MÍRIAM FÉLIX LEMES DA SILVA

YARA CRISTINA ALVES RIBEIRO

APLICAÇÃO E ESTUDO DO PROTOCOLO SNMP

INCONFIDENTES/MG

2017

MÍRIAM FÉLIX LEMES DA SILVA

YARA CRISTINA ALVES RIBEIRO

APLICAÇÃO E ESTUDO DO PROTOCOLO SNMP

Trabalho de Conclusão de Curso apresentado como pré-requisito de conclusão do curso de Graduação de Tecnologia em Redes de Computadores no Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais - Campus Inconfidentes, para obtenção do título de Tecnólogo em Redes de Computadores.

Orientador: Prof.º Kleber Marcelo da Silva Rezende.

INCONFIDENTES/MG

2017

MÍRIAM FÉLIX LEMES DA SILVA
YARA CRISTINA ALVES RIBEIRO

APLICAÇÃO E ESTUDO DO PROTOCOLO SNMP

Data de aprovação: ___ de _____20__

_____

Orientador: Kleber Marcelo da Silva Rezende
(IFSULDEMINAS – Campus Inconfidentes)

_____

Prof.: Alessandro de Castro Borges
(IFSULDEMINAS – Campus Inconfidentes)

_____

Prof.: Ivan Paulino Pereira
(IFSULDEMINAS – Campus Inconfidentes)

Dedicamos este trabalho a João Anésio Ribeiro – ex-aluno do curso de Tecnologia em Redes de Computadores, in memoriam. Obrigada por nos mostrar que não existem limites.

AGRADECIMENTOS

Agradeço primeiramente a minha família pela paciência e compreensão no período de elaboração do projeto. Também aos professores Bruno Rezende, pelo apoio inicial e ao professor Kleber Rezende por nos orientar e acreditar no nosso empenho.

Também a galera do PyBeginners pelas dicas, apoio e incentivo, em especial ao Anderson Soares, que acabou se tornando um grande amigo.

E ao coordenador do curso pela oportunidade.

Miriam Félix Lemes Da Silva

Primeiramente agradeço a Deus por sempre me dar forças para superar dificuldades e sempre me presentear com muitas conquistas. Aos meus pais João Anésio e Lucimar, ao meu noivo Bruno e ao meu filho Igor por sempre me ajudar nos momentos difíceis me mostrando sempre a importância de continuar e nunca desistir ou desanimar, sou muito grata por toda a minha família que sempre me deu todo o apoio e me ajudou com muito amor e carinho. E agradeço ao professor Kleber orientador do nosso trabalho aos colegas e todos outros professores que sempre esteve presente para atender todas as necessidades de forma paciente.

Yara Cristina Alves Ribeiro

Bonito é melhor que feio.
Explícito é melhor que implícito.
Simples é melhor que complexo.
Complexo é melhor que complicado.
Plano é melhor do que aninhado.
Esparso é melhor que denso.
Legibilidade conta.
Casos especiais não são especiais o bastante para quebrar as regras.
Embora a simplicidade supere o purismo.
Erros nunca deveriam passar silenciosamente.
A menos que explicitamente silenciados.
Ao encara a ambiguidade, recuse a tentação de adivinhar.
Deveria haver uma - e preferencialmente apenas uma – maneira óbvia de se fazer isso.
Embora aquela maneira possa não ser óbvia à primeira vista se você não for holandês.
Agora é melhor que nunca.
Embora nunca, seja muitas vezes melhor que pra já.
Se a implementação é difícil de explicar, é uma má ideia.
Se a implementação é fácil de explicar, pode ser uma boa ideia.
Namespaces são uma grande ideia — vamos fazer mais deles!
The Zen of Python - Tim Peters

RESUMO

O presente projeto se propõe a desenvolver uma ferramenta que possibilite o gerenciamento de redes, demonstrar a importância do gerenciamento, descrever as funcionalidades do protocolo SNMP e suas possibilidades. Embora existam vários dispositivos no mercado capazes de fazer esse tipo de tarefa, é importante explorar novas ferramentas e o desenvolvimento de soluções para gerência de redes. A partir dessa ideia, surgiu o interesse de se explorar a linguagem Python por ser uma linguagem em constante crescimento e com ferramentas que possibilitam a criação de aplicativos eficazes para gerência. Serão apresentados conceitos sobre o protocolo SNMP, sobre a linguagem de programação Python e sobre as bibliotecas utilizadas, derivadas do Python. O resultado final foi o aprofundamento no conhecimento do protocolo SNMP, bem como da linguagem de programação utilizada, e a criação de uma ferramenta para gerenciamento utilizando o Python e o protocolo SNMP.

Palavras-chave: SNMP, Python, PySNMP, Gerenciamento, Redes.

ABSTRACT

The present project proposes to develop a tool that allows the management of networks, demonstrate the importance of management, describe the functionalities of the SNMP protocol and its possibilities. While there are several devices on the market that are capable of doing this, it is important to explore new tools and develop solutions for network management. From this idea came the interest of exploring the Python language because it is a language that is constantly growing and with tools that allow the creation of effective applications for management. It will present concepts about the SNMP protocol, the Python programming language and the libraries used, derived from Python. The final result was the deepening of knowledge of the SNMP protocol, as well as the programming language used, and the creation of a management tool using Python and the SNMP protocol.

Keywords: SNMP, Python, PySNMP, management, network.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIAÇÕES

ASN.1	-	*Autonomous System Number One*
GPL	-	*General Public License*
GPU	-	*Graphics Processing Unit*
HLAPI	-	*High Level API*
IP	-	*Internet Protocol*
IETF	-	*Internet Engineering Task Force*
MIB	-	*Management Information Base*
MIT	-	*Massachusetts Institute of Technology.*
NMA	-	*Network Manage Application*
NME	-	*Network Management Entity*
OID	-	*Object Identifier*
OG	-	Objeto Gerenciável
ORM	-	*Object Relational Mapper*
PSF	-	*Python Software Foundation*
PDU	-	*Protocol Data Unit*
RFC	-	*Request for Comments*
SNMP	-	*Simple Network Management Protocol*
SMTP	-	*Simple Mail Transfer Protocol*
SMI	-	*Structure of Management Information*
SO	-	Sistema Operacional
SQL	-	*Structured Query Language*
TCP	-	*Transmission Control Protocol*
UDP	-	*User Datagram Protocol*

SUMÁRIO

1.INTRODUÇÃO

A quantidade de dispositivos interligados, a variedade desses objetos e a capacidade de escalabilidade das redes, fez com que a tarefa de gerenciamento passasse de algo trivial para uma atividade complexa.

Em uma rede de computadores comum hoje em dia diversos dispositivos são conectados e desconectados instantaneamente, e a capacidade de expansão dessas redes varia na mesma velocidade. A partir dessa diversidade de dispositivos, de conexões possíveis e das mudanças constantes em uma rede surge, quase que abruptamente, a necessidade de gerenciar e monitorar equipamentos de redes.

Para um gerenciamento de rede eficaz, que possa evitar futuros problemas, é indispensável a utilização de alguma ferramenta de gerenciamento, um bom planejamento de rede e um protocolo capaz de se comunicar com as estações gerenciadas.

Para isso pode-se considerar o protocolo SNMP (*Simple Network Management Protocol*) que, segundo Kurose e Ross (2010), é o protocolo mais utilizado em quase todos os dispositivos, trabalhando com grande desempenho e eficiência, podendo ser suportado mundialmente por todos os tipos de hardware por ser um protocolo padrão da internet.

Este trabalho busca criar um aplicativo capaz de receber dados de uma estação de trabalho gerenciada, apresentar os pormenores relacionados ao protocolo SNMP, a fim de realizar um gerenciamento de rede eficaz e também busca explorar a linguagem de programação Python e suas possibilidades por se tratar de uma linguagem que possui uma

sintaxe bastante clara e intuitiva, favorecendo uma melhor flexibilidade do código fonte, tornando a linguagem mais fácil e produtiva.

1.1. OBJETIVO GERAL

Desenvolver um software, a partir do estudo sobre o protocolo SNMP, para monitorar dispositivos de rede através do endereço IP encaminhando mensagens "*get"* aos objetos gerenciados da rede, para se obter como resultado as informações solicitadas.

1.2. OBJETIVO ESPECÍFICO

- Apresentar a evolução do protocolo SNMP.
- Apresentar as principais funções e características do protocolo SNMP.
- Apresentar a estrutura do software e descrever seu funcionamento.
- Desenvolver um aplicativo capaz de auxiliar no gerenciamento.
- Apresentar os resultados dos testes realizados.
- Demonstrar pontos positivos e negativos do trabalho.

1.3. RELEVÂNCIA DO TRABALHO

Segundo Stallings (2005), as redes e os sistemas distribuídos são de grande importância e crescente em todos os tipos de empresa. Uma grande rede não pode ser gerenciada apenas pelo esforço humano, devido à complexidade de tais redes, tornando-se indispensável a criação de soluções para o seu gerenciamento.

> "Para existir um gerenciamento que consiga suprir as necessidades do gerente e da instituição a qual essa rede pertença, são necessárias ferramentas para auxiliar essa monitoração. No entanto é sempre interessante que se desenvolvam e estudem novas possibilidades de ferramentas utilizadas para gerência, buscando-se assim acompanhar o crescimento das redes, mantendo a eficiência no gerenciamento das mesmas." (Stallings,2005).

Como afirmado por Kurose e Ross (2010), com a internet pública, as intranets cresceram e se transformaram em grandes estruturas globais. Cresceram também a necessidade de se gerenciar mais sistematicamente a enorme quantidade de componentes de hardware e software dentro dessas redes.

1.4. PROPOSTA

Este trabalho se propõe a desenvolver uma ferramenta, com o intuito de contribuir para o estudo e crescimento do desenvolvimento de aplicativos para o gerenciamento de redes.

> "Com os avanços tecnológicos e novos equipamentos e tecnologias que vieram melhorar as condições de trabalho, também começaram a surgir novos problemas com o grande crescimento e complexidade das redes. Então a necessidade de gerenciar e monitorar tanto equipamentos e informações são importantes para otimização de uma rede." (Fraga, 2009).

Neste trabalho foram descritos os requisitos da gerência de redes, abordando detalhes e funcionamento do protocolo SNMP, um protocolo que é amplamente utilizado para o gerenciamento de redes e também foram feitos estudos sobre a linguagem de programação Python e suas ferramentas como o PySNMP e Kivy utilizadas para a criação do software.

2. GERENCIAMENTO DE REDES

Gerenciamento de redes é a capacidade de monitorar as redes e traçar um perfil de utilização da rede pelos hosts, a fim de detectar atividades anormais ou medir a capacidade da rede.

