()), MONTH(CURDATE())-1,
       MONTH(CURDATE())-2)
23 group by ipadress, month(squidtimestamp), year(squidtimestamp);
```

```
24
25
26
27 INSERT INTO squidBI.ipTotal (totalAc, sumByte, id_ipMonth)
28 select log.fullUrl, log.bytes, ipMonth.id from ipMonth, (select count(
       fullUrl) fullUrl, sum(bytes) bytes, year(squidtimestamp) dtYear,
       ipadress, month(squidtimestamp) dtMonth from logSquid.squidLog
29 where month(squidtimestamp) in (MONTH(CURDATE()), MONTH(CURDATE())-1,
       MONTH(CURDATE())-2)
30 group by ipadress, month(squidtimestamp), year(squidtimestamp)) log
31 where year_num = log.dtYear
32 and month_num  = log.dtMonth
33 and ip = log.ipadress collate utf8mb4_unicode_ci
34 group by ipMonth.id;
35
36
37 INSERT INTO squidBI.ipRedeSo (redeSoci, qtdARS, id_ipMonth)
38 select SUBSTRING_INDEX(fullUrl, '/', 3), count(SUBSTRING_INDEX(fullUrl, '/
       ', 3)), ipMonth.id from ipMonth, logSquid.squidLog
39 where month(squidtimestamp) in (MONTH(CURDATE()), MONTH(CURDATE())-1,
       MONTH(CURDATE())-2)
40 and SUBSTRING_INDEX(fullUrl, '/', 3) LIKE "%twitter%" and
41 year_num = year(squidtimestamp) and
42 month_num  = month(squidtimestamp) and
43 ip = ipadress collate utf8mb4_unicode_ci
44 group by ipMonth.id,
45 SUBSTRING_INDEX(fullUrl, '/', 3) LIKE "%twitter%" ;
46
47 INSERT INTO squidBI.ipRedeSo (redeSoci, qtdARS, id_ipMonth)
48 select SUBSTRING_INDEX(fullUrl, '/', 3), count(SUBSTRING_INDEX(fullUrl, '/
       ', 3)), ipMonth.id from ipMonth, logSquid.squidLog
49 where month(squidtimestamp) in (MONTH(CURDATE()), MONTH(CURDATE())-1,
       MONTH(CURDATE())-2)
50 and SUBSTRING_INDEX(fullUrl, '/', 3) LIKE "%facebook%" and
51 year_num = year(squidtimestamp) and
52 month_num  = month(squidtimestamp) and
53 ip = ipadress collate utf8mb4_unicode_ci
54 group by ipMonth.id,
55 SUBSTRING_INDEX(fullUrl, '/', 3) LIKE "%facebook%" ;
```

```
56
57 INSERT INTO squidBI.ipRedeSo (redeSoci, qtdARS, id_ipMonth)
58 select SUBSTRING_INDEX(fullUrl, '/', 3), count(SUBSTRING_INDEX(fullUrl, '/
       ', 3)), ipMonth.id from ipMonth, logSquid.squidLog
59 where month(squidtimestamp) in (MONTH(CURDATE()), MONTH(CURDATE())-1,
       MONTH(CURDATE())-2)
60 and SUBSTRING_INDEX(fullUrl, '/', 3) LIKE "%instagram%" and
61 year_num = year(squidtimestamp) and
62 month_num  = month(squidtimestamp) and
63 ip = ipadress collate utf8mb4_unicode_ci
64 group by ipMonth.id,
65 SUBSTRING_INDEX(fullUrl, '/', 3) LIKE "%instagram%" ;
66
67
68 INSERT INTO squidBI.ipRedeSo (redeSoci, qtdARS, id_ipMonth)
69 select SUBSTRING_INDEX(fullUrl, '/', 3), count(SUBSTRING_INDEX(fullUrl, '/
       ', 3)), ipMonth.id from ipMonth, logSquid.squidLog
70 where month(squidtimestamp) in (MONTH(CURDATE()), MONTH(CURDATE())-1,
       MONTH(CURDATE())-2)
71 and SUBSTRING_INDEX(fullUrl, '/', 3) LIKE "%youtube%" and
72 year_num = year(squidtimestamp) and
73 month_num  = month(squidtimestamp) and
74 ip = ipadress collate utf8mb4_unicode_ci
75 group by ipMonth.id,
76 SUBSTRING_INDEX(fullUrl, '/', 3) LIKE "%youtube%" ;
77
78
79
80
81 INSERT INTO squidBI.ipMimeTi (mimeType, qtdBytes, id_ipMonth)
82 select typeReq, sum(bytes), ipMonth.id from ipMonth, logSquid.squidLog
83 where month(squidtimestamp) in (MONTH(CURDATE()), MONTH(CURDATE())-1,
       MONTH(CURDATE())-2) and
84 year_num = year(squidtimestamp) and
85 month_num  = month(squidtimestamp) and
86 ip = ipadress collate utf8mb4_unicode_ci
87 group by ipMonth.id,
88 typeReq;
89
```

```
90
91
92  INSERT INTO squidBI.ipMimeTi (mimeType, qtdBytes, id_ipMonth)
93  select typeReq, sum(bytes), ipMonth.id from ipMonth, logSquid.squidLog
94  where month(squidtimestamp) in (MONTH(CURDATE()), MONTH(CURDATE())-1,
        MONTH(CURDATE())-2) and
95  year_num = year(squidtimestamp) and
96  month_num  = month(squidtimestamp) and
97  ip = ipadress collate utf8mb4_unicode_ci
98  group by ipMonth.id,
99  typeReq;
100
101
102
103
104 INSERT INTO squidBI.relaRede (month_num, year_num, a_hour, totalAc,
        qtdBytes, falha)
105 select month(squidtimestamp), year(squidtimestamp), hour(squidtimestamp),
        count(fullUrl), sum(bytes), count(if(requestMethod = 'NONE',1,null))
        from logSquid.squidLog
106 where month(squidtimestamp) in (MONTH(CURDATE()), MONTH(CURDATE())-1,
        MONTH(CURDATE())-2)
107 group by month(squidtimestamp), year(squidtimestamp), hour(squidtimestamp)
        ;
108
109
110
111 commit;
```

APÊNDICE C – APLICATIVO

C.1 *MAIN ACTIVITY* DO APLICATIVO FONTE: DOS AUTORES

```
1  package parallelcodes.mysqlapp;
2
3  import android.content.Intent;
4  import android.os.AsyncTask;
5  import android.os.Bundle;
6  import android.os.StrictMode;
7  import android.support.annotation.CallSuper;
8  import android.support.annotation.NonNull;
9  import android.support.design.widget.BottomNavigationView;
10 import android.support.v7.app.AppCompatActivity;
11 import android.view.MenuItem;
12 import android.view.View;
13 import android.widget.ArrayAdapter;
14 import android.widget.Button;
15 import android.widget.ListView;
16 import android.widget.TextView;
17 import android.widget.Toast;
18 import android.widget.ImageButton;
19
20 import android.support.v7.widget.SearchView;
21
22 import java.sql.Connection;
23 import java.sql.DriverManager;
24 import java.sql.ResultSet;
25 import java.sql.ResultSetMetaData;
26 import java.sql.Statement;
27 import java.util.ArrayList;
28 import android.widget.AdapterView;
29
30
31 public class MainActivity extends AppCompatActivity implements
       BottomNavigationView.OnNavigationItemSelectedListener {
32
33     private BottomNavigationView navigationView;
```

```
34
35      private ListView lv;
36      public TextView TV;
37      private Button analiticB, homeB, searchB;
38
39
40
41      ArrayList<String> Ar = new ArrayList<String>();
42
43
44
45      @Override
46      @CallSuper
47      protected void onCreate(Bundle savedInstanceState) {
48          super.onCreate(savedInstanceState);
49          setContentView(R.layout.activity_main);
50          getSupportActionBar().setTitle("Lista de Usu rios");
51          if (android.os.Build.VERSION.SDK_INT > 9) {
52              StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.
                    Builder().permitAll().build();
53              StrictMode.setThreadPolicy(policy);
54          } // Allow networking classes in main
55
56
57          navigationView = (BottomNavigationView) findViewById(R.id.
                navigationView);
58          navigationView.setOnNavigationItemSelectedListener(this);
59          TV = (TextView) findViewById(R.id.textView);
60          lv = (ListView) findViewById(R.id.LvIP);
61
62
63
64          GetData mydata =new GetData();
65          Ar = mydata.doInBackground("","","",1);
66
67
68          ArrayAdapter<String> ArAd = new ArrayAdapter<String>(this, android
                .R.layout.simple_list_item_1, Ar);
69          lv.setAdapter(ArAd);
```

```
70
71          lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
72              @Override
73              public void onItemClick(AdapterView<?> parent, View view, int
                    position, long id) {
74                  // Get the selected item text from ListView
75                  //String selectedItem = (String) parent.getItemAtPosition(
                        position);
76
77                 // TV.setText(Ar.get(position));
78
79                  Intent intent = new Intent(MainActivity.this,
                        UserDataActivity.class);
80                  intent.putExtra("EXTRA_SESSION_ID", Ar.get(position));
81                  startActivity(intent);
82              }
83          });
84
85          TV.setText(mydata.ConnectionResult);
86      }
87
88
89      @Override
90      public boolean onNavigationItemSelected(@NonNull MenuItem item) {
91          switch (item.getItemId()) {
92              case R.id.analiticbut: {
93                  Intent i=new Intent(MainActivity.this, NetworkActivity.
                        class);
94                  startActivity(i);
95                  break;
96              }
97              case R.id.homebut: {
98                  break;
99              }
100             case R.id.searchbut: {
101                 Intent i=new Intent(MainActivity.this, SearchActivity.
                        class);
102                     startActivity(i);
103                 break;
```