> “Gerenciamento de rede inclui o oferecimento, a integração e a coordenação de elementos de hardware, software e humanos, para monitorar, testar, consultar, configurar, analisar, avaliar e controlar os recursos da rede, e de elementos, para satisfazer às exigências operacionais, de desempenho e de qualidade de serviço em tempo real a um custo razoável”. (Saydam 1996, apud Kurose e Ross, 2010)

A necessidade de gerenciamento surgiu com o aumento da complexidade das redes e o aumento de dispositivos que podem fazer parte dessa rede, assim como a capacidade de escalabilidade das mesmas.

> "As informações que circulam em uma rede de computadores devem ser transportadas de modo confiável e rápido. Para que isso aconteça é importante que os dados sejam monitorados de maneira que os problemas que porventura possam existir sejam detectados rapidamente e sejam solucionados eficientemente. Uma rede sem mecanismos de gerência pode apresentar problemas que irão afetar o tráfego dos dados, bem como sua integridade, como problemas de congestionamento do tráfego, recursos mal utilizados, recursos sobrecarregados, problemas com segurança entre outros“ (Pinheiro, 2002).

Segundo Stallings (2005) e Kurose e Ross (2010) existem cinco áreas de gerenciamento de redes, propostas pela ISO (*International Organization for Standardization*), que fornecem uma maneira útil de organizar a análise dos requisitos. São as seguintes áreas citadas nos tópicos posteriores.

• **Gerenciamento de falhas:**

O gerenciamento de falhas deve detectar e corrigir operações anormais do ambiente.

> "Para manter o correto funcionamento de uma rede complexa, é necessário ter cuidado para que os sistemas como um todo, e cada componente individualmente, esteja na ordem de funcionamento correta. " (Stallings, 2005)

Ainda com base no estudo de Stallings (2005), falhas devem ser distinguidas de erros, uma falha é reconhecida como um funcionamento anormal da rede, ou excesso de erros.

Após detectada a falha, e corrigida, o gerente deve garantir que os problemas não retornarão, isso é chamado de rastreamento e controle de problemas.

• **Gerenciamento de contabilidade:**

Segundo Stallings (2005), gerenciamento de contabilização é uma área responsável por estabelecer parâmetros para a utilização dos recursos da rede. Que permite que o administrador da rede especifique, registre e controle o acesso de usuários e dispositivos garantindo desempenho na rede e aos recursos de rede. Quotas de utilização, cobrança por utilização, uso ineficientes da rede e evitar que usuários abusem de acessos privilegiados a recursos fazem parte do gerenciamento de contabilização.

• **Gerenciamento de configuração e de nome:**

Como citado em Stallings (2005), componentes individuais da rede podem ser utilizados para tarefas distintas, uma vez definidas como este componente será utilizado o gerente de configuração estabelece o software apropriado, conjunto de atributos e valores para a utilização desse componente.

> "O gerenciamento de configuração permite que um administrador de rede saiba quais dispositivos fazem parte da rede administrada e quais são suas configurações de hardware e software. O [RFC 3139] oferece uma visão geral de gerenciamento e requisitos de configuração para redes IP ". (Stallings, 2005)

• **Gerenciamento de desempenho:**

Como descrito em Stallings (2005), o gerenciamento de desempenho é dividido entre monitoramento e controle, o primeiro acompanha as atividades na rede e o segundo permite que se façam ajuste para melhorar o desempenho da rede.

> A meta de gerenciamento de desempenho é quantificar, medir, informar, analisar e controlar o desempenho (por exemplo, utilização e vazão) de diferentes componentes da rede. (Stallings,2005)

Ainda em Stallings (2005) foram descritas algumas questões que fazem parte do gerenciamento de desempenho, como por exemplo:

•. Qual o nível da utilização de capacidade?

•. Existe tráfego excessivo?

• A vazão caiu a níveis inaceitáveis?

•. Existem gargalos?

Para lidar com essas questões Stallings (2005) propõe determinar um valor inicial a ser monitorado para avaliar os níveis de desempenho.

Segundo Kurose e Ross (2009), padrões do protocolo SNMP [RFC 3410] desempenham um papel fundamental no gerenciamento de desempenho da Internet.

• **Gerenciamento de segurança:**

Como se pode observar em Stallings (2005) no gerenciamento de segurança se concentra em controlar os recursos da rede na geração de distribuição e armazenamento de chaves de criptografia. Os logs são de grande importância para o gerenciamento de segurança que busca, gerenciar, armazenar e analisar esses recursos, bem como ativar e desativá-los.

2.1. SISTEMAS DE GERENCIAMENTO DE REDES

Como citado em Stallings (2005), uma arquitetura de sistema de gerenciamento de rede é composta por uma grande variedade de ferramentas para monitoramento e controle de rede. Integrando uma interface de operador com um conjunto de comandos para realizar as tarefas de gerenciamento, e uma quantidade mínima de equipamento para gerenciar a rede.

> "Um sistema de gerenciamento de rede consiste em adições incrementais de hardware e software implementadas entre os componentes de rede existentes. O software usado para realizar as tarefas de gerenciamento reside nos computadores hosts e nos processadores de comunicações". (Stallings,2005)

Segundo Stallings (2005), em um sistema de gerenciamento cada host possui um software que atua com gerenciamento, esse software está descrito como NME (*Network Management Entity* - Unidade de Gerenciamento de Rede). Dentro da rede pelo menos um host é o host de controle, esse é o gerente. Para fazer o gerenciamento o gerente dispõe de

uma aplicação de gerenciamento de rede, descrita como NMA (*Network Management Application* – Aplicação de gerenciamento de rede), que possibilita o gerenciamento do gerente para os outros hosts, os agentes.

Todos os membros da rede podem utilizar uma aplicação que auxilie nessa rotina. Podemos observar na Figura 1, um esquema de arquitetura de rede como descrito.

NME (*Network Management Entity* - Unidade de Gerenciamento de Rede)

NMA (*Network Management Application* – Aplicação de gerenciamento de rede)

Apl. - Aplicação utilizada para gerenciamento.

SO – Sistema operacional

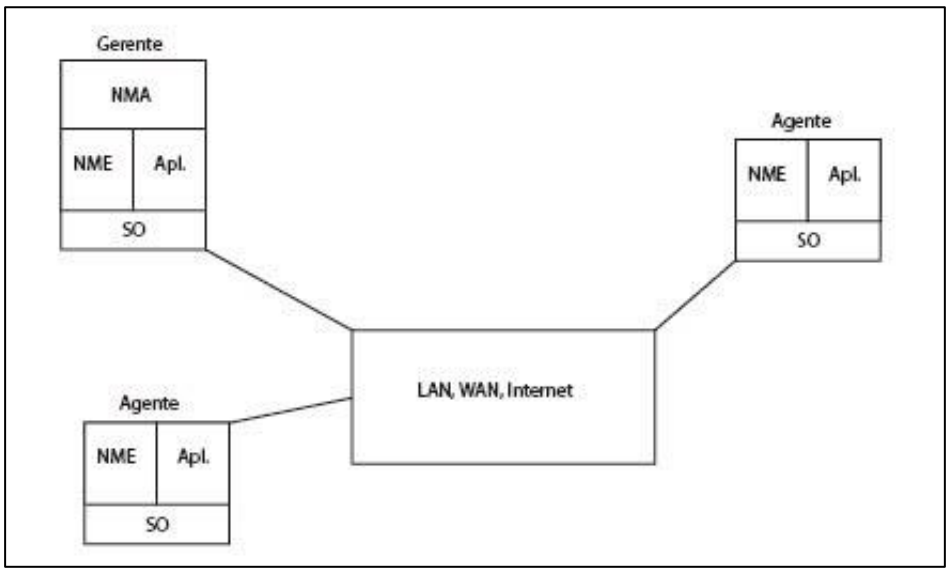

Figura 1 -Elementos de gerenciamento de rede. Fonte: baseado em (Stallings,2005)

Cada componente da rede é considerado um objeto e cada objeto possui um identificador, OID (*Object Identifier* – Identificador de Objeto).

> Os OIDs são identificadores de objetos gerenciáveis, que formam uma árvore hierárquica. O OID é composto por uma sequência de números que identifica a posição do objeto na árvore da MIB. Essa estrutura não tem limites e, de acordo com a necessidade, pode ser atualizada e expandida. (Kurose e Ross, 2010)

Nessa árvore pode-se identificar a posição de cada objeto. A Figura 2, mostra a sequência de OIDs na árvore.

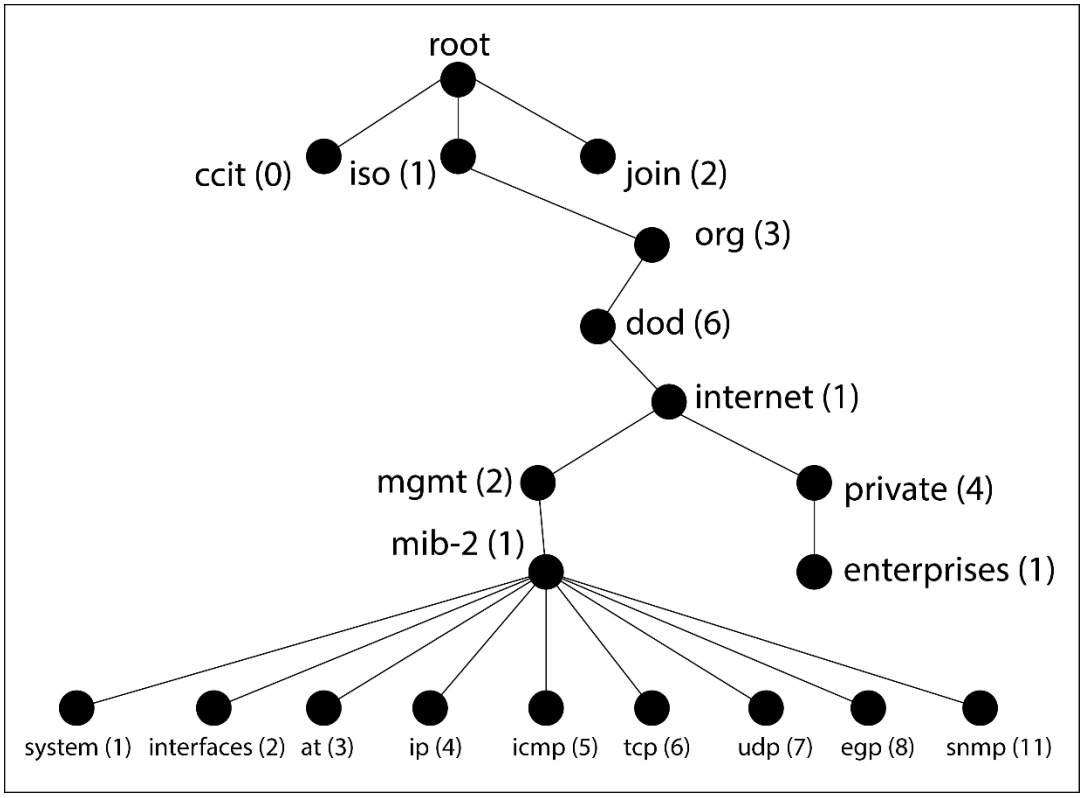

Figura 2- Árvore de identificadores de objetos ASN.1. Fonte: baseado em Kurose e Ross (2010)

2.2. LINGUAGEM DE DEFINIÇÃO DE DADOS

Segundo Kurose e Ross (2010), a linguagem de definição de dados, SMI (*Struture of Management Information* – Estrutura de informação de gerenciamento), define os tipos de dados onde são especificados os objetos MIB, essa linguagem garante que não exista ambiguidade na definição dos dados. Essa linguagem está definida nas [RFC 2578, RFC 2579, RFC 2580].