```
104             }
105         }
106         return true;
107     }
108
109
110
111     protected String doInBackground(String... params) {
112         return null;
113         }
114
115
116
117
118     }
```

C.2 CLASSE DE CONEXÃO AO BANCO DE DADOS FONTE: DOS AUTORES

```
1  package parallelcodes.mysqlapp;
2
3  import android.os.AsyncTask;
4  import android.util.Log;
5  import android.widget.Toast;
6
7  import java.sql.Connection;
8  import java.sql.DriverManager;
9  import java.sql.ResultSet;
10 import java.sql.ResultSetMetaData;
11 import java.sql.Statement;
12
13 public class DBConnection {
14
15     private static final String url = "jdbc:mysql://10.1.1.249:3306/
           squidBI";
16     private static final String user = "squid";
17     private static final String pass = "198@gorA";
18
19
20     public Connection connectionclass() {
21         Connection con = null;
```

```
22          try {
23
24              Class.forName("com.mysql.jdbc.Driver");
25              con = DriverManager.getConnection(url, user, pass);
26              System.out.println("Databaseection success");
27
28
29          } catch (Exception e) {
30             e.printStackTrace();
31          }
32
33          return con;
34      }
35  }
```

C.3 CLASSE GETDATA - CONSULTA AO BANCO DE DADOS FONTE: DOS AUTORES

```
1  package parallelcodes.mysqlapp;
2
3  import java.sql.Connection;
4  import java.sql.ResultSet;
5  import java.sql.ResultSetMetaData;
6  import java.sql.Statement;
7  import java.util.ArrayList;
8  import java.util.HashMap;
9  import java.util.List;
10 import java.util.Map;
11
12 public class GetData {
13
14
15     String ConnectionResult = "";
16     Boolean isSuccess = false;
17
18     ArrayList<String> Ar1 = new ArrayList<String>();
19     ArrayList<String> PrimaryAr1 = new ArrayList<String>();
20
21     ArrayList<String> ips = new ArrayList<String>();
22     ArrayList<String> dias = new ArrayList<String>();
```

```
23      ArrayList<String> Url = new ArrayList<String>();
24
25      ArrayList<String> monthIN = new ArrayList<String>();
26      ArrayList<String> totalAC = new ArrayList<String>();
27      ArrayList<String> sumByte = new ArrayList<String>();
28
29      ArrayList<Integer> hour_a = new ArrayList<Integer>();
30      ArrayList<Integer> qtdAcess = new ArrayList<Integer>();
31      ArrayList<Integer> totalDadosGB = new ArrayList<Integer>();
32      ArrayList<Integer> falhAs = new ArrayList<Integer>();
33
34      ArrayList<String> typeMime = new ArrayList<String>();
35      ArrayList<Double> bytesMime = new ArrayList<Double>();
36
37      String totalFace = "";
38      String totalInsta = "";
39      String totalYou = "";
40      String totalTwitter = "";
41      String totalAcessos = "";
42
43
44      public ArrayList<String> doInBackground(String year, String month,
            String searchText, int switchValue) {
45          Connection conexaoLocal;
46
47          try
48          {
49              DBConnection SolicitaCon = new DBConnection();
50              conexaoLocal = SolicitaCon.connectionclass();        //
                    Connect to database
51              if (conexaoLocal == null)
52              {
53                  ConnectionResult = "Check Your Internet Access!";
54              }
55              else
56              {
57                  // Change below query according to your own database.
58
59                  Statement stmt = conexaoLocal.createStatement();
```

```
60
61                  switch (switchValue) {
62                      case 1: {
63
64                          ResultSet rs = stmt.executeQuery("SELECT ip FROM
                               ipMonth GROUP BY ip");
65                          ResultSetMetaData rsmd = rs.getMetaData();
66
67                          Ar1.clear();
68
69                          while (rs.next()) {
70
71                              Ar1.add(rs.getString(1).toString());
72                          }
73
74
75                          break;
76                      }
77                      case 2: {
78
79                          ResultSet rs = stmt.executeQuery("select ipadress,
                               day(squidtimestamp), SUBSTRING_INDEX(fullUrl,
                               '/', 3)  from logSquid.squidLog where month(
                              squidtimestamp) = "+ month +" and fullUrl LIKE
                              \"%"+ searchText +"%\" group by ipadress");
80                          ResultSetMetaData rsmd = rs.getMetaData();
81
82                          ips.clear();
83                          dias.clear();
84                          Url.clear();
85
86                          while (rs.next()) {
87
88                              ips.add(rs.getString(1).toString());
89                              dias.add(rs.getString(2).toString());
90                              Url.add(rs.getString(3).toString());
91                          }
92
93
```

```
94                          break;
95                      }
96
97                      case 3: {
98
99
100                         ResultSet rs = stmt.executeQuery("select monthIN,
                               totalAC, max(sumByte/1000000) from (select (id)
                                idIN, (month_num) monthIN from ipMonth where
                               ip = '" +  searchText +"') idpIN, ipTotal \n" +
101                                "where id_ipMonth = idpIN.idIN \n" +
102                                "group by idpIN.idIN, id_ipMonth");
103                         ResultSetMetaData rsmd = rs.getMetaData();
104
105                         monthIN.clear();
106                         totalAC.clear();
107                         sumByte.clear();
108
109                         while (rs.next()) {
110
111                             monthIN.add(rs.getString(1).toString());
112                             totalAC.add(rs.getString(2).toString());
113                             sumByte.add(rs.getString(3).toString());
114                         }
115
116
117
118                         break;
119                     }
120
121                     case 4: {
122
123
124                         ResultSet rs = stmt.executeQuery("select
                               SUBSTRING_INDEX(fullUrl, '/', 3), count(
                               SUBSTRING_INDEX(fullUrl, '/', 3)) from logSquid
                               .squidLog \n" +
125                                "where ipadress = '" + searchText + "' and
                                    month(squidtimestamp) = "+month+"
```

```
                                group by\n" +
126                             "ipadress, month(squidtimestamp),
                                SUBSTRING_INDEX(fullUrl, '/', 3) order
                                by 2 desc\n" +
127                             "limit 3");
128                     ResultSetMetaData rsmd = rs.getMetaData();
129                     Url.clear();
130                     Ar1.clear();
131
132                     while (rs.next()) {
133                         //reaproveitando variaveis
134                         Url.add(rs.getString(1).toString()); // URL
135                         Ar1.add(rs.getString(2).toString()); // QTD
                            Acesso
136                     }
137
138
139
140                     break;
141                 }
142
143                 case 5: {
144
145                     ResultSet rs = stmt.executeQuery("select qtdARS,
                        totalAc from ipRedeSo, ipMonth, ipTotal where
                        ipRedeSo.id_ipMonth = ipMonth.id and ipMonth.
                        month_num = "+ month +" and ipMonth.ip = '" +
                        searchText + "' and redeSoci LIKE \"%facebook
                        %\" \n" +
146                             "group by ipMonth.ip");
147                     while (rs.next()) {
148                     totalFace = rs.getString(1).toString();
149                     }
150
151                     ResultSet rs2 = stmt.executeQuery("select qtdARS,
                        totalAc from ipRedeSo, ipMonth, ipTotal \n" +
152                             "where ipRedeSo.id_ipMonth = ipMonth.id
                                and ipMonth.month_num = "+ month +" and
                                 ipMonth.ip = '" + searchText + "' and
```

```
                    redeSoci LIKE \"%instagram%\" \n" +
153                 "group by ipMonth.ip");
154             while (rs2.next()) {
155             totalInsta = rs2.getString(1).toString();
156             }
157
158             ResultSet rs3 = stmt.executeQuery("select qtdARS,
                    totalAc from ipRedeSo, ipMonth, ipTotal \n" +
159                 "where ipRedeSo.id_ipMonth = ipMonth.id
                        and ipMonth.month_num = "+ month +" and
                         ipMonth.ip = '" + searchText + "' and
                        redeSoci LIKE \"%youtube%\" \n" +
160                 "group by ipMonth.ip");
161             while (rs3.next()) {
162             totalYou = rs3.getString(1).toString();
163             }
164
165             ResultSet rs4 = stmt.executeQuery("select qtdARS,
                    totalAc from ipRedeSo, ipMonth, ipTotal \n" +
166                 "where ipRedeSo.id_ipMonth = ipMonth.id
                        and ipMonth.month_num = "+ month +" and
                          ipMonth.ip = '" + searchText + "' and
                         redeSoci LIKE \"%twitter%\" \n" +
167                 "group by ipMonth.ip");
168             while (rs4.next()) {
169             totalTwitter = rs4.getString(1).toString();
170             }
171
172             ResultSet rs5 = stmt.executeQuery("select  totalAC
                     from (select (id) idIN, (month_num) monthIN
                    from ipMonth where ip = '"+ searchText +"')
                    idpIN, ipTotal\n" +
173                 "where id_ipMonth = idpIN.idIN and monthIN
                         = "+ month +" group by idpIN.idIN,
                        id_ipMonth");
174
175             while (rs5.next()) {
176                 totalAcessos = rs5.getString(1).toString();
177             }
```