Como citado em Kurose e Ross (2010), na RFC 2578 a SMI, foi baseada na linguagem ASN.1 (*Abstract Syntax Notation One* – Notação de sintaxe abstrata 1) e possui 11 tipos de dados básicos definidos nessa RFC como mostrados na Tabela 1.

Tabela 1 - Tipos básicos da SMI.

TIPO DE DADO	**Descrição**
INTEGER	Número inteiro de 32 bits, como definido em ASN.1, com valor entre -2^{31} a 2^{31} -1.
INTEGER32	Número inteiro de 32 bits, com valor entre -2^{31} a 2^{31} -1.
UNSIGNED32	Número inteiro de 32 bits, sem sinal na faia de 0 a 2^{32} -1.

OCTET STRING	Cadeia de bytes de formato ASN.1 que representa dados binários arbitrários ou de texto de até 65.535 bytes de comprimento.
OBJECT IDENTIFIER	Formato ASN.1 atribuído administrativamente (nome estruturado)
ENDEREÇO IP	Endereço Internet de 32 bits, na ordem de bytes de rede.
COUNTER32	Contador de 32 bits que cresce de 0 a $2^{32} – 1$ e volta a 0.
COUNTER64	Contador 64 bits.
GAUGE32	Número inteiro de 32 bits que não excede 2^{32} -1 nem menor que 0
TIMETICKS	Tempo, medido em centésimos de segundo.
OPAQUE	Cadeia ASN.1 não interpretada, necessária para compatibilidade com versões anteriores.

Fonte: Stallings (2005)

3. PROTOCOLO SNMP

À medida em que as redes públicas e privadas passaram de pequenas redes, em que com apenas alguns "*pings*" pode-se localizar a origem do problema, para uma grande infraestrutura que possuem vários componentes, a necessidade de se desenvolver meios para o gerenciamento dessas redes também cresceu. Diante dessa necessidade, como abordado por Kurose e Ross (2010), um grupo de pesquisadores trabalhou na implementação de um protocolo que pudesse atender rapidamente esta demanda.

No início da década de 80 o protocolo *Simple Network Management Protocol* – SNMP começou a ser desenvolvido pelo *Internet Engineering Task Force* – IETF, com o objetivo de disponibilizar uma forma simples e prática de realizar o controle de equipamentos em uma rede de computadores.

> "Ao contrário do seu predecessor SGMP (Simple Gateway Management Protocol) que só servia para gerir roteadores, o SNMP pode ser usado para gerir uma maior diversidade de dispositivos" (Pedrosa, 2004).

Segundo Pinheiro (2002), o SNMP é um protocolo que teve sucesso por ser o primeiro protocolo de gerenciamento não proprietário, público e de fácil implementação, pode ser usado para auditoria e gerenciamento de redes, para monitorar o desempenho da rede, detectar acessos inadequados e falhas na rede. E pode ser utilizado tanto pelo gerente como pelo agente.

> "Assim, para manusear a grande quantidade de dados provenientes da ampla gama de tipos de equipamentos existentes nas redes, o uso de protocolos de gerenciamento padronizados específicos para o gerenciamento de redes se torna necessário. O protocolo SNMP (Simple Network Management Protocol) é um protocolo desenvolvido para este fim, permitindo o acesso às informações em ambientes com equipamentos de múltiplos fabricantes" (Guilhermo, 2008).

Ainda como observado em Kurose e Ross (2010), o SNMP trabalha na camada de aplicação. Ele obtém informações dos agentes da rede baseada na pilha de protocolos TCP/IP, que trabalha com o modelo gerente-agente. O SNMP faz o intermédio de mensagens entre o gerente e os objetos gerenciáveis. A SMI (*Structure Management Information*) define os objetos gerenciáveis, onde se encontram na pilha de gerenciamento e como se comportam.

3.1. VERSÕES

O protocolo SNMP sofreu várias modificações para acompanhar as mudanças e necessidades das redes atuais. Hoje, o SNMP possui três versões, para que fossem corrigidos erros e falhas na segurança. Ao longo desta evolução foram feitas algumas melhorias, que tornam esse protocolo de grande importância na gerência e monitoramento de redes, sendo o protocolo em maior uso atualmente, as versões do protocolo são as seguintes:

- SNMPv1 em 1988: revisão inicial, referenciada na RFC 1157, tipos de dados simples, como citado por Stallings (2005), foi desenvolvido para servir de modelo de gerenciamento para redes utilizando o protocolo de transmissão TCP/IP. O termo SNMP foi usado para definir a coleção de especificações para gerenciamento de rede. Este conceito de processo de gerenciamento controla o acesso a MIB, encaminha mensagens *GetRequest*, *GetNextRequest*, *SetRequest*, entre outras utilizando o protocolo UDP (*User Datagram Protocol*) sem conexão entre as estações de gerenciamento e seus agentes. Todas as requisições de gerenciamento são enviadas para o UDP porta 161 e UDP porta 162 as traps de agentes.

Segundo Fraga (2009), a primeira versão do protocolo tinha uma segurança muito limitada, o mecanismo baseado em comunidades não garantia uma transmissão segura de mensagens SNMP, sendo, portanto, falível a ataques de espionagem. Assim, qualquer intruso poderia observar uma mensagem, retirando a informação necessária para poder atacar uma rede, podendo modificar as suas configurações. De modo a evitar esses ataques, os fabricantes

restringiram a utilização de alguns comandos, o que acabou por limitar as funções de monitorização. Com o objetivo de superar estas falhas surgiu então o protocolo SNMPv2.

● SNMPv2 em 1993: melhorias em relação a versão v1, referenciadas nas RFCs 1901, 1902, 3416 e 3417. Segundo Stallings (2005), possibilita que agentes emitam mensagens TRAP para o gerente. Um dos principais avanços dessa versão são as mensagens GetBulkRequest, que permite que grande quantidade de informações de gerenciamento sejam encaminhadas ao gerente.

> "A SNMPv2 oferecia novas funcionalidades e uma maior eficiência que a versão original. Como referimos anteriormente introduziu novas operações, às quatro operações básicas. No sentido de permitir a transferência de grandes quantidades de informação, comunicação entre estações de gestão e normalização de mensagens de notificação foram criadas". (Fraga, 2009)

● SNMPv3 em 1999: referenciada nas RFCs 3410, 3411, 3412, 3413, 3414, 3415, 3416, 3417, 3584, 3826, 5343, e nas RFCs 5590, 5591, 5953, que prezam pela correção de segurança que é o grande marco dessa versão em relação às anteriores.

Segundo Stallings (2005) essa versão veio corrigir as deficiências da versão anterior, fornecendo três serviços importantes, autenticação, privacidade definidos pelo mecanismo de autenticação USM - (*User-BasedSegurityModel*) e controle de acesso definido peloVACM - (*View-Based Access Control Model*). O USM permite que gerenciadores e agentes criptografem mensagens utilizando DES (*Data Encryption Standard*).

> "O SNMPv3 surgiu como forma de combater este problema de segurança que existia nas versões anteriores. Esta versão não tem novas operações, e suporta todas as operações definidas para a SNMPv1 e SNMPv2. Nessa nova versão houve melhorias significativas na parte da segurança das informações, autenticação, maior privacidade, também pode ser feita configuração remota através de operadores SNMP entre outras modificações." (Fraga, 2009)

3.2. OPERAÇÕES DO PROTOCOLO SNMP

De acordo com Kurose e Ross (2010), o SNMP define mensagens PDUs (*Protocol Data Units*- Unidades de Dados de Protocolos), que são enviadas de um gerente a um agente e vice-versa para fazer requisições aos objetos MIB. São as PDUs citados na Tabela 2.

Tabela 2 - Mensagens SNMP.

Mensagem	Descrição
GetRequest	enviada de gerente para agente, requisita o valor das instâncias do objeto MIB
GetNextRequest	enviada de gerente para agente, requisita o valor da próxima instância do objeto MIB
GetBulkRequest	enviada de gerente para agente, requisita valores em blocos
SetRequest	enviada de gerente para agente, define valores de uma ou mais instâncias de objetos MIB
Response	enviada de agente a gerente ou gerente a gerente, gerada em resposta a *GetRequest, GetNextRequest, SetRequest, GetBulkRequest*
Trap	enviada de agente para gerente, informa ao gerente um evento excepcional, um alerta.
Inform-Request	Confirmação do recebimento da PDU de trap

Fonte: baseado em Kurose e Ross(2010).

Uma mensagem SNMP é formada de vários componentes, como podemos ver na Figura 3, os componentes da mensagem são:

- Versão do SNMP;
- Comunidade requisitada;
- PDU que contém o corpo da mensagem SNMP que são os GetRequest, GetResponse, SetRequest e etc.
- O Request ID, que é um índice permite que o gerenciador SNMP corresponda a uma resposta recebida para a solicitação apropriada
- Error, se trata de um código gerado pelo agente SNMP se ocorrer um erro ao processar a solicitação;
- Error Index que é um ponteiro para o objeto que causou o erro;
- Varbindslist que se trata de uma sequência de varbinds
- Varbind, contêm informações adicionais sobre o evento relatado;
- OID, valor do ID de Objeto, que foi especificado na seção 2.1;
- Value, que atua como um espaço reservado para os dados de retorno.

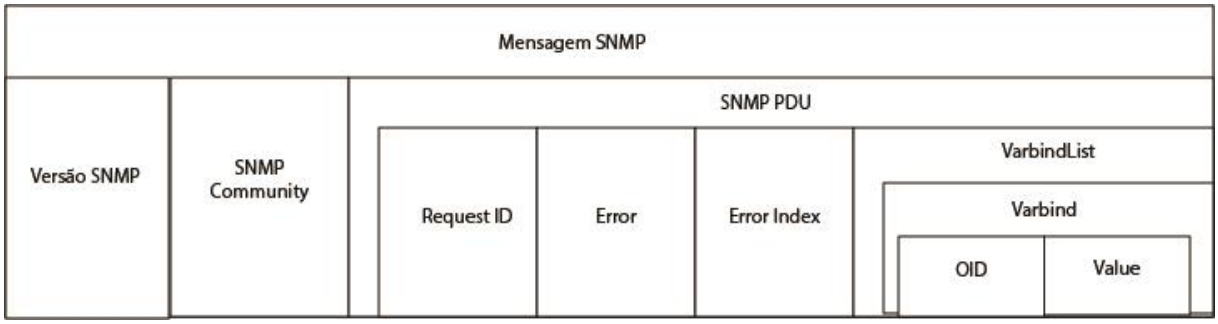

Figura 3- Mensagem SNMP. Fonte: baseado em. (SNMP library for Python, 2017)

.

4. PYTHON

Bruce Eckel afirma: "*Python fits your mind! "* (Python encaixa na sua mente), ou seja, Python é uma linguagem de fácil interação e aprendizado, de alto nível, de código aberto mantido sob a licença PSF (*Python Software Foundation, 2017*) que é compatível com a *General Public License* (GPL), multi plataforma, que proporciona um rápido desenvolvimento e fácil acesso as documentações para estudo.

> "A linguagem foi criada em 1990 por Guido van Rossum, no Instituto Nacional de Pesquisa para Matemática e Ciência da Computação da Holanda (CWI) e tinha originalmente foco em usuários como físicos e engenheiros. O Python foi concebido a partir de outra linguagem existente na época, chamada ABC." (Borges, 2010)

Segundo *Python Software Fundation*, (2017), trata-se de uma linguagem de programação robusta, utilizada por grandes empresas e projetos, como a ILM (*Industrial Light & Magic*) criada pelo cineasta George Lucas para criação de efeitos especiais, onde a linguagem contribuiu em várias áreas do desenvolvimento, devido a sua versatilidade.

O ponto forte do Python é sua documentação atualizada e de fácil acesso, disponível para toda a comunidade que utiliza essa linguagem. Aliás a comunidade Python é realmente grande e muito organizada. Segundo a documentação oficial (*Python Software Fundation*, 2017), Python já nasceu como software livre de código aberto e desde o início foi distribuída para que a comunidade pudesse contribuir com seu código, o próprio site onde consta a documentação do Python dispõe de um campo para a interação com o usuário, que

auxilia desde o iniciante na linguagem até o especialista, ajuda a divulgação e a contribuição na linguagem. Essa interação com a comunidade faz com que a linguagem esteja sempre evoluindo e com novos recursos.

Quando se fala em Python refere-se a três coisas, a linguagem Python, com seu conjunto de regras e gramáticas, um programa escrito na linguagem Python que é salvo com a extensão .py e o interpretador que é capaz de ler e executar esse arquivo.

O Python é dividido em módulos, portanto consegue-se utilizar um módulo em vários programas distintos. Existem vários módulos padrões que já vem com o Python, mas também pode-se criar módulos para executar recursos necessários para diariamente, assim como também compartilhar esses módulos na comunidade Python.

Segundo a documentação oficial (*Python Software Fundation,* 2017), a linguagem está na sua versão 3.6, porém a versão 2.7.13 ainda é muito utilizada.

4.1. PYSNMP

O PySNMP é um framework em Python que permite a implementação e desenvolvimento de um sistema SNMP. Software gratuito e de código aberto, multi plataforma que atualmente é possível implementar tanto agentes como gestores, com total eficiência sobre todas as versões do protocolo SNMP, a partir de métodos simples facilitando assim a criação de um sistema SNMP, trazendo maior flexibilidade aos usuários

Segundo a documentação oficial, SNMP *library for Python* (2017), trata-se de um módulo ou biblioteca Python que pode ser executado em Python 2.7 a 3.5, suporta configurações para IPv4 e IPv6 e vem acompanhando a evolução do protocolo SNMP em todas suas versões.

Com o PySNMP é possível criar um código capaz de enviar mensagens SNMP, como o "*get*", para os dispositivos agentes. O código do PySNMP é gratuito e disponível no GitHub[1], e na documentação oficial do Python está sendo distribuído pela licença BSD (*Berkeley Software Distribution*).

O módulo PySNMP possui classes específicas para trabalhar com SNMP, todas as operações envolvem a classe *Snmp Engine* independente. Para realizar a comunicação e

[1] https://github.com/etingof/pysnmp

autenticação entre gerente e agente nas versões 1 ou 2 do SNMP, utiliza-se a classe *Community Data* para a terceira versão do SNMP utiliza-se a classe *Usm User Data.*

O PySNMP suporta algoritmos de autenticação de mensagens MD5 e SHA, algoritmos de criptografia DES, AES128 / 192/256 e 3DES. O SNMP utiliza o protocolo UDP para encaminhar suas mensagens. Para tanto, no PySNMP devemos configurar a classe *Udp Transport Target*, para IPv4 ou Udp6TransportTarget, para IPv6.

Para enviar a mensagem "*get*" utiliza-se a função *getCmd* (). O *Context Data* é um parâmetro no cabeçalho de mensagem que endereça coleção específica de MIBs. O mecanismo SNMP pode servir muitos objetos MIB idênticos representando instâncias completamente diferentes de hardware ou software, a classe responsável por isso é a *Object Identity*. A classe *Object Type* que representa *OBJECT-TYPE SMI* é um objeto que faz referência *Object Identity* e instâncias de tipo SNMP. Os dados são armazenados em instâncias de objeto MIB endereçadas, adicionando informações (conhecidas como índice) aos identificadores de objeto MIB.

Por padrão, o PySNMP procurará no seu sistema de arquivos local os arquivos MIB ASN.1 referenciados no *ObjectIdentity*. A PDU (*Protocol Data Unit*) de resposta carregará uma lista de objetos MIB e seus valores exatamente na mesma ordem em que estavam na mensagem de solicitação.

4.2. KIVY

Segundo a documentação oficial o Kivy (2017) é uma biblioteca ou módulo Python *open source* que permite rápido desenvolvimento de aplicações *multitouch*. O Kivy está distribuído sobre os termos da licença do MIT a partir da versão 1.7.2.

Com o Kivy é possível desenvolver o mesmo código para Windows, Linux, Mac, iOS e Android e atualmente o Raspberrypi, sem muitas mudanças.

Ele foi criado a partir do zero, especialmente para esse requisito multitouch. Implementa os módulos em C, o que garante um alto desempenho para as aplicações. Um dos diferenciais do Kivy é que ele processa preferencialmente do GPU (*Graphics Processing Unit*) o que torna bem mais rápido o processamento.

Para poder utilizar o Kivy são necessários alguns requisitos básicos: o python a partir da versão 2.7 e o Cython.

Ele oferece uma linguagem de design, que trabalha como uma página de estilos, e utiliza um arquivo .kv, que atua junto com o seu arquivo .py facilitando a edição, auxiliando na configuração da parte gráfica e facilitando alteração.

4.3. SQLALCHEMY

O SQLAlchemy, segundo a documentação oficial SQLAlchemy (2017), é uma biblioteca Python que dá permite ter todo o poder e flexibilidade do SQL (*Structured Query Language* - Linguagem de Consulta Estruturada). Se trata de um ORM (*Object Relational Mapper* - Mapeamento Objeto-Relacional) ou seja um framework que permite trabalhar com banco de dados sem precisar escrever códigos de conexão com o banco, querys de SQL a todo momento. Com ele é possível escrever menos código e programa com muito mais produtividade, desenvolver um programa com código mais elegante.

Lançado em 2006, se tornou uma das mais utilizadas ferramentas ORM, juntamente com o ORM do Django no Python. Ou seja, se trata de uma ferramenta madura que está em constante desenvolvimento e possui um conjunto de ferramentas de alto desempenho e precisão.

Ele permite que sejam construídas funções em Python baseadas em estruturas SLQ compatíveis com qualquer banco de dados (SQLite, Postgresql, MySQL, Oracle, MS-SQL, Firebird, Sybase e outros).

Existe uma grande variedade de documentação sobre o SQLAlchemy, oficial e não oficial, no entanto a documentação oficial é bem completa.

5. METODOLOGIA

Para o desenvolvimento do trabalho teórico e prático foi realizado um levantamento bibliográfico sobre o protocolo SNMP e a linguagem de programação Python em materiais como artigos científicos, trabalhos acadêmicos, livros e internet.

A partir dos dados obtidos, foi possível entender e compreender melhor as ferramentas sobre o protocolo SNMP e construir um aplicativo de gerenciamento de redes.

5.1. DESENVOLVIMENTO DA FERRAMENTA

Baseado no protocolo SNMP, foi desenvolvida uma ferramenta utilizando a linguagem de programação Python capaz de enviar mensagens "*get"* ao cliente, a fim de receber os dados contato, descrição, id do objeto, localização, tempo desde que foi reiniciado, pacotes IP recebidos e pacotes IP enviados. Esses dados foram armazenados numa tabela no banco de dados e mostrados na tela.

Com essa aplicação também é possível que o gestor faça o agendamento de envio requisições ao cliente, definindo um intervalo de n segundos entre cada mensagem "*get*".

A partir dos dados recebidos dos clientes é possível gerar um relatório em csv, que permite analisar os dados. Todos esses processos geram um log com a data e a hora de acesso, criado pelo logs.py.

5.2. MAPA MENTAL

Segundo (Buzan, 1996 apud Keidann, Glaucia L. 2013), o criador desta técnica, mapas mentais são ferramentas de pensamento que permitem refletir exteriormente o que se passa na mente. São uma forma de organizar os pensamentos e utilizar ao máximo as capacidades mentais.

Para ficar claro o desenvolvimento da aplicação, foi criado um mapa mental, mostrado na Figura 4, a fim de organizar melhor a sequência de fatos e também para que se torne mais clara a explicação da criação da aplicação.