```
178
179
180
181                         break;
182                     }
183
184                     case 6: {
185
186
187                         ResultSet rsRede = stmt.executeQuery("select
                               a_hour, totalAC, qtdBytes/1000000000, falha
                               from relaRede where month_num = "+ month +" and
                                year_num = "+ year);
188                         ResultSetMetaData rsmd = rsRede.getMetaData();
189
190                         hour_a.clear();
191                         qtdAcess.clear();
192                         totalDadosGB.clear();
193                         falhAs.clear();
194
195                         while (rsRede.next()) {
196
197                             hour_a.add(rsRede.getInt(1));
198                             qtdAcess.add(rsRede.getInt(2));
199                             totalDadosGB.add(rsRede.getInt(3));
200                             falhAs.add(rsRede.getInt(4));
201                         }
202
203
204                         break;
205                     }
206
207                     case 7: {
208
209                         ResultSet rsMime = stmt.executeQuery("select
                               SUBSTRING_INDEX(mimeType, '/', 1), ipMimeTi.
                               qtdBytes/1000000 from ipMimeTi, ipMonth \n" +
210                                    "where ipMonth.id = ipMimeTi.id_ipMonth
                                          and ip = '"+ searchText +"' and
```

```
                                        month_num = "+ month +" and \n" +
211                                   "year_num = "+ year + " group by
                                        SUBSTRING_INDEX(mimeType, '/', 1)");
212                         ResultSetMetaData rsmd = rsMime.getMetaData();
213
214                         bytesMime.clear();
215                         typeMime.clear();
216
217                         while (rsMime.next()) {
218                             typeMime.add(rsMime.getString(1).toString());
219                             bytesMime.add(rsMime.getDouble(2));
220                         }
221
222
223                         break;
224                     }
225
226                 }
227
228
229
230                 ConnectionResult = totalFace;
231                 isSuccess=true;
232                 conexaoLocal.close();
233             }
234         }
235         catch (Exception ex)
236         {
237             isSuccess = false;
238             ConnectionResult = ex.getMessage();
239         }
240
241         return Ar1;
242     }
243
244
245
246
247 }
```

C.4 CLASSE DA PICKERVIEW FONTE: DOS AUTORES

```
1  package parallelcodes.mysqlapp;
2
3  import android.app.DatePickerDialog;
4  import android.app.Dialog;
5  import android.content.DialogInterface;
6  import android.os.Bundle;
7  import android.support.v4.app.DialogFragment;
8  import android.support.v7.app.AlertDialog;
9  import android.view.LayoutInflater;
10 import android.view.View;
11 import android.widget.NumberPicker;
12
13 import java.util.Calendar;
14
15 /**
16  * Created by ankititjunkies on 20/03/18.
17  */
18
19 public class MonthYearPickerDialog extends DialogFragment {
20
21     private static final int MAX_YEAR = 2099;
22     private DatePickerDialog.OnDateSetListener listener;
23
24     public void setListener(DatePickerDialog.OnDateSetListener listener) {
25         this.listener = listener;
26     }
27
28     @Override
29     public Dialog onCreateDialog(Bundle savedInstanceState) {
30         AlertDialog.Builder builder = new AlertDialog.Builder(getActivity
               ());
31         // Get the layout inflater
32         LayoutInflater inflater = getActivity().getLayoutInflater();
33
34         Calendar cal = Calendar.getInstance();
35
36         View dialog = inflater.inflate(R.layout.month_year_picker_dialog,
```

```
            null);
37          final NumberPicker monthPicker = (NumberPicker) dialog.
                findViewById(R.id.picker_month);
38          final NumberPicker yearPicker = (NumberPicker) dialog.findViewById
                (R.id.picker_year);
39
40          monthPicker.setMinValue(1);
41          monthPicker.setMaxValue(12);
42          monthPicker.setValue(cal.get(Calendar.MONTH) + 1);
43
44          int year = cal.get(Calendar.YEAR);
45          yearPicker.setMinValue(2000);
46          yearPicker.setMaxValue(year);
47          yearPicker.setValue(year);
48
49          builder.setView(dialog)
50                  // Add action buttons
51                  .setPositiveButton(R.string.ok, new DialogInterface.
                        OnClickListener() {
52                      @Override
53                      public void onClick(DialogInterface dialog, int id) {
54                          listener.onDateSet(null, yearPicker.getValue(),
                                monthPicker.getValue(), 0);
55                      }
56                  })
57                  .setNegativeButton(R.string.cancel, new DialogInterface.
                        OnClickListener() {
58                      public void onClick(DialogInterface dialog, int id) {
59                          MonthYearPickerDialog.this.getDialog().cancel();
60                      }
61                  });
62          return builder.create();
63      }
64  }
```

C.5 *ACTIVITY* DE RELATÓRIO DA REDE FONTE: DOS AUTORES

```
1   package parallelcodes.mysqlapp;
2
3   import android.app.DatePickerDialog;
```

```
4  import android.content.Intent;
5  import android.graphics.Color;
6  import android.graphics.Typeface;
7  import android.os.StrictMode;
8  import android.support.annotation.NonNull;
9  import android.support.design.widget.BottomNavigationView;
10 import android.support.v7.app.AppCompatActivity;
11 import android.os.Bundle;
12 import android.view.Menu;
13 import android.view.MenuItem;
14 import android.widget.DatePicker;
15 import android.widget.TextView;
16 import android.widget.Toast;
17
18 import com.jjoe64.graphview.GraphView;
19 import com.jjoe64.graphview.series.BarGraphSeries;
20 import com.jjoe64.graphview.series.DataPoint;
21 import com.jjoe64.graphview.series.LineGraphSeries;
22
23 import java.util.ArrayList;
24 import java.util.Calendar;
25 import java.util.List;
26
27 import lecho.lib.hellocharts.model.PieChartData;
28 import lecho.lib.hellocharts.model.SliceValue;
29 import lecho.lib.hellocharts.view.PieChartView;
30
31 public class NetworkActivity extends AppCompatActivity implements
       BottomNavigationView.OnNavigationItemSelectedListener {
32
33
34     private BottomNavigationView navigationView;
35
36     GraphView graphAcesso;
37     GraphView graphBytes;
38     GraphView graphFalhas;
39
40     int yearSelected = 2019;
41     int monthSelected = 11;
```

```
42
43      TextView TV;
44
45      @Override
46      protected void onCreate(Bundle savedInstanceState) {
47          super.onCreate(savedInstanceState);
48          setContentView(R.layout.activity_network);
49          if (android.os.Build.VERSION.SDK_INT > 9) {
50              StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.
                    Builder().permitAll().build();
51              StrictMode.setThreadPolicy(policy);
52          }
53          getSupportActionBar().setTitle("Relat rio de Rede");
54          graphAcesso = (GraphView) findViewById(R.id.graphACessos);
55          graphBytes = (GraphView) findViewById(R.id.graphQuantidade);
56          graphFalhas = (GraphView) findViewById(R.id.graphFalhas);
57
58
59          navigationView = (BottomNavigationView) findViewById(R.id.
                navigationView);
60          navigationView.setOnNavigationItemSelectedListener(this);
61
62          Calendar calendar = Calendar.getInstance();
63          yearSelected = calendar.get(Calendar.YEAR);
64          monthSelected = calendar.get(Calendar.MONTH)+1;
65
66
67          GetData mydataNet = new GetData();
68          mydataNet.doInBackground(String.valueOf(yearSelected), String.
                valueOf(monthSelected), "", 6);
69
70          DataPoint[] dpAcesso = new DataPoint[mydataNet.hour_a.size()];
71
72
73          for (int i=0;i < mydataNet.hour_a.size();i++){
74              dpAcesso[i] = new DataPoint(mydataNet.hour_a.get(i), mydataNet
                    .qtdAcess.get(i));
75          }
76          LineGraphSeries<DataPoint> series = new LineGraphSeries<DataPoint
```

```
        >(dpAcesso);
77      series.setAnimated(true);
78      series.setDrawBackground(true);
79
80      graphAcesso.getViewport().setMinX(0);
81      graphAcesso.getViewport().setMaxX(23);
82      graphAcesso.addSeries(series);
83      graphAcesso.getViewport().setScalable(true);
84
85      graphAcesso.getViewport().setXAxisBoundsManual(true);
86
87
88
89
90      DataPoint[] dpByte = new DataPoint[mydataNet.hour_a.size()];
91
92      for (int i=0;i < mydataNet.hour_a.size();i++){
93          dpByte[i] = new DataPoint(mydataNet.hour_a.get(i), mydataNet.
                totalDadosGB.get(i));
94      }
95      LineGraphSeries<DataPoint> seriesGB = new LineGraphSeries<
            DataPoint>(dpByte);
96      seriesGB.setAnimated(true);
97      seriesGB.setColor(Color.CYAN);
98      seriesGB.setDrawBackground(true);
99
100     graphBytes.getViewport().setScalable(true);
101     graphBytes.addSeries(seriesGB);
102     graphBytes.getViewport().setMinX(0);
103     graphBytes.getViewport().setMaxX(23);
104     graphBytes.getViewport().setXAxisBoundsManual(true);
105
106     DataPoint[] dpFalha = new DataPoint[mydataNet.hour_a.size()];
107
108     for (int i=0;i < mydataNet.hour_a.size();i++){
109         dpFalha[i] = new DataPoint(mydataNet.hour_a.get(i), mydataNet.
                falhAs.get(i));
110     }
111     BarGraphSeries<DataPoint> seriesFalha = new BarGraphSeries<
```

```
            DataPoint>(dpFalha);
112         seriesFalha.setAnimated(true);
113         seriesFalha.setColor(Color.RED);
114         seriesFalha.setSpacing(4);
115
116         graphFalhas.getViewport().setScalable(true);
117         graphFalhas.addSeries(seriesFalha);
118         graphFalhas.getViewport().setMinX(0);
119         graphFalhas.getViewport().setMaxX(23);
120         graphFalhas.getViewport().setXAxisBoundsManual(true);
121
122
123
124
125         graphAcesso.addSeries(series);
126
127     }
128
129
130     @Override
131     public boolean onNavigationItemSelected(@NonNull MenuItem item) {
132         switch (item.getItemId()) {
133             case R.id.analiticbut: {
134                 break;
135             }
136             case R.id.homebut: {
137                 Intent i=new Intent(NetworkActivity.this, MainActivity.
                        class);
138                 startActivity(i);
139                 break;
140             }
141             case R.id.searchbut: {
142                 Intent i=new Intent(NetworkActivity.this, SearchActivity.
                        class);
143                 startActivity(i);
144                 break;
145             }
146         }
147         return true;
```

```
148     }
149
150     @Override
151     public boolean onCreateOptionsMenu(Menu menu) {
152         // Inflate the menu; this adds items to the action bar if it is
                present.
153         getMenuInflater().inflate(R.menu.menu_main_dual, menu);
154
155
156         return true;
157     }
158
159     @Override
160     public boolean onOptionsItemSelected(MenuItem item) { switch(item.
            getItemId()) {
161
162
163         case R.id.action_user:
164
165             Calendar calendar = Calendar.getInstance();
166             yearSelected = calendar.get(Calendar.YEAR);
167             monthSelected = calendar.get(Calendar.MONTH);
168
169
170
171             MonthYearPickerDialog pickerDialog = new MonthYearPickerDialog
                    ();
172             pickerDialog.setListener(new DatePickerDialog.
                    OnDateSetListener() {
173                 @Override
174                 public void onDateSet(DatePicker datePicker, int year, int
                        month, int i2) {
175                     Toast.makeText(NetworkActivity.this, year + "-" +
                            month, Toast.LENGTH_SHORT).show();
176                     yearSelected = year;
177                     monthSelected = month;
178
179
180
```

```
181
182                 GetData mydataNet = new GetData();
183                 mydataNet.doInBackground(String.valueOf(yearSelected),
                        String.valueOf(monthSelected), "", 6);
184
185                 DataPoint[] dpAcesso = new DataPoint[mydataNet.hour_a.
                        size()];
186
187                 //TV = (TextView) findViewById(R.id.textAcessos);
188                 //TV.setText("Tamanho: "+mydataNet.hour_a.size() + " "
                        + monthSelected);
189
190                 for (int i=0;i < mydataNet.hour_a.size();i++){
191                     dpAcesso[i] = new DataPoint(mydataNet.hour_a.get(i
                            ), mydataNet.qtdAcess.get(i));
192                 }
193                 LineGraphSeries<DataPoint> series = new
                        LineGraphSeries<DataPoint>(dpAcesso);
194                 series.setAnimated(true);
195                 series.setDrawBackground(true);
196
197                 graphAcesso.getViewport().setMinX(0);
198                 graphAcesso.getViewport().setMaxX(23);
199                 graphAcesso.removeAllSeries();
200                 graphAcesso.addSeries(series);
201                 graphAcesso.getViewport().setScalable(true);
202
203                 graphAcesso.getViewport().setXAxisBoundsManual(true);
204
205
206
207
208                 DataPoint[] dpByte = new DataPoint[mydataNet.hour_a.
                        size()];
209
210                 for (int i=0;i < mydataNet.hour_a.size();i++){
211                     dpByte[i] = new DataPoint(mydataNet.hour_a.get(i),
                            mydataNet.totalDadosGB.get(i));
212                 }
```

```
213                 LineGraphSeries<DataPoint> seriesGB = new
                        LineGraphSeries<DataPoint>(dpByte);
214                 seriesGB.setAnimated(true);
215                 seriesGB.setColor(Color.CYAN);
216                 seriesGB.setDrawBackground(true);
217
218                 graphBytes.getViewport().setScalable(true);
219                 graphBytes.removeAllSeries();
220                 graphBytes.addSeries(seriesGB);
221                 graphBytes.getViewport().setMinX(0);
222                 graphBytes.getViewport().setMaxX(23);
223                 graphBytes.getViewport().setXAxisBoundsManual(true);
224
225                 DataPoint[] dpFalha = new DataPoint[mydataNet.hour_a.
                        size()];
226
227                 for (int i=0;i < mydataNet.hour_a.size();i++){
228                     dpFalha[i] = new DataPoint(mydataNet.hour_a.get(i)
                            , mydataNet.falhAs.get(i));
229                 }
230                 BarGraphSeries<DataPoint> seriesFalha = new
                        BarGraphSeries<DataPoint>(dpFalha);
231                 seriesFalha.setAnimated(true);
232                 seriesFalha.setColor(Color.RED);
233                 seriesFalha.setSpacing(4);
234
235                 graphFalhas.getViewport().setScalable(true);
236                 graphFalhas.removeAllSeries();
237                 graphFalhas.addSeries(seriesFalha);
238                 graphFalhas.getViewport().setMinX(0);
239                 graphFalhas.getViewport().setMaxX(23);
240                 graphFalhas.getViewport().setXAxisBoundsManual(true);
241
242
243
244
245                 // series2.setAnimated(true);
246                 //series2.setSpacing(4);
247                 //   series2.setDataWidth(5);
```

```
248                         //series2.setColor(Color.BLACK);
249
250                         graphAcesso.addSeries(series);
251
252                     }
253                 });
254                 pickerDialog.show(getSupportFragmentManager(), "
                        MonthYearPickerDialog");
255
256
257
258
259                 return(true);
260         }
261             return(super.onOptionsItemSelected(item));
262         }
263
264 }
```

C.6 *ACTIVITY* DE PESQUISA NA BASE FONTE: DOS AUTORES

```
1  package parallelcodes.mysqlapp;
2
3  import android.app.DatePickerDialog;
4  import android.app.ProgressDialog;
5  import android.content.Intent;
6  import android.graphics.Color;
7  import android.os.Handler;
8  import android.os.StrictMode;
9  import android.support.annotation.NonNull;
10 import android.support.design.widget.BottomNavigationView;
11 import android.support.v4.view.MenuItemCompat;
12 import android.support.v7.app.AppCompatActivity;
13 import android.os.Bundle;
14 import android.view.Menu;
15 import android.view.MenuItem;
16 import android.view.View;
17 import android.widget.DatePicker;
18 import android.widget.ListView;
19 import android.widget.SearchView;
```

```
20 import android.widget.TextView;
21 import android.widget.Toast;
22
23
24 import java.util.ArrayList;
25 import java.util.Calendar;
26 import java.util.List;
27
28 import de.codecrafters.tableview.TableView;
29 import de.codecrafters.tableview.listeners.TableDataClickListener;
30 import de.codecrafters.tableview.model.TableColumnWeightModel;
31 import de.codecrafters.tableview.toolkit.SimpleTableDataAdapter;
32 import de.codecrafters.tableview.toolkit.SimpleTableHeaderAdapter;
33
34
35
36 public class SearchActivity extends AppCompatActivity implements
       BottomNavigationView.OnNavigationItemSelectedListener, SearchView.
       OnQueryTextListener{
37
38     private BottomNavigationView navigationView;
39     public TextView TV2;
40     public ListView listView;
41
42     int yearSelected;
43     int monthSelected;
44
45     ArrayList<String> Ar = new ArrayList<String>();
46     ArrayList<String> userList;
47
48     TableView tableView;
49
50     String[][] DATA;
51     GetData mydata =new GetData();
52
53     private static final String[] TABLE_HEADERS = { "IP", "Url", "Dia" };
54
55     @Override
56     protected void onCreate(Bundle savedInstanceState) {
```

```
57          super.onCreate(savedInstanceState);
58          setContentView(R.layout.activity_search);
59          getSupportActionBar().setTitle("Busca na Base");
60          if (android.os.Build.VERSION.SDK_INT > 9) {
61              StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.
                    Builder().permitAll().build();
62              StrictMode.setThreadPolicy(policy);
63          }
64
65          tableView = (TableView) findViewById(R.id.tableView);
66          tableView.setHeaderAdapter(new SimpleTableHeaderAdapter(this,
                TABLE_HEADERS));
67
68          TableColumnWeightModel columnModel = new TableColumnWeightModel(3)
                ;
69          columnModel.setColumnWeight(0, 4);
70          columnModel.setColumnWeight(1, 7);
71          columnModel.setColumnWeight(2, 2);
72          //columnModel.setColumnWeight(3, 2);
73          tableView.setColumnModel(columnModel);
74
75
76          //listView = (ListView) findViewById(R.id.listViewData);
77
78
79          Calendar calendar = Calendar.getInstance();
80          yearSelected = calendar.get(Calendar.YEAR);
81          monthSelected = calendar.get(Calendar.MONTH);
82
83          navigationView = (BottomNavigationView) findViewById(R.id.
                navigationView);
84          navigationView.setOnNavigationItemSelectedListener(this);
85
86          MonthYearPickerDialog pickerDialog = new MonthYearPickerDialog();
87          pickerDialog.setListener(new DatePickerDialog.OnDateSetListener()
                {
88              @Override
89              public void onDateSet(DatePicker datePicker, int year, int
                    month, int i2) {
```

```
90                  Toast.makeText(SearchActivity.this, year + "-" + month,
                        Toast.LENGTH_SHORT).show();
91                  yearSelected = year;
92                  monthSelected = month;
93              }
94          });
95          pickerDialog.show(getSupportFragmentManager(), "
                MonthYearPickerDialog");
96      }
97
98
99
100     @Override
101     public boolean onCreateOptionsMenu(Menu menu) {
102
103         getMenuInflater().inflate(R.menu.menu_search, menu);
104
105         MenuItem menuItem = menu.findItem(R.id.search);
106         SearchView searchView = (SearchView) MenuItemCompat.getActionView(
                menuItem);
107         searchView.setOnQueryTextListener(this);
108         return true;
109     }
110     private ProgressDialog progress;
111
112     private Handler handler;
113     @Override
114     public boolean onQueryTextSubmit(final String query) {
115         if (android.os.Build.VERSION.SDK_INT > 9) {
116             StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.
                    Builder().permitAll().build();
117             StrictMode.setThreadPolicy(policy);
118         }
119
120        // ArrayList<SearchData> resultPesquisa = new ArrayList<SearchData
               >();
121         String[][] resultPesquisa = null;
122
123         String[] header={"IP","Dia do mes","Site"};
```