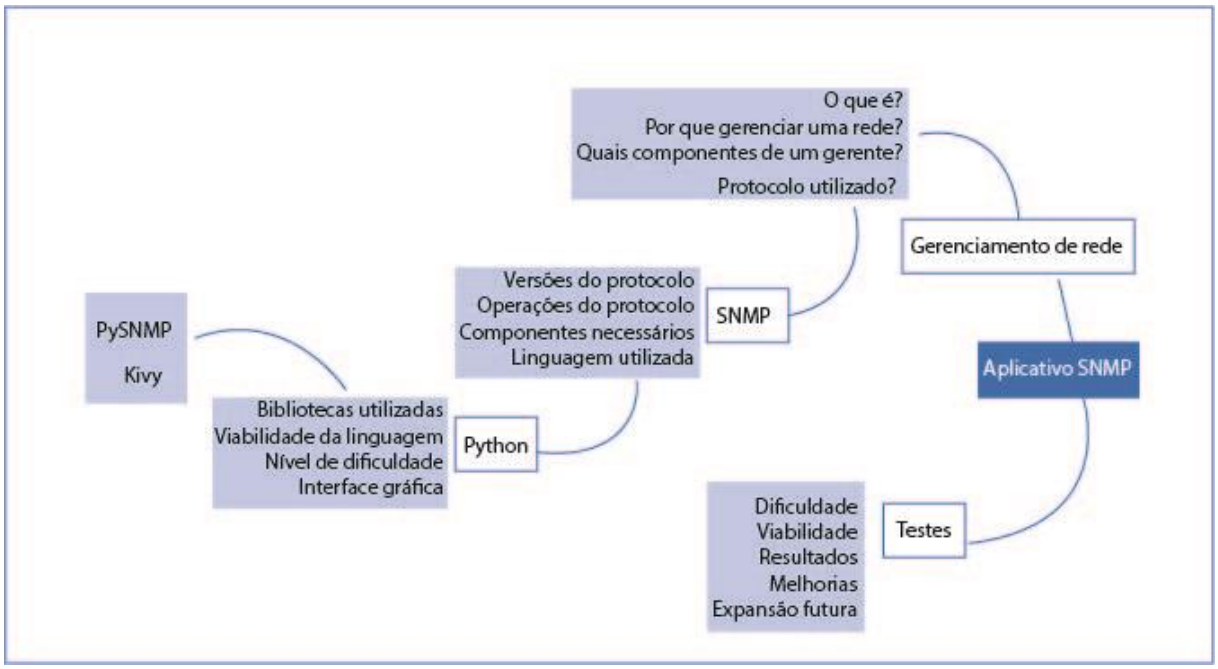

Figura 4- Mapa mental. Fonte: Autoral

Na Figura 4, pode-se observar os passos para a criação da aplicação para alcançar o objetivo final, que é o aplicativo de gerência SNMP. Para isso foi realizado primeiramente o estudo sobre gerenciamento de redes, levantando as questões. O que é? Por que gerenciar uma rede? Quais os componentes da rede? Qual protocolo utilizado? Com este último segue-se para o próximo galho do mapa mental em que se realiza o estudo do protocolo SNMP, suas versões, operações, componentes necessários e a linguagem a ser utilizada para se trabalhar com o protocolo. Foi definida a linguagem Python, onde leva-se em consideração, a

viabilidade e a facilidade de se desenvolver com a linguagem, bibliotecas que podem ser utilizadas, e a possibilidade de se desenvolver uma interface gráfica. Foram definidos o PySNMP como biblioteca utilizada para realizar as requisições SNMP e o Kivy como biblioteca responsável pelo desenvolvimento de uma interface gráfica. Por fim, com a aplicação finalizada, estende-se um outro galho do mapa mental, relacionado aos testes, que levantam questões como: melhoria, viabilidade, resultados, dificuldade e possibilidades de expansões futuras.

5.3. CONFIGURAÇÕES NECESSÁRIAS NO AGENTE

Para o funcionamento do SNMP deve ser configurada a comunidade e ser ativado o processo no agente. Para configurar o agente no Windows deve-se ir em serviços, e ativar o serviço SNMP. Em seguida em propriedades do serviço configurar a comunidade pela qual o agente responderá, a fim de viabilizar a leitura. Se não for feita a configuração da comunidade, ela será considerada *public*.

Esta configuração para Windows pode ser observada no APÊNDICE F nas Figuras 11 e 12. Para dispositivos Linux, a configuração encontra-se no APÊNDICE E.

5.4. REQUISITOS

A aplicação foi desenvolvida utilizando o sistema operacional Ubuntu 16.04.1. A maioria das distribuições Linux já vem com Python instalado, porém nem sempre com a versão mais atual. Para instalar a versão mais atualizada do Python foi utilizado o pyenv. Os passos para instalação estão no APÊNDICE G.

Além do Python, foram necessárias outras ferramentas importantes para o desenvolvimento do software e também para sua utilização. Estas ferramentas estão listadas no documento requeriments.txt disponível juntamente com o código fonte e a documentação do projeto no GitHub[2] e também está contido no APÊNDICE H.

O módulo PySNMP, citado na seção 4.1, implementa todas as versões do SNMP inteiramente em Python. Por esta razão, é necessário que este último esteja instalado antes da

[2] https://github.com/miriamfx/Projeto-final

utilização do referido módulo. Os comandos para instalação estão especificados no APÊNDICE G. Para esta aplicação utilizou-se o SNMPv2.

Também se utilizou o Kivy que, como citado na seção 4.2, é a biblioteca do Python responsável pela interação gráfica com o software. A instalação se encontra no APÊNDICE I.

No desenvolvimento do banco de dados foi utilizado o SQLAlchemy, outra biblioteca Python citada na seção 4.3, cuja instalação está descrita no APÊNDICE I.

5.5. DESENVOLVIMENTO DO SOFTWARE

O desenvolvimento da aplicação se divide em três partes: o estudo das ferramentas necessárias, o desenvolvimento da aplicação e os testes.

Antes de desenvolver a aplicação foi necessário traçar qual caminho seria tomado para alcançar o objetivo final. Como pode-se ver na Figura 5, o diagrama de caso de uso descreve como seria a interação do usuário com o aplicativo.

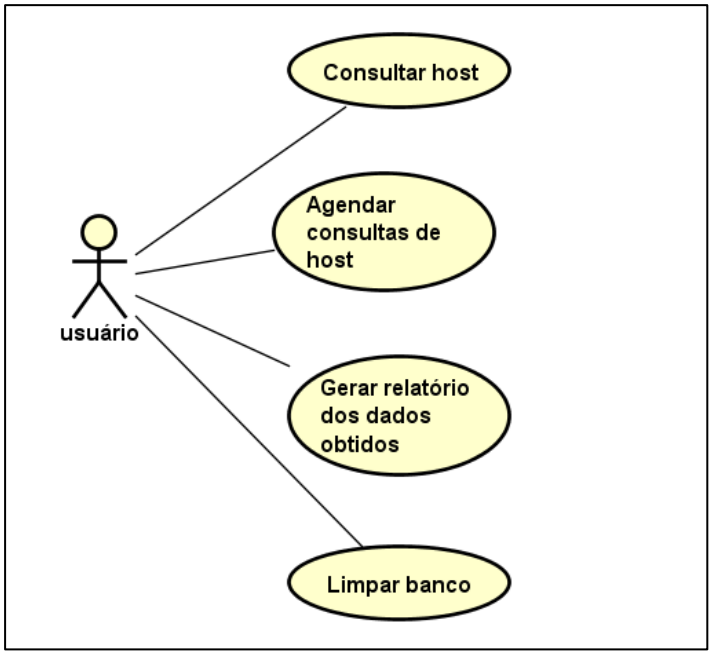

Figura 5 - Diagrama de caso de uso.

Como citado na seção 5.4, o Python é um requisito para o desenvolvimento pois todo o código foi escrito nessa linguagem.

Na Figura 6 segue um diagrama de classe que exemplifica o funcionamento do aplicativo e as relações entre as classes do aplicativo.

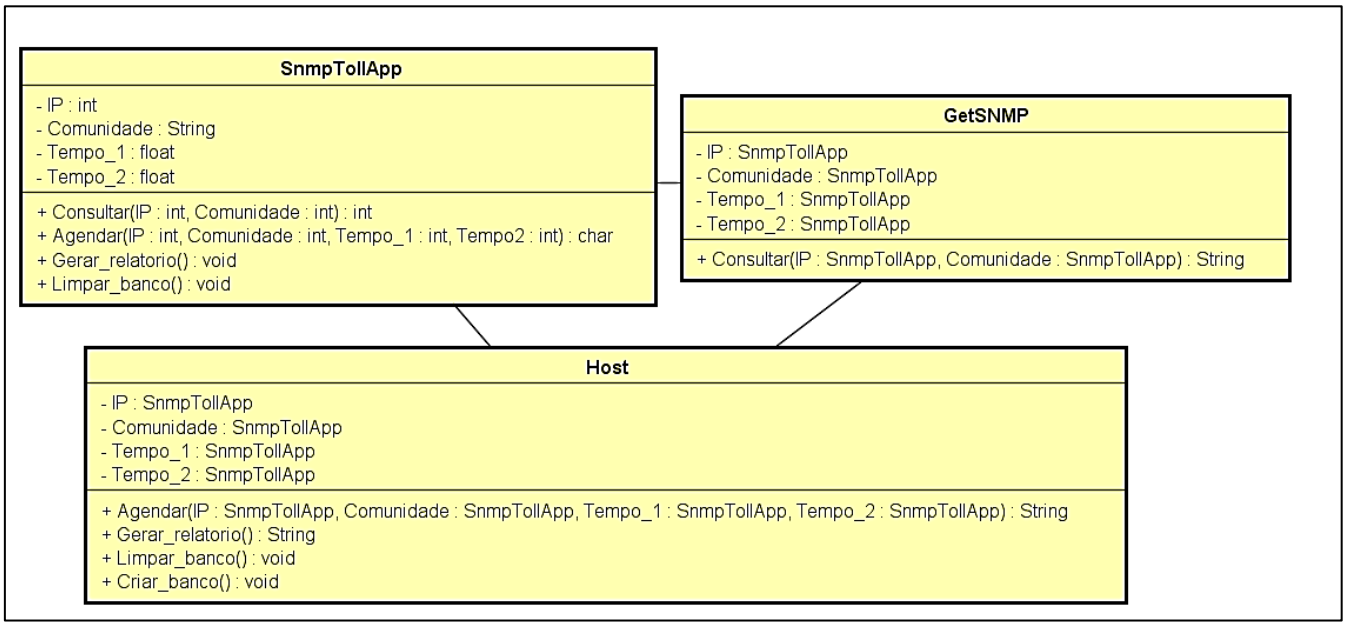

Figura 6 - Diagrama de classes.

Como representado no diagrama pode-se observar na Figura 7 a tela inicial do aplicativo, que apresenta os quatro campos do tipo text_input direcionados a entrada de dados, que são:

- IP;
- Comunidade;
- Get/sec, responsável pela quantidade em segundos em que o agendamento será executado;
- Dur/sec, responsável pela duração total do agendamento.

Na tela principal ainda, pode-se ver um campo adicional que é responsável por apresentar os resultados na tela.

Por fim, podemos observar os botões que executam as funções propostas pela aplicação, que são:

- Consultar, que envia uma mensagem "*get"* ao host para o qual foram informados o IP e a comunidade.
- Agendar, que após determinação dos tempos Get/sec e Dur/sec, envia mensagens "*get"* ao host informado, no tempo especificado.
- Gerar Rel, que gera um relatório em .csv com todos os dados colhidos pela aplicação

- Limpar, realiza a limpeza do banco de dados.