```
124
125         String yearString = String.valueOf(yearSelected);
126         String monthString = String.valueOf(monthSelected);
127
128         if (query != "") {
129
130             progress = new ProgressDialog(this);
131             progress.setMessage("Carregando...");
132             progress.setCancelable(false);
133             progress.setProgressStyle(ProgressDialog.STYLE_SPINNER);
134             progress.show();
135
136
137             final int totalProgressTime = 100;
138             final Thread t = new Thread() {
139                 @Override
140                 public void run() {
141                     int jumpTime = 0;
142
143                     String yearString = String.valueOf(yearSelected);
144                     String monthString = String.valueOf(monthSelected);
145                     String TextTyped = query;
146
147                     //while(jumpTime < totalProgressTime) {
148                     try {
149                         mydata.doInBackground(yearString, monthString,
                                TextTyped, 2);
150                         // jumpTime += 5;
151                         // progress.setProgress(jumpTime);
152                     } catch (Exception e) {
153                         // TODO Auto-generated catch block
154                         e.printStackTrace();
155                     }
156                     // }
157
158                     DATA = new String[mydata.ips.size()][3];
159                     int a = 0, b = 0, c = 0;
160
161                     for (int i = 0; i < DATA.length; i++) {
```

```
162                         for (int j = 0; j < DATA[i].length; j++) {
163
164                             if (j == 0) {
165                                 DATA[i][j] = mydata.ips.get(a).toString();
166                                 a++;
167                             } else if (j == 1) {
168                                 DATA[i][j] = mydata.Url.get(b).toString();
169                                 b++;
170                             } else if (j == 2) {
171                                 DATA[i][j] = mydata.dias.get(c).toString()
                                        ;
172                                 c++;
173                             }
174                         }
175                     }
176                     handler.obtainMessage().sendToTarget();
177                     // tableView.getDataAdapter().notifyDataSetChanged();
178                     //  Bundle tempBundle = new Bundle();
179                     //  onCreate(tempBundle);
180                 }
181             };
182             t.start();
183
184             handler = new Handler() {
185                 public void handleMessage(android.os.Message msg) {
186                     tableView.getDataAdapter().notifyDataSetChanged();
187                     if (progress.isShowing()) progress.dismiss();
188                     onQueryTextSubmit("");
189                 }
190             };
191
192         }
193         DATA = new String[mydata.ips.size()][3];
194         int a = 0, b = 0, c =0;
195
196         for (int i = 0; i < DATA.length; i++) {
197             for (int j = 0; j < DATA[i].length; j++) {
198
199                 if (j==0){
```

```
200                 DATA[i][j] = mydata.ips.get(a).toString();
201                 a++;
202                 }else if(j==1){
203                     DATA[i][j] = mydata.Url.get(b).toString();
204                     b++;
205                 }else if(j==2){
206                     DATA[i][j] = mydata.dias.get(c).toString();
207                     c++;
208                 }
209
210
211             }
212         }
213
214         tableView.setDataAdapter(new SimpleTableDataAdapter(this, DATA));
215         return false;
216     }
217
218
219
220
221     @Override
222     public boolean onQueryTextChange(String newText) {
223         // User changed the text
224         return false;
225
226     }
227
228     @Override
229     public boolean onNavigationItemSelected(@NonNull MenuItem item) {
230         switch (item.getItemId()) {
231             case R.id.analiticbut: {
232                 Intent i=new Intent(SearchActivity.this, NetworkActivity.
                        class);
233                 startActivity(i);
234                 break;
235             }
236             case R.id.homebut: {
237                 Intent i=new Intent(SearchActivity.this, MainActivity.
```

```
                    class);
238                 startActivity(i);
239                 break;
240             }
241             case R.id.searchbut: {
242
243                 break;
244             }
245         }
246         return true;
247     }
248 }
```

C.7 *ACTIVITY* DE DADOS DE IP FONTE: DOS AUTORES

```
1  package parallelcodes.mysqlapp;
2
3  import android.annotation.SuppressLint;
4  import android.app.DatePickerDialog;
5  import android.app.ProgressDialog;
6  import android.content.Intent;
7  import android.graphics.Color;
8  import android.os.Handler;
9  import android.os.StrictMode;
10 import android.support.annotation.NonNull;
11 import android.support.design.widget.BottomNavigationView;
12 import android.support.v7.app.AppCompatActivity;
13 import android.os.Bundle;
14 import android.view.Menu;
15 import android.view.MenuItem;
16 import android.view.View;
17 import android.widget.Button;
18 import android.widget.DatePicker;
19 import android.widget.TextView;
20 import android.widget.Toast;
21 import android.app.Activity;
22 import android.app.ProgressDialog;
23 import android.os.Bundle;
24 import android.os.Handler;
25
```

```
26 import com.jjoe64.graphview.GraphView;
27 import com.jjoe64.graphview.series.DataPoint;
28 import com.jjoe64.graphview.series.LineGraphSeries;
29 import com.jjoe64.graphview.series.BarGraphSeries;
30
31 import java.util.Calendar;
32
33 import de.codecrafters.tableview.TableView;
34 import de.codecrafters.tableview.model.TableColumnWeightModel;
35 import de.codecrafters.tableview.toolkit.SimpleTableDataAdapter;
36 import de.codecrafters.tableview.toolkit.SimpleTableHeaderAdapter;
37
38 public class UserDataActivity extends AppCompatActivity implements
       BottomNavigationView.OnNavigationItemSelectedListener {
39
40     private Handler handler;
41     private ProgressDialog dialog;
42
43     private BottomNavigationView navigationView;
44     int yearSelected = 2019;
45     int monthSelected = 11;
46     public TextView TV;
47     public Button selectDate;
48     GraphView graph;
49     public String sessionId = "";
50
51     private static final String[] TABLE_HEADERS = { "Url", "Acessos" };
52     TableView tableView;
53
54     String[][] DATA = new String[3][2];
55
56     @Override
57     protected void onCreate(Bundle savedInstanceState) {
58         super.onCreate(savedInstanceState);
59         setContentView(R.layout.activity_user_data);
60
61         if (android.os.Build.VERSION.SDK_INT > 9) {
62             StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.
                   Builder().permitAll().build();
```

```
63              StrictMode.setThreadPolicy(policy);
64          }
65
66
67          dialog = new ProgressDialog(this);
68
69          TV = (TextView) findViewById(R.id.textViewforTest);
70          selectDate = (Button) findViewById(R.id.buttonSelectMonth);
71
72          navigationView = (BottomNavigationView) findViewById(R.id.
                navigationView);
73          navigationView.setOnNavigationItemSelectedListener(this);
74
75          sessionId = getIntent().getStringExtra("EXTRA_SESSION_ID");
76          getSupportActionBar().setTitle("Rel. de Dados - "+sessionId);
77
78          tableView = (TableView) findViewById(R.id.tableView);
79          tableView.setHeaderAdapter(new SimpleTableHeaderAdapter(this,
                TABLE_HEADERS));
80
81
82          TableColumnWeightModel columnModel = new TableColumnWeightModel(2)
                ;
83          columnModel.setColumnWeight(0, 7);
84          columnModel.setColumnWeight(1, 3);
85          //columnModel.setColumnWeight(3, 2);
86          tableView.setColumnModel(columnModel);
87
88          Calendar calendar = Calendar.getInstance();
89          yearSelected = calendar.get(Calendar.YEAR);
90          monthSelected = calendar.get(Calendar.MONTH);
91
92          //tableView.setDataAdapter(new SimpleTableDataAdapter(this, DATA))
                ;
93
94
95          selectDate.setOnClickListener(new View.OnClickListener() {
96
97              public void onClick(View v) {
```

```
98
99                  Calendar calendar = Calendar.getInstance();
100                 yearSelected = calendar.get(Calendar.YEAR);
101                 monthSelected = calendar.get(Calendar.MONTH);
102
103                 MonthYearPickerDialog pickerDialog = new
                        MonthYearPickerDialog();
104                 pickerDialog.setListener(new DatePickerDialog.
                        OnDateSetListener() {
105                     @Override
106                     public void onDateSet(DatePicker datePicker, int year,
                            int month, int i2) {
107                         Toast.makeText(UserDataActivity.this, year + "-" +
                                month, Toast.LENGTH_SHORT).show();
108                         yearSelected = year;
109                         monthSelected = month;
110
111                         dialog.setMessage("Carregando...");
112                         dialog.setCancelable(false);
113                         dialog.setProgressStyle(ProgressDialog.
                                STYLE_SPINNER);
114                         dialog.show();
115
116                         new Thread() {
117                             public void run() {
118
119                                 Calendar c = Calendar.getInstance();
120                                 int year = c.get(Calendar.YEAR);
121                                 int month = c.get(Calendar.MONTH);
122                                 yearSelected = year;
123                                 monthSelected = month;
124                                 GetData mydataTable = new GetData();
125                                 mydataTable.doInBackground(String.valueOf(
                                        yearSelected), String.valueOf(
                                        monthSelected), sessionId, 4);
126
127
128                                 if (mydataTable.Url.size() > 0) {
129
```