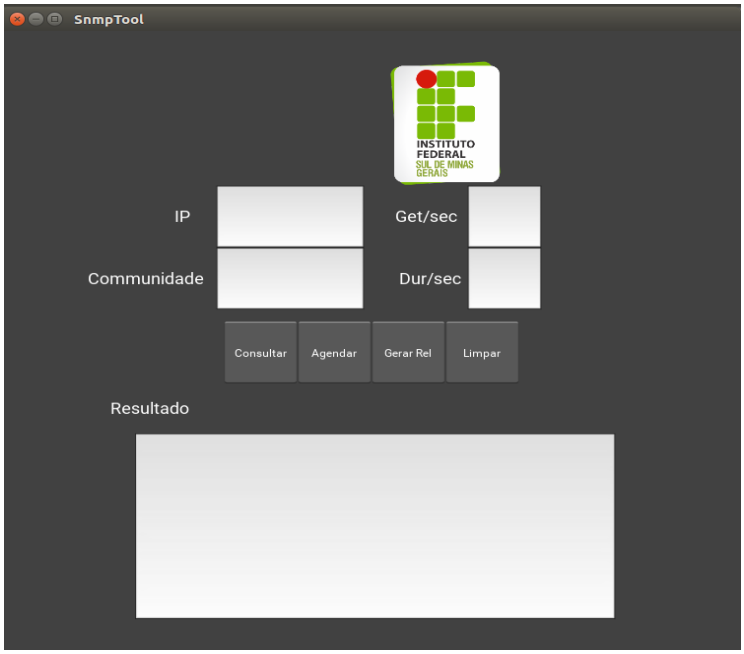

Figura 7 - Tela principal do programa.

5.6. ANÁLISE DO CÓDIGO

Antes de analisar a classe é interessante frisar que o PySNMP oferece várias funções para usar o SNMP, mas é aconselhável utilizar o HLAPI, que significa *High Level API* e oferece funções de alto nível, bastante fáceis de configurar.

Na classe SimpleSnmp é declarada a função GetSNMP, que é responsável pelas requisições. Como é possível analisar na Figura 8, para enviar uma requisição SNMP deve-se especificar a OID através do *ObjectType*, juntamente com a classe *Object Identity* onde informou-se a OID e a MIB. O identificador de objeto pode ser definido tanto em alto nível, como visto no exemplo do *sysLocation* quanto em baixo nível através da identificação 1.3.6.1.2.1.1.6, onde apresenta o mesmo resultado.

```
22      def GetSNMP(self): #realiza o get snmp
23          data = (
24              ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysLocation', 0)),
25              ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysDescr', 0)),
26              ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysObjectID', 0)),
27              ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysUpTime', 0)),
28              ObjectType(ObjectIdentity('IP-MIB', 'ipInDelivers', 0)),
29              ObjectType(ObjectIdentity('IP-MIB', 'ipOutRequests', 0))
```

Figura 8 - get.py declaração de OID.

Outro ponto importante a ser analisado nessa função é o *SnmpEngine*, que pode ser observado na Figura 9. Ele é o um objeto central que controla os outros componentes do sistema SNMP. O *Community Data* é responsável por receber a comunidade. O mpModel define a versão do SNMP utilizada, no caso a versão v2, para mudar para v1 é necessário apenas alterar o parâmetro para 0, para utilizar a versão v3 é necessário ser instanciada a

classe *Usm User Data*.

O *Udp Transport Target* recebe o IPv4 requisitado juntamente com a porta de comunicação 161. Para ser utilizado IPv6 deve ser definido Udp6TransportTarget.

```
33          g = getCmd(SnmpEngine()
34                     , CommunityData(self.community, mpModel=1)
35                     , UdpTransportTarget((self.ip, 161))
36                     , ContextData()
37                     , *data)
38
39          errorIndication, errorStatus, errorIndex, varBinds = next(g)
```

Figura 9 – Snmp Engine

O arquivo responsável pelo banco de dados é o dbmanager.py contido no APÊNDICE D. Nele foi criada a tabela hosts que recebe todos os dados gerados pelo get.py.

A função gerar_rel no main.py é acionada com o botão gerar relatório, que encaminha para a função rel_hosts do dbmanager, onde retorna os valores contidos no banco de dados criado no arquivo snmpdb.db e encaminha para um arquivo csv chamado snmp_rel.csv.

A função clean no arquivo main.py é acionada pelo botão limpar e encaminha para a função drop no manager.py e realiza a limpeza total do banco de dados e depois recria as tabelas vazias.

E enfim a função get_agendado no main.py acionada pelo botão agendar, recebe o IP, comunidade, time1 e o time2. Esta função executa o get.py com intervalos estipulados no time1 até o tempo limite estipulado no time2, para que possa ser executada outras funções enquanto o agendamento está em execução foi utilizada a função Thread no main.py que executa a função get_agendado em uma thread sem comprometer o funcionamento do mesmo.

Todos os arquivos do Python quando executados acionam o logs.py, descrito no APÊNDICE E, que gera um log de execução do programa com data e hora de execução.

6. TESTES

Foram realizados testes com o aplicativo a fim de qualificar e analisar seu funcionamento. Nesses testes era esperada a obtenção dos dados passados nos parâmetros do código fonte de três formas:

- Apresentados na tela;
- Por meio de um relatório;
- Banco de dados.

Com os testes pode-se analisar o funcionamento das funções e o comportamento do aplicativo em relação ao tempo de resposta e à facilidade de utilização.

6.1. CONFIGURAÇÃO DA REDE DE TESTE

O cenário onde feito o teste continha um computador desktop (agente 1) e dois notebooks (agente 2 e gerente). O diagrama da rede está apresentado na Figura 10.

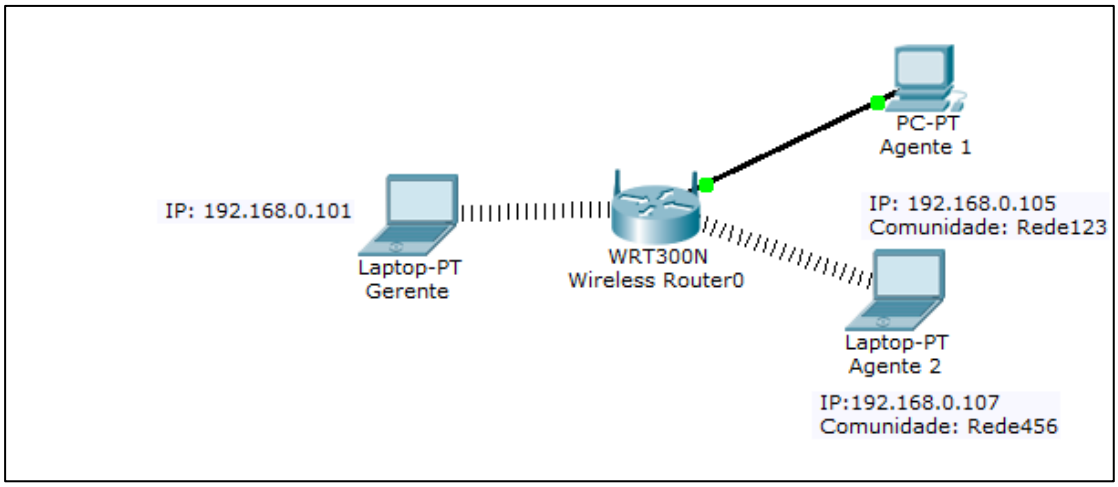

Figura 10- Diagrama da rede. Fonte: Autoral.

Na Tabela 3 é apresentada a configuração da rede em que foi realizado o teste:

Tabela 3 - Configuração da rede.

	Sistema Operacional	**IP**	**Comunidade**
Gerente	Ubuntu	192.168.0.101	
Agente 1	Windows 7	192.168.0.105	Rede123
Agente 2	Windows 7	192.168.0.107	Rede123

6.2. RESULTADO DO TESTE

O programa localizado no gerente recebe como parâmetros de entrada o IP e a comunidade, através do módulo main.py.

O módulo main.py encaminha para o módulo get.py a variável OID. Sequencialmente são enviadas requisições "*get"* pela estação de gerenciamento para o agente. O agente recebe, processa a requisição e transmite uma mensagem "*get-response"* para a estação de gerenciamento e retorna os valores obtidos na MIB de cada objeto da lista de requisições "*get*". Estes valores são exibidos na tela e salvos no banco de dados.

Segue descrição das OID que foram obtidas no teste, segundo documentação da MIB-2, na Tabela 4.

Tabela 4 - OID utilizados

Objeto SNMP	**Grupo**	**Descrição**
sysLocation	System	Especifica a localização física do computador
sysDescr	System	Uma descrição textual da entidade. Esse valor deve incluir

		o nome completo e a identificação da versão do tipo de hardware do sistema, do sistema operacional do software e do software de rede
sysObjectID	System	Identificação autorizada do fornecedor do subsistema de gestão de rede contido na entidade.
sysUpTime	System	O tempo (em centésimos de segundo) desde que a parte de gerenciamento de rede do sistema foi reinicializada pela última vez.
ipInDelivers	IP	Entrada de pacotes IP
ipOutRequests	IP	Saída de pacotes IP

Na Tabela 5, podemos analisar os resultados obtidos através do módulo GetSNMP, executando uma requisição com as OID citadas na Tabela 4. Como pode-se observar nesta tabela, os dados foram retornados, sequencialmente, na mesma ordem em que foram requisitados pelo gerente.

Tabela 5 - Resultados dos testes nos Agentes 1 e 2.

Agente 1	**Agente 2**
['SNMPv2-MIB::**sysLocation**.0 = Casa', 'SNMPv2-MIB::**sysDescr**.0 = Hardware: Intel64 Family 6 Model 58 Stepping 9 AT/AT COMPATIBLE - Software: Windows Version 6.1 (Build 7601 Multiprocessor Free)', 'SNMPv2-MIB::**sysObjectID**.0 = SNMPv2-SMI::enterprises.311.1.1.3.1.1', 'SNMPv2-MIB::**sysUpTime**.0 = 1083491', 'IP-MIB::**ipInDelivers**.0 = 51076', 'IP-MIB::**ipOutRequests**.0 = 40771' **IP 192.168.0.105 e Community REDE123**	['SNMPv2-MIB::**sysLocation**.0 = note yara', 'SNMPv2-MIB::**sysDescr**.0 = Hardware: Intel64 Family 6 Model 37 Stepping 5 AT/AT COMPATIBLE - Software: Windows Version 6.1 (Build 7600 Multiprocessor Free)', 'SNMPv2-MIB::**sysObjectID**.0 = SNMPv2-SMI::enterprises.311.1.1.3.1.1', 'SNMPv2-MIB::**sysUpTime**.0 = 57191', 'IP-MIB::**ipInDelivers**.0 = 1280', 'IP-MIB::**ipOutRequests**.0 = 1291' **IP 192.168.0.107 e Community rede123**

Este resultado foi mostrado na tela e salvo no banco de dados. Para melhor visualização do resultado na tabela, em negrito se encontra a OID, e destacado com fundo azul o resultado do valor obtido da OID de cada host.

Realizou-se também um teste utilizando o agendamento, foi estipulado o tempo para cada requisição de 2 segundos e o tempo de duração total do agendamento de 10 segundos.

Os resultados obtidos foram enviados para o banco de dados e resgatados através da criação de um relatório em formato .csv, gerado a partir do botão “gerar rel” da aplicação.

Podemos observar os resultados nas Tabelas 7 e 8, criadas a partir dos dados obtidos no relatório.

Tabela 6 - Dados obtidos no teste de agendamento do Agente 1

Requisição 1
192.168.0.105
REDE123
SNMPv2-MIB::**sysLocation**.0 = Casa
SNMPv2-MIB::**sysDescr**.0 = Hardware: Intel64 Family 6 Model 58 Stepping 9 AT/AT COMPATIBLE - Software: Windows Version 6.1 (Build 7601 Multiprocessor Free)
SNMPv2-MIB::**sysObjectID**.0 = SNMPv2-SMI::enterprises.311.1.1.3.1.1
SNMPv2-MIB::**sysUpTime**.0 = 1092806
IP-MIB::**ipInDelivers**.0 = 51277
IP-MIB::**ipOutRequests**.0 = 40968
Requisição 2
192.168.0.105
REDE123
SNMPv2-MIB::**sysLocation**.0 = Casa
SNMPv2-MIB::**sysDescr**.0 = Hardware: Intel64 Family 6 Model 58 Stepping 9 AT/AT COMPATIBLE - Software: Windows Version 6.1 (Build 7601 Multiprocessor Free)
SNMPv2-MIB::**sysObjectID**.0 = SNMPv2-SMI::enterprises.311.1.1.3.1.1
sysUpTime.0 = 1093061
IP-MIB::**ipInDelivers**.0 = 51283
MIB::**ipOutRequests**.0 = 40972
Requisição 3
192.168.0.105
REDE123
SNMPv2-MIB::**sysLocation**.0 = Casa
SNMPv2-MIB::**sysDescr**.0 = Hardware: Intel64 Family 6 Model 58 Stepping 9 AT/AT COMPATIBLE - Software: Windows Version 6.1 (Build 7601 Multiprocessor Free)
SNMPv2-MIB::**sysObjectID**.0 = SNMPv2-SMI::enterprises.311.1.1.3.1.1
sysUpTime.0 = 1093314
IP-MIB::**ipInDelivers**.0 = 51289
MIB::**ipOutRequests**.0 = 40976

Tabela 7 - Dados obtidos no teste de agendamento do Agente 2.

Requisição 1
192.168.0.107 **rede123** SNMPv2-MIB::**sysLocation**.0 = note yara SNMPv2-MIB::**sysDescr**.0 = Hardware: Intel64 Family 6 Model 37 Stepping 5 AT/AT COMPATIBLE - Software: Windows Version 6.1 (Build 7600 Multiprocessor Free) SNMPv2-MIB::**sysObjectID**.0 = SNMPv2-SMI::enterprises.311.1.1.3.1.1 SNMPv2-MIB::**sysUpTime**.0 = 97755 IP-MIB::**ipInDelivers**.0 = 1831 IP-MIB::**ipOutRequests**.0 = 1879
Requisição 2
192.168.0.107 **rede123** SNMPv2-MIB::**sysLocation**.0 = note yara SNMPv2-MIB::sysDescr.0 = Hardware: Intel64 Family 6 Model 37 Stepping 5 AT/AT COMPATIBLE - Software: Windows Version 6.1 (Build 7600 Multiprocessor Free) SNMPv2-MIB::**sysObjectID**.0 = SNMPv2-SMI::enterprises.311.1.1.3.1.1 SNMPv2-MIB::**sysUpTime**.0 = 98029 IP-MIB::**ipInDelivers.0** = 1840 IP-MIB::**ipOutRequests**.0 = 1888
Requisição 3
192.168.0.107 **rede123** SNMPv2-MIB::**sysLocation**.0 = note yara SNMPv2-MIB::**sysDescr**.0 = Hardware: Intel64 Family 6 Model 37 Stepping 5 AT/AT COMPATIBLE - Software: Windows Version 6.1 (Build 7600 Multiprocessor Free) SNMPv2-MIB::**sysObjectID**.0 = SNMPv2-SMI::enterprises.311.1.1.3.1.1 SNMPv2-MIB::**sysUpTime**.0 = 98280 IP-MIB::**ipInDelivers**.0 = 1845 IP-MIB::**ipOutRequests**.0 = 1893

Como pode ser observado, obviamente as únicas variáveis que obtém alteração são a *sysUpTime*, *ipInDelivers*, *ipOutRequests*. As outras informações geradas são estáticas e só irão mudar em algum caso excepcional, como a troca de um hardware ou alteração da configuração SNMP do host e servem para identificação.

Com os dados obtidos nesse relatório pode-se analisar o índice de entrada e saída de pacotes IP por host analisado, que são dados interessantes para se poder validar o impacto das mudanças planejadas na rede, descobrir a causas do congestionamento da rede, medir o consumo de cada usuário, entre outros.

CONCLUSÃO

Este trabalho apresentou o desenvolvimento de uma ferramenta de código aberto sob licença GPL3, desenvolvida em Python para auxiliar no gerenciamento de rede.

A ferramenta é capaz de coletar informações dos hosts indicados em tempo real como localização ~~do host~~, descrição do hardware/software, contato do responsável pelo equipamento, tempo de reinicialização da rede, identificação do objeto, número de pacotes IP enviados e pacotes IP recebidos.

Com base nos dados coletados pode-se analisar o índice de entrada e saída de IP e traçar um comportamento dos equipamentos gerenciados, prevenindo futuros problemas na rede, identificando gargalos ou possíveis falhas devido às mudanças na rede.

No processo de estudo constatou-se que o PySNMP é uma ferramenta de fácil compreensão para se desenvolver um aplicativo mais completo que vise o gerenciamento de redes. Também é interessante lembrar que, como software livre, pode servir de colaboração na comunidade Python e ser atualizada e melhorada constantemente.

Foi observada também a possibilidade de executar este aplicativo no Android, visto que o Kivy possibilita a criação de aplicativos multitoutch o que tornaria mais prática sua execução.

Ao final deste projeto pode-se concluir a eficiência do protocolo SNMP, com base nos estudos dos autores consultados e na implementação da ferramenta. Também as amplas

opções dadas pelo Python com a biblioteca PySNMP para serem realizados projetos futuros, como a melhoria da ferramenta para enviar mensagens SetSNMP e receber outros tipos de OID, a escolha do gerente sem precisar alterar o código fonte ou a geração de gráficos a partir do banco de dados.

.

REFERÊNCIAS BIBLIOGRÁFICAS

BORGES, L. E. **Python para Desenvolvedores**. 2° edição Rio de Janeiro, 2010.

DIAS, B. Z., ALVES JR. N. **Protocolo de gerenciamento SNMP**.

FRAGA, S. D. R. **Monitorização de Processos Multimedia.** Monografia de Mestrado (Faculdade de Engenharia da Universidade do Porto FEUP). Jun 2009

KEIDANN, Glaucia L. **Utilização de Mapas Mentais na Inclusão Digital.** Universidade Regional do Noroeste do Rio Grande do Sul - Ijuí, RS. Jun 2013.

Míriam Félix. Github. Disponível em:<https://github.com/miriamfx/Projeto-Final> Data do acesso: 01/09/2016.

GUILLERMO, O. E. P. **Uso de Agentes SNMP para monitoramento de Servidores e equipamentos de rede com mobilidade**. Trabalho de Conclusão de Curso (Universidade Federal do Rio Grande do Sul- Instituto de Informática). Porto Alegre, dez 2008.

Kivy documentation. Disponível em: <https://kivy.org/#home>. Data do acesso: 01/09/2016.

KUROSE, J. F., ROSS, K. W. **Redes de Computadores e a Internet: Uma abordagem top-down.** James F. Kurose e Keith W. Ross. São Paulo: Addison Wesley- 2010. 5° edição. 9, p. 555, p. 556

PEDROSA, F., TEIXEIRA, J., OLIVEIRA, F. **SNMP**. Monografia (Departamento de Ciência de Computadores, FCUP). 17 mai 2004.

PINHEIRO, J. M. D. S. **Gerenciamento de Redes de Computadores**. Versão 2.0. Agosto 2002.

Python 3.5.2 documentation. Disponível em: <https://docs.python.org/3/>.Data do acesso: 01/09/2016.

SNMP library for Python. Disponível em: <http://pysnmp.sourceforge.net/>. Data do acesso: 01/09/2016.

SQLAlchemy 1.2 Documentation. Disponível em: < https://goo.gl/WOhNDS >. Data do acesso: 29/03/2017.

STALLINGS, W. **Redes e Sistemas de Comunicação de Dados: Teoria e aplicações corporativas**/ William Stallings. Rio de Janeiro: Elsevier, 2005. 5° edição. p. 410, p. 411, p.412

APÊNDICE A

Tabela 8 - Linhas de código do arquivo main.py.

```
1  #!/usr/bin/env python
2  #coding: utf-8
3
4  from kivy.app import App
5  from kivy.properties import ObjectProperty, StringProperty
6  from kivy.uix.floatlayout import FloatLayout
7  from kivy.uix.button import Button
8  from kivy.uix.label import Label
9  from kivy.uix.textinput import TextInput
10 from kivy.core.window import Window
11 from kivy.uix.popup import Popup
12 from get import SimpleSnmp
13 import dbmanager
14 import time
15 import re
16 import sys
17 from logs import logGet,logRel,logDrop,logAgendGet
18 from threading import Thread, Timer
19
20 #O main é responsável pela delegação das funções
21 #Cada botão executa uma requisição a um modulo do aplicativo
22 #Classe principal SnmpTool
23
24 class SnmpToolApp(App):
25     btn1 = 'Consultar'#executa a função get_values_form()
26     btn2 = 'Agendar' #executa a função agendar()
27     btn3 = 'Gerar Rel' #executa a função gera_rel()
28     btn4 = 'Limpar' #executa a função clean
29
30     def gerar_rel(self): # Esta função é chamada no main.kv no Button btn3
31         dbmanager.host.rel_hosts(self)#executa a função host.rel_hosts no dbmanager.py
32         result = 'Relatório gerado com sucesso.' #result recebe uma mensagem de sucesso
33         self.set_result_form(result) #exibe a mensagem na função set_result_for()
34         logRel(self)
35
36     def clean(self):# Realiza o Drop na tabela do banco de dados
37         dbmanager.host.drop(self) #executa a função host.drop() no dbmanage.py
38         result = 'Drop table realizado com sucesso.'
39         self.set_result_form(result) #exibe a mensagem na função set_result_for()
40         logDrop(self)
41
42     def agendar(self, ip, community, time1, time2):
43         Thread(target=self.get_agendado, kwargs={
44
45             'ip': ip,
```

```
46              'community': community,
47              'time1': time1,
48              'time2': time2
49          }).start()
50
51      def get_agendado(self, ip, community, time1, time2):
52          # Variavel para a contagem de tmepo decorrido
53          time_delay = int(time1)
54          time_final = int(time2)
55          tempo_inicial = time.time()
56
57          while True:
58              # Executa o primeiro get
59              self.get_values_form(ip, community)
60              # Pausa no itervalo especificado
61              time.sleep(time_delay)
62              # se o tempo atual for maior que o tempo esperado o loop sera encerrado
63              if time.time() > tempo_inicial + time_final:
64                  Break
65
66          self.set_result_form(self.result)
67          logAgendGet(self)
68
69      def build(self):#responsável pela montagem do layout
70          Window.size = (750, 750) #tamanho da janela
71          self.load_kv('main.kv') #caminho do arquivo main.kv
72
73      def set_result_form(self, resultado): #responsável por mostrar as mensagens na tela
74          self.root.ids.textinput_resultado.text = resultado
75
76          print (resultado)
77
78      def get_values_form(self, ip, community): #executa o get.py
79          a = SimpleSnmp(ip, community)
80          result = a.GetSNMP()
81          result = result + '\n IP ' + ip
82          result = result + ' e Community ' + community
83          self.set_result_form(result)
84          logGet(self)
85
86  if __name__ == "__main__":
87      SnmpToolApp().run()
```

APÊNDICE B

Tabela 9 - Linhas de código do arquivo main.kv.

```
<Button>:
    color: 1, 1, 1, 1
    font_size: 12
    size_hint: .1, .1

FloatLayout:
    orientation: 'horizontal'
    canvas:
        Color:
            rgb: [.255, .255, .255]
        Rectangle:
            pos: self.pos
            size: self.size

    Image:
        source:'500.png'
        size_hint_x: 1
        size_hint_y: .2
        pos_hint: {'x': .10, 'top': .95}
    Label:
        text: "IP  "
        font_size: 18
        color: 1, 1, 1, 1
        pos_hint: {'x': .20, 'top': .75}
        size_hint: .1, .1

    TextInput:
        id: textinput_ip
        text: ''
        pos_hint: {'x': .29, 'top': .75}
        font_size: 12
        size_hint: .2, .1
    Label:
        text: "Communidade  "
        font_size: 18
        color: 1, 1, 1, 1
        pos_hint: {'x': .15, 'top': .65}
        size_hint: .1, .1
    TextInput:
        id: textinput_community
        text: ''
        pos_hint: {'x': .29, 'top': .65}
        font_size: 12
        size_hint: .2, .1
    Label:
```

```
46          text: "Get/sec  "
47          font_size: 18
48          color: 1, 1, 1, 1
49          pos_hint: {'x': .53, 'top': .75}
50          size_hint: .1, .1
51      Label:
52          text: "Dur/sec"
53          font_size: 18
54          color: 1, 1, 1, 1
55          pos_hint: {'x': .53, 'top': .65}
56          size_hint: .1, .1
57      TextInput:
58          id: textinput_time1
59          text: "
60          pos_hint: {'x': .63, 'top': .75}
61          font_size: 12
62          size_hint: .1, .1
63      TextInput:
64          id: textinput_time2
65          text: "
66          pos_hint: {'x': .63, 'top': .65}
67          font_size: 12
68          size_hint: .1, .1
69      Button:
70          text: app.btn1
71          pos_hint: {'x': .3, 'top': .53}
72        on_press:app.get_values_form(textinput_ip.text,textinput_community.text)
73          id: btn_consultar
74      Button:
75          text: app.btn2
76          pos_hint: {'x': .4, 'top': .53}
77  on_press:app.agendar(textinput_ip.text,textinput_community.text,textinput_time1.text,te
    xtinput_time2.text)
78          id: btn_agendar
79      Button:
80          text: app.btn3
81          pos_hint: {'x': .5, 'top': .53}
82          on_press:app.gerar_rel( )
83          id: btn_gerar_rel
84      Button:
85          text: app.btn4
86          pos_hint: {'x': .6, 'top': .53}
87          on_press:app.clean( )
88          id: btn_limpar
89      Label:
90          text: "Resultado"
91          font_size: 18
92          color: 1,1,1,1
```

```
93          pos_hint: {'x': .10, 'top': .44}
94          size_hint: .2, .1
95      TextInput:
96          id: textinput_resultado
97          text: ''
98          pos_hint: {'x': .18, 'top': .35}
99          font_size: 12
10          size_hint: .65, .30
0
```

APÊNDICE C

Tabela 10 - Linhas de código do arquivo get.py.

```
1  #!/usr/bin/env python
2  #coding: utf-8
3
4
5
6
7
8  from pysnmp.hlapi import *
9  from kivy.properties import ObjectProperty
10 from os import *
11 import dbmanager
12 import os.path
13 import os
14 from datetime import datetime
15
16 class SimpleSnmp():
17     def __init__(self, ip, community):#recebe os atributos de entrada do main
18         self.ip = ip
19         self.community = community
20
21
22     def GetSNMP(self): #realiza o get snmp
23         data = (
24             ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysLocation', 0)),
25             ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysDescr', 0)),
26             ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysObjectID', 0)),
27             ObjectType(ObjectIdentity('SNMPv2-MIB', 'sysUpTime', 0)),
28             ObjectType(ObjectIdentity('IP-MIB', 'ipInDelivers', 0)),
29             ObjectType(ObjectIdentity('IP-MIB', 'ipOutRequests', 0))
30
31         )
32
33         g = getCmd(SnmpEngine()
34                    , CommunityData(self.community, mpModel=0)
35                    , UdpTransportTarget((self.ip, 161))
36                    , ContextData()
37                    , *data)
38
39         errorIndication, errorStatus, errorIndex, varBinds = next(g)
40
41         if errorIndication:
42             print(errorIndication)
43         elif errorStatus:
44             print('%s at %s' % (
45                 errorStatus.prettyPrint(),
```

```
46              errorIndex and varBinds[int(errorIndex) - 1][0] or '?'
47          )
48
49               )
50      else:
51          lista = []
52          cont = 0
53          for varBind in varBinds:
54              if cont < 7:
55                  lista.append((' = '.join([x.prettyPrint() for x in varBind])))
56                  cont = cont + 1
57
58
59      host = dbmanager.host()
60      #salva a requisição no banco de dados
61      host.ip = self.ip
62      host.comunidade = self.community
63      host.contact = str(lista[0])
64      host.desc = str(lista[1])
65      host.idObject = str(lista[2])
66      host.location = str(lista[3])
67      host.uptime = str(lista[4])
68      host.ipInDelivers = str(lista[5])
69      host.ipOutRequests = str(lista[-1])
70      host.data = str(datetime.now)
71
72      host.save()
73
74      return str(lista)
```

APÊNDICE D

Tabela 11 - Linhas de código do arquivo manager.py.

```
1  #!/usr/bin/env python
2  #coding: utf-8
3
4
5
6
7
8  from sqlalchemy import create_engine, Table, Column, MetaData,engine
9  from sqlalchemy import Table, Column, Integer, String, ForeignKey, MetaData
10 from sqlalchemy.ext.declarative import declarative_base
11 from sqlalchemy.ext.serializer import loads, dumps
12 from sqlalchemy.orm import sessionmaker
13 from sqlalchemy import delete, select
14 import csv
15 import subprocess
16 import get
17 import os.path
18
19 #conecta o banco de dados
20 base = declarative_base()
21 engine = create_engine('sqlite:///snmpdb.db')
22 base.metadata.bind = engine
23
24 #cria a tabela
25 class host(base):
26     __tablename__ = 'hosts'
27     id = Column(Integer,primary_key=True)
28     ip = Column(String(15))
29     comunidade = Column(String(8))
30     contact = Column(String)
31     desc = Column(String)
32     uptime = Column(String)
33     idObject = Column(String(100))
34     location = Column(String(100))
35     ipInDelivers = Column(String)
36     ipOutRequests = Column(String)
37     data = Column(String)
38
39
40     def save(self):
41         DBSession = sessionmaker(bind=engine)
42         session = DBSession()
43         session.add(self)
44         session.commit()
45
```

```
46      #realiza um dump na tabela do banco de dados e salva no arquivo csv (executado pelo botão gera rel)
47      def to_dict(self):
48          _t = {
49              'id': self.id, 'ip': self.ip, 'comunidade': self.comunidade, 'contact': self.contact,
50              'desc': self.desc, 'uptime': self.uptime, 'idObject': self.idObject, 'location': self.location,
51              'ipInDelivers': self.ipInDelivers, 'ipOutRequests': self.ipOutRequests,
52              'data': self.data,
53          }
54          return _t
55      def rel_hosts(self):
56          DBSession = sessionmaker(bind=engine)
57          session = DBSession()
58
59          with open('snmp_rel.csv', 'w') as csv_file:  # Just use 'w' mode in 3.x
60              fieldnames = [
61                  'id', 'ip', 'comunidade', 'contact', 'desc', 'uptime', 'idObject', 'location',
62                  'ipInDelivers','ipOutRequests','data'
63              ]
64              writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
65              #writer.writeheader()
66              for row in session.query(host).order_by('id'):
67                  writer.writerow(row.to_dict())
68
69      #Exclui toda a tabela (executado pelo botão Limpar)
70      def drop(self):
71          host.metadata.drop_all(engine) #drop da tabela
72          base.metadata.create_all(engine) #cria a tabela vazia
73
74  base.metadata.create_all(engine)
```

APÊNDICE E

Tabela 12 - Linhas de código do arquivo logs.py.

```
1  #!/usr/bin/env python
2  #coding: utf-8
3  import get
4  import dbmanager
5  import os,sys
6  from os import path
7  from datetime import datetime
8  import time
10 def logGet(self):
11     date = str(time.strftime("%Y-%m-%d"))  #
12     now = str(datetime.now())
13     logfile = open('%s-getlog.txt' % date,'w')  # Cria o arquivo de Log
14     texto = []
15     texto.append(now)
16     texto.append(' - get REALIZADO')
17     logfile.writelines(texto)
18     logfile.close()
19
20 def logRel(self):
21     date = str(time.strftime("%Y-%m-%d"))  #
22     now = str(datetime.now())
23     logfile = open('%s-relatoriolog.txt' % date, 'w')  # Cria o arquivo de Log
24     texto = []
25     texto.append(now)
26     texto.append(' - relatorio criado')
27     logfile.writelines(texto)
28     logfile.close()
29
30 def logAgendGet(self):
31     date = str(time.strftime("%Y-%m-%d"))  #
32     now = str(datetime.now())
33     logfile = open('%s-Agendamentolog.txt' % date,'w')  # Cria o arquivo de Log