```
130                                 for (int i=0;i<mydataTable.Url.size();
                                        i++) {
131                                     DATA[i][0]=mydataTable.Url.get(i).
                                            toString();
132                                 }
133                                 for (int i=0;i<mydataTable.Ar1.size();
                                        i++) {
134                                     DATA[i][1]=mydataTable.Ar1.get(i).
                                            toString();
135                                 }
136
137
138                             }
139
140
141
142                             handler.obtainMessage().sendToTarget();
143
144                         }
145                     }.start();
146
147
148                     handler = new Handler() {
149                         public void handleMessage(android.os.Message
                                msg) {
150                             if(dialog.isShowing()) dialog.dismiss();
151                             tableView.setDataAdapter(new
                                    SimpleTableDataAdapter(
                                    getApplicationContext(), DATA));
152                             // tableView.getDataAdapter().
                                    notifyDataSetChanged();
153
154                         }
155                     };
156
157
158                 }
159             });
160             pickerDialog.show(getSupportFragmentManager(), "
```

```
                    MonthYearPickerDialog");
161
162
163
164
165                 //tableView.getDataAdapter().notifyDataSetChanged();
166                //  TV.setText(DATA[0][0]);
167             }
168         });
169
170
171         GetData mydata =new GetData();
172         mydata.doInBackground("","",sessionId,3);
173
174
175         GraphView graphAC = (GraphView) findViewById(R.id.graphAC);
176         GraphView graphByte = (GraphView) findViewById(R.id.graphTF);
177
178         DataPoint[] dpAcesso = new DataPoint[mydata.monthIN.size()];
179
180
181                 for (int i=0;i < mydata.monthIN.size();i++){
182                     dpAcesso[i] = new DataPoint(Integer.parseInt(mydata.
                        monthIN.get(i)), Integer.parseInt(mydata.totalAC.
                        get(i)));
183
184                 }
185
186         LineGraphSeries<DataPoint> series = new LineGraphSeries<DataPoint
            >(dpAcesso);
187         series.setAnimated(true);
188         series.setDrawBackground(true);
189
190         DataPoint[] dpMBytes = new DataPoint[mydata.monthIN.size()];
191
192
193
194        for (int i=0;i < mydata.monthIN.size();i++){
195             dpMBytes[i] = new DataPoint(Integer.parseInt(mydata.monthIN.
```

```
                get(i)), Double.parseDouble(mydata.sumByte.get(i)));
196
197         }
198
199
200         BarGraphSeries<DataPoint> series2 = new BarGraphSeries<DataPoint>(
                dpMBytes);
201         series2.setAnimated(true);
202         series2.setSpacing(4);
203        //   series2.setDataWidth(5);
204         series2.setColor(Color.BLACK);
205
206         graphAC.getViewport().setScalable(true);
207         graphAC.addSeries(series);
208         graphAC.getViewport().setMinX(0);
209         graphAC.getViewport().setMaxX(12);
210
211
212         graphByte.addSeries(series2);
213         graphByte.getViewport().setScalable(true);
214         graphByte.getViewport().setMinX(0);
215         graphByte.getViewport().setMaxX(12);
216
217         graphByte.setScaleX(1);
218         graphAC.setScaleX(1);
219
220
221         graphByte.getViewport().setXAxisBoundsManual(true);
222         graphAC.getViewport().setXAxisBoundsManual(true);
223
224     }
225
226
227
228     @Override
229     public boolean onNavigationItemSelected(@NonNull MenuItem item) {
230         switch (item.getItemId()) {
231             case R.id.analiticbut: {
232                 Intent i=new Intent(UserDataActivity.this, NetworkActivity
```

```
                    .class);
233                 startActivity(i);
234
235                 break;
236             }
237             case R.id.homebut: {
238                 Intent i=new Intent(UserDataActivity.this, MainActivity.
                        class);
239                 startActivity(i);
240                 break;
241             }
242             case R.id.searchbut: {
243                 Intent i=new Intent(UserDataActivity.this, SearchActivity.
                        class);
244                 startActivity(i);
245                 break;
246             }
247         }
248         return true;
249     }
250
251
252     @Override
253     public boolean onCreateOptionsMenu(Menu menu) {
254         // Inflate the menu; this adds items to the action bar if it is
                present.
255         getMenuInflater().inflate(R.menu.menu_main, menu);
256         return true;
257     }
258
259     @Override
260     public boolean onOptionsItemSelected(MenuItem item) { switch(item.
            getItemId()) {
261         case R.id.action_user:
262
263
264             Intent intent = new Intent(UserDataActivity.this,
                    SocialMediaActivity.class);
265             intent.putExtra("EXTRA_SESSION_ID", sessionId);
```

```
266                 startActivity(intent);
267
268
269                 return(true);
270         }
271             return(super.onOptionsItemSelected(item));
272         }
273
274 }
```

C.8 *ACTIVITY* DE DADOS DE REDES SOCIAIS FONTE: DOS AUTORES

```
1  package parallelcodes.mysqlapp;
2
3  import android.app.DatePickerDialog;
4  import android.content.Intent;
5  import android.graphics.Color;
6  import android.graphics.Typeface;
7  import android.os.StrictMode;
8  import android.support.design.widget.BottomNavigationView;
9  import android.support.design.widget.NavigationView;
10 import android.support.v4.content.ContextCompat;
11 import android.support.v7.app.AppCompatActivity;
12 import android.os.Bundle;
13 import android.support.v7.view.menu.ActionMenuItem;
14 import android.view.Menu;
15 import android.view.MenuItem;
16 import android.widget.DatePicker;
17 import android.widget.TextView;
18 import android.widget.Toast;
19
20 import java.util.ArrayList;
21 import java.util.Calendar;
22 import java.util.List;
23
24 import lecho.lib.hellocharts.model.PieChartData;
25 import lecho.lib.hellocharts.model.SliceValue;
26 import lecho.lib.hellocharts.view.PieChartView;
27
28 public class SocialMediaActivity extends AppCompatActivity {
```

```
29
30      public String sessionId = "";
31
32      private BottomNavigationView navigationView;
33      int yearSelected = 2019;
34      int monthSelected = 11;
35
36      TextView ResumoIP;
37      TextView TotalAcessosRede;
38      TextView qtdYoutube;
39      TextView qtdFace;
40      TextView qtdTwitter;
41      TextView qtdInsta;
42
43      @Override
44      protected void onCreate(Bundle savedInstanceState) {
45          super.onCreate(savedInstanceState);
46          setContentView(R.layout.activity_social_media);
47        //  TV2 = (TextView) findViewById(R.id.textView4);
48
49
50
51          if (android.os.Build.VERSION.SDK_INT > 9) {
52              StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.
                    Builder().permitAll().build();
53              StrictMode.setThreadPolicy(policy);
54          }
55
56          ResumoIP = (TextView) findViewById(R.id.textTitle);
57          TotalAcessosRede = (TextView) findViewById(R.id.textQtdTotal);
58          qtdYoutube = (TextView) findViewById(R.id.textYou);
59          qtdFace = (TextView) findViewById(R.id.textQtFace);
60          qtdTwitter = (TextView) findViewById(R.id.textTwitter);
61          qtdInsta = (TextView) findViewById(R.id.textInst);
62
63          Calendar calendar = Calendar.getInstance();
64          yearSelected = calendar.get(Calendar.YEAR);
65          monthSelected = calendar.get(Calendar.MONTH)+1;
66
```

```
67          sessionId = getIntent().getStringExtra("EXTRA_SESSION_ID");
68          getSupportActionBar().setTitle("Rel. Redes Sociais");
69
70          GetData mydata = new GetData();
71          mydata.doInBackground(String.valueOf(yearSelected), String.valueOf
                (monthSelected), sessionId, 5);
72
73           ResumoIP.setTextSize(15);
74           ResumoIP.setTypeface(ResumoIP.getTypeface(), Typeface.BOLD);
75           ResumoIP.setText("Resumo de Acessos - IP: "+ sessionId);
76
77          TotalAcessosRede.setTextSize(12);
78          TotalAcessosRede.setText("Quantidade de Acessos(m s): "+mydata.
                totalAcessos);
79
80          qtdYoutube.setTextSize(12);
81          qtdYoutube.setText("Acessos YouTube(m s): "+mydata.totalYou);
82
83          qtdFace.setTextSize(12);
84          qtdFace.setText("Acessos Facebook(m s): "+mydata.totalFace);
85
86          qtdTwitter.setTextSize(12);
87          qtdTwitter.setText("Acessos Twitter(m s): "+mydata.totalTwitter);
88
89          qtdInsta.setTextSize(12);
90          qtdInsta.setText("Acessos Instagram(m s): "+mydata.totalInsta);
91
92
93
94
95
96          //
                ____________________________________________________________

97          PieChartView pieChartViewFace = findViewById(R.id.chartFace);
98          List<SliceValue> pieDataFace = new ArrayList<>();
99
100         if((mydata.totalAcessos != "") && (mydata.totalFace != "")) {
101
```

```
102
103             pieDataFace.add(new SliceValue(Integer.parseInt(mydata.
                    totalFace), Color.BLUE));
104             pieDataFace.add(new SliceValue(Integer.parseInt(mydata.
                    totalAcessos), Color.GRAY));
105
106             PieChartData pieChartDataFace = new PieChartData(pieDataFace);
107             pieChartDataFace.setHasCenterCircle(true).setCenterText1("
                    Facebook");
108             pieChartDataFace.setCenterText1FontSize(11);
109
110             pieChartViewFace.setPieChartData(pieChartDataFace);
111         }
112             // TV2.setText("Face: " + mydata.totalFace + "  Total: " +
                   mydata.totalAcessos + " Session: " + sessionId + " data: " +
                    mydata.ConnectionResult);
113
114
115     //
        ____________________________________________________________________

116         PieChartView pieChartViewInsta = findViewById(R.id.chartInsta);
117         List<SliceValue> pieDataInsta = new ArrayList<>();
118
119         if((mydata.totalAcessos != "") && (mydata.totalInsta != "")) {
120
121         pieDataInsta.add(new SliceValue(Integer.parseInt(mydata.totalInsta
                ), Color.MAGENTA));
122         pieDataInsta.add(new SliceValue(Integer.parseInt(mydata.
                totalAcessos), Color.GRAY));
123
124             PieChartData pieChartDataInsta = new PieChartData(pieDataInsta
                    );
125             pieChartDataInsta.setHasCenterCircle(true).setCenterText1("
                    Instagram");
126             pieChartDataInsta.setCenterText1FontSize(11);
127
128             pieChartViewInsta.setPieChartData(pieChartDataInsta);
129         }
```

```
130
131
132
133         //
                ____________________________________________________________________


134
135         PieChartView pieChartViewYou = findViewById(R.id.chartYoutube);
136         List<SliceValue> pieDataYou = new ArrayList<>();
137
138         if((mydata.totalAcessos != "") && (mydata.totalYou != "")) {
139             pieDataYou.add(new SliceValue(Integer.parseInt(mydata.totalYou
                    ), Color.RED));
140             pieDataYou.add(new SliceValue(Integer.parseInt(mydata.
                    totalAcessos), Color.GRAY));
141
142             PieChartData pieChartDataYou = new PieChartData(pieDataYou);
143             pieChartDataYou.setHasCenterCircle(true).setCenterText1("
                    YouTube");
144             pieChartDataYou.setCenterText1FontSize(11);
145
146             pieChartViewYou.setPieChartData(pieChartDataYou);
147         }
148
149
150         //
                ____________________________________________________________________


151
152         PieChartView pieChartViewTwitter = findViewById(R.id.chartTwitter)
                ;
153         List<SliceValue> pieDataTwitter = new ArrayList<>();
154
155         if((mydata.totalAcessos != "") && (mydata.totalTwitter != "")) {
156         pieDataTwitter.add(new SliceValue(Integer.parseInt(mydata.
                totalTwitter), Color.CYAN));
157         pieDataTwitter.add(new SliceValue(Integer.parseInt(mydata.
                totalAcessos), Color.GRAY));
158             PieChartData pieChartDataTwitter = new PieChartData(
```

```
                pieDataTwitter);
159             pieChartDataTwitter.setHasCenterCircle(true).setCenterText1("
                    Twitter");
160             pieChartDataTwitter.setCenterText1FontSize(11);
161
162             pieChartViewTwitter.setPieChartData(pieChartDataTwitter);
163         }
164
165
166
167
168     }
169
170
171     @Override
172     public boolean onCreateOptionsMenu(Menu menu) {
173         // Inflate the menu; this adds items to the action bar if it is
                present.
174         getMenuInflater().inflate(R.menu.menu_main_trial, menu);
175
176         // MenuItem iten = (MenuItem) findViewById(R.id.action_user);
177
178      //    iten.setIcon(ContextCompat.getDrawable(this ,R.drawable.leftarrow
            ));
179
180         return true;
181     }
182
183     @Override
184     public boolean onOptionsItemSelected(MenuItem item) { switch(item.
            getItemId()) {
185
186
187         case R.id.action_user:
188
189             Calendar calendar = Calendar.getInstance();
190             yearSelected = calendar.get(Calendar.YEAR);
191             monthSelected = calendar.get(Calendar.MONTH);
192
```

```
193             // navigationView = (BottomNavigationView) findViewById(R.id.
                    navigationView);
194             //  navigationView.setOnNavigationItemSelectedListener(this);
195
196               MonthYearPickerDialog pickerDialog = new MonthYearPickerDialog
                      ();
197               pickerDialog.setListener(new DatePickerDialog.
                      OnDateSetListener() {
198                   @Override
199                   public void onDateSet(DatePicker datePicker, int year, int
                          month, int i2) {
200                       Toast.makeText(SocialMediaActivity.this, year + "-" +
                              month, Toast.LENGTH_SHORT).show();
201                       yearSelected = year;
202                       monthSelected = month;
203
204                       GetData mydata = new GetData();
205                       mydata.doInBackground(String.valueOf(year), String.
                              valueOf(month), sessionId, 5);
206
207                       ResumoIP.setTextSize(15);
208                       ResumoIP.setTypeface(ResumoIP.getTypeface(), Typeface.
                              BOLD);
209                       ResumoIP.setText("Resumo de Acessos - IP: "+ sessionId
                              );
210
211                       TotalAcessosRede.setTextSize(12);
212                       TotalAcessosRede.setText("Quantidade de Acessos(m s):
                               "+mydata.totalAcessos);
213
214                       qtdYoutube.setTextSize(12);
215                       qtdYoutube.setText("Acessos YouTube(m s): "+mydata.
                              totalYou);
216
217                       qtdFace.setTextSize(12);
218                       qtdFace.setText("Acessos Facebook(m s): "+mydata.
                              totalFace);
219
220                       qtdTwitter.setTextSize(12);
```

```
221                 qtdTwitter.setText("Acessos Twitter(m s): "+mydata.
                        totalTwitter);
222
223                 qtdInsta.setTextSize(12);
224                 qtdInsta.setText("Acessos Instagram(m s): "+mydata.
                        totalInsta);
225
226
227                 PieChartView pieChartViewFace = findViewById(R.id.
                        chartFace);
228                 List<SliceValue> pieDataFace = new ArrayList<>();
229
230                 if((mydata.totalAcessos != "") && (mydata.totalFace !=
                         "")) {
231
232
233                     pieDataFace.add(new SliceValue(Integer.parseInt(
                            mydata.totalFace), Color.BLUE));
234                     pieDataFace.add(new SliceValue(Integer.parseInt(
                            mydata.totalAcessos), Color.GRAY));
235
236                     PieChartData pieChartDataFace = new PieChartData(
                            pieDataFace);
237                     pieChartDataFace.setHasCenterCircle(true).
                            setCenterText1("Facebook");
238                     pieChartDataFace.setCenterText1FontSize(11);
239
240                     pieChartViewFace.setPieChartData(pieChartDataFace)
                            ;
241                 }
242                 // TV2.setText("Face: " + mydata.totalFace + "  Total:
                        " + mydata.totalAcessos + " Session: " + sessionId +
                         " data: " + mydata.ConnectionResult);
243
244
245                 //
                        ____________________________________________________
246                 PieChartView pieChartViewInsta = findViewById(R.id.
```

```
                chartInsta);
247             List<SliceValue> pieDataInsta = new ArrayList<>();
248
249             if((mydata.totalAcessos != "") && (mydata.totalInsta
                    != "")) {
250
251                 pieDataInsta.add(new SliceValue(Integer.parseInt(
                        mydata.totalInsta), Color.MAGENTA));
252                 pieDataInsta.add(new SliceValue(Integer.parseInt(
                        mydata.totalAcessos), Color.GRAY));
253
254                 PieChartData pieChartDataInsta = new PieChartData(
                        pieDataInsta);
255                 pieChartDataInsta.setHasCenterCircle(true).
                        setCenterText1("Instagram");
256                 pieChartDataInsta.setCenterText1FontSize(11);
257
258                 pieChartViewInsta.setPieChartData(
                        pieChartDataInsta);
259             }
260
261             //
                    ____________________________________________________________


262
263             PieChartView pieChartViewYou = findViewById(R.id.
                    chartYoutube);
264             List<SliceValue> pieDataYou = new ArrayList<>();
265
266             if((mydata.totalAcessos != "") && (mydata.totalYou !=
                    "")) {
267                 pieDataYou.add(new SliceValue(Integer.parseInt(
                        mydata.totalYou), Color.RED));
268                 pieDataYou.add(new SliceValue(Integer.parseInt(
                        mydata.totalAcessos), Color.GRAY));
269
270                 PieChartData pieChartDataYou = new PieChartData(
                        pieDataYou);
271                 pieChartDataYou.setHasCenterCircle(true).
```

```
                setCenterText1("YouTube");
272             pieChartDataYou.setCenterText1FontSize(11);
273
274             pieChartViewYou.setPieChartData(pieChartDataYou);
275         }
276
277
278
279         //
                ____________________________________________________________
280
281         PieChartView pieChartViewTwitter = findViewById(R.id.
                chartTwitter);
282         List<SliceValue> pieDataTwitter = new ArrayList<>();
283
284         if((mydata.totalAcessos != "") && (mydata.totalTwitter
                != "")) {
285             pieDataTwitter.add(new SliceValue(Integer.parseInt
                    (mydata.totalTwitter), Color.CYAN));
286             pieDataTwitter.add(new SliceValue(Integer.parseInt
                    (mydata.totalAcessos), Color.GRAY));
287
288             PieChartData pieChartDataTwitter = new
                    PieChartData(pieDataTwitter);
289             pieChartDataTwitter.setHasCenterCircle(true).
                    setCenterText1("Twitter");
290             pieChartDataTwitter.setCenterText1FontSize(11);
291
292             pieChartViewTwitter.setPieChartData(
                    pieChartDataTwitter);
293         }
294
295
296
297
298
299
300
```

```
301                     }
302                 });
303                 pickerDialog.show(getSupportFragmentManager(), "
                        MonthYearPickerDialog");
304
305
306
307
308                 return(true);
309
310             case R.id.action_user_trial:
311
312                 Intent intent = new Intent(SocialMediaActivity.this,
                        MimeTypeActivity.class);
313                 intent.putExtra("EXTRA_SESSION_ID", sessionId);
314                 startActivity(intent);
315
316                 return(true);
317         }
318             return(super.onOptionsItemSelected(item));
319         }
320 }
```

C.9 *ACTIVITY* DE TIPOS DE DADOS TRAFEGADOS FONTE: DOS AUTORES

```
1   package    parallelcodes.mysqlapp;
2
3  import android.app.DatePickerDialog;
4  import android.content.Intent;
5  import android.graphics.Color;
6  import android.graphics.Typeface;
7  import android.os.StrictMode;
8  import android.support.v7.app.AppCompatActivity;
9  import android.os.Bundle;
10 import android.view.Menu;
11 import android.view.MenuItem;
12 import android.widget.DatePicker;
13 import android.widget.Toast;
14
15 import java.util.ArrayList;
```

```
16 import java.util.Calendar;
17 import java.util.List;
18
19 import de.codecrafters.tableview.TableView;
20 import de.codecrafters.tableview.model.TableColumnWeightModel;
21 import de.codecrafters.tableview.toolkit.SimpleTableDataAdapter;
22 import de.codecrafters.tableview.toolkit.SimpleTableHeaderAdapter;
23 import lecho.lib.hellocharts.model.PieChartData;
24 import lecho.lib.hellocharts.model.SliceValue;
25 import lecho.lib.hellocharts.view.PieChartView;
26
27 public class MimeTypeActivity extends AppCompatActivity {
28
29     public String sessionId = "";
30
31     int yearSelected = 2019;
32     int monthSelected = 11;
33
34     String[][] DATA;
35
36     private static final String[] TABLE_HEADERS = { "Mime Type", "Volume (
           MB)" };
37     TableView tableView;
38
39     @Override
40     protected void onCreate(Bundle savedInstanceState) {
41         super.onCreate(savedInstanceState);
42         setContentView(R.layout.activity_mime_type);
43         if (android.os.Build.VERSION.SDK_INT > 9) {
44             StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.
                   Builder().permitAll().build();
45             StrictMode.setThreadPolicy(policy);
46         }
47
48         Calendar calendar = Calendar.getInstance();
49         yearSelected = calendar.get(Calendar.YEAR);
50         monthSelected = calendar.get(Calendar.MONTH)+1;
51
52         sessionId = getIntent().getStringExtra("EXTRA_SESSION_ID");
```

```
53
54          getSupportActionBar().setTitle("Relat rio de Tipos - " +
                sessionId);
55
56
57          tableView = (TableView) findViewById(R.id.tableViewMime);
58          tableView.setHeaderAdapter(new SimpleTableHeaderAdapter(this,
                TABLE_HEADERS));
59
60
61          TableColumnWeightModel columnModel = new TableColumnWeightModel(2)
                ;
62          tableView.setColumnModel(columnModel);
63
64          //Query
65          GetData mydataMime = new GetData();
66          mydataMime.doInBackground(String.valueOf(yearSelected), String.
                valueOf(monthSelected), sessionId, 7);
67
68          DATA = new String[mydataMime.typeMime.size()][2];
69         // String[][] DATA = new String[2][2];
70         //          DATA[0][0] = "Oi";
71          int indexBarra = -1;
72          if (mydataMime.typeMime.size() > 0) {
73
74              for (int i=0;i<mydataMime.typeMime.size();i++) {
75                  DATA[i][0]=mydataMime.typeMime.get(i).toString();
76                      if(mydataMime.typeMime.get(i).toString().contains("-")
                            ){
77                          indexBarra = i;
78                      }
79              }
80              for (int i=0;i<mydataMime.bytesMime.size();i++) {
81                  DATA[i][1]=mydataMime.bytesMime.get(i).toString();
82              }
83          }
84
85          tableView.setDataAdapter(new SimpleTableDataAdapter(this, DATA));
86
```

```
87          ArrayList<String> typeLegend = new ArrayList<String>();
88          ArrayList<Double> bytePie = new ArrayList<Double>();
89
90          typeLegend = mydataMime.typeMime;
91          bytePie=mydataMime.bytesMime;
92          if(indexBarra !=-1 ){
93              typeLegend.remove(indexBarra);
94              bytePie.remove(indexBarra);
95          }
96
97
98
99          Integer[] VectColor = {Color.BLACK, Color.BLUE, Color.CYAN, Color.
                DKGRAY, Color.GREEN, Color.MAGENTA, Color.LTGRAY, Color.RED,
                Color.WHITE};
100
101
102         PieChartView pieChartView = findViewById(R.id.chartMime);
103         List<SliceValue> pieData = new ArrayList<>();
104
105         if((mydataMime.typeMime.size() > 0) && (mydataMime.bytesMime.size
                ()> 0)) {
106
107             for (int i = 0; i< mydataMime.bytesMime.size();i++){
108                 pieData.add(new SliceValue(Float.valueOf(String.valueOf(
                        bytePie.get(i))), VectColor[i]).setLabel(typeLegend.get
                        (i)));
109
110             }
111
112             PieChartData pieChartData = new PieChartData(pieData);
113             pieChartData.setHasLabels(true).setValueLabelTextSize(14);
114             pieChartData.setHasCenterCircle(true).setCenterText1("Tipos
                    Trafegados").setCenterText1FontSize(14).setCenterText2("
                    exceto '-' \n "+ monthSelected+"-"+yearSelected).
                    setCenterText2FontSize(9);
115             pieChartView.setPieChartData(pieChartData);
116
117
```

```
118         }
119
120
121
122     }
123
124     @Override
125     public boolean onCreateOptionsMenu(Menu menu) {
126         // Inflate the menu; this adds items to the action bar if it is
                present.
127         getMenuInflater().inflate(R.menu.menu_main_dual, menu);
128
129         // MenuItem iten = (MenuItem) findViewById(R.id.action_user);
130
131         //    iten.setIcon(ContextCompat.getDrawable(this ,R.drawable.
                leftarrow));
132
133         return true;
134     }
135
136     @Override
137     public boolean onOptionsItemSelected(MenuItem item) { switch(item.
            getItemId()) {
138
139         case R.id.action_user:
140
141             Calendar calendar = Calendar.getInstance();
142             yearSelected = calendar.get(Calendar.YEAR);
143             monthSelected = calendar.get(Calendar.MONTH);
144
145
146
147             MonthYearPickerDialog pickerDialog = new MonthYearPickerDialog
                    ();
148             pickerDialog.setListener(new DatePickerDialog.
                    OnDateSetListener() {
149                 @Override
150                 public void onDateSet(DatePicker datePicker, int year, int
                        month, int i2) {
```

```
151                 Toast.makeText(MimeTypeActivity.this, year + "-" +
                        month, Toast.LENGTH_SHORT).show();
152                 yearSelected = year;
153                 monthSelected = month;
154
155                 //Query
156                 GetData mydataMime = new GetData();
157                 mydataMime.doInBackground(String.valueOf(yearSelected)
                        , String.valueOf(monthSelected), sessionId, 7);
158
159                 DATA = new String[mydataMime.typeMime.size()][2];
160
161                 int indexBarra = -1;
162                 if (mydataMime.typeMime.size() > 0) {
163
164                     for (int i=0;i<mydataMime.typeMime.size();i++) {
165                         DATA[i][0]=mydataMime.typeMime.get(i).toString
                                ();
166                         if(mydataMime.typeMime.get(i).toString().
                                contains("-")){
167                             indexBarra = i;
168                         }
169                     }
170                     for (int i=0;i<mydataMime.bytesMime.size();i++) {
171                         DATA[i][1]=mydataMime.bytesMime.get(i).
                                toString();
172                     }
173                 }
174
175                 tableView.setDataAdapter(new SimpleTableDataAdapter(
                        getApplicationContext(), DATA));
176                 //tableView.setDataAdapter(new SimpleTableDataAdapter(
                        this, DATA));
177
178                 ArrayList<String> typeLegend = new ArrayList<String>()
                        ;
179                 ArrayList<Double> bytePie = new ArrayList<Double>();
180
181                 typeLegend = mydataMime.typeMime;
```

```
182                 bytePie=mydataMime.bytesMime;
183                 if(indexBarra !=-1 ){
184                     typeLegend.remove(indexBarra);
185                     bytePie.remove(indexBarra);
186                 }
187
188
189
190                 Integer[] VectColor = {Color.BLACK, Color.BLUE, Color.
                        CYAN, Color.DKGRAY, Color.GREEN, Color.MAGENTA,
                        Color.LTGRAY, Color.RED, Color.WHITE};
191
192
193                 PieChartView pieChartView = findViewById(R.id.
                        chartMime);
194                 List<SliceValue> pieData = new ArrayList<>();
195
196                 if((mydataMime.typeMime.size() > 0) && (mydataMime.
                        bytesMime.size()> 0)) {
197
198                     for (int i = 0; i< mydataMime.bytesMime.size();i
                            ++){
199                         pieData.add(new SliceValue(Float.valueOf(
                                String.valueOf(bytePie.get(i))), VectColor[
                                i]).setLabel(typeLegend.get(i)));
200
201                     }
202
203
204                     PieChartData pieChartData = new PieChartData(
                            pieData);
205                     pieChartData.setHasLabels(true).
                            setValueLabelTextSize(14);
206                     pieChartData.setHasCenterCircle(true).
                            setCenterText1("Tipos Trafegados").
                            setCenterText1FontSize(14).setCenterText2("
                            exceto '-' \n "+ monthSelected+"-"+yearSelected
                            ).setCenterText2FontSize(9);
207                     pieChartView.setPieChartData(pieChartData);
```

```
208
209
210
211                         }
212
213
214
215                     }
216                 });
217                 pickerDialog.show(getSupportFragmentManager(), "
                        MonthYearPickerDialog");
218
219
220
221
222                 return(true);
223
224         }
225             return(super.onOptionsItemSelected(item));
226         }
227 }
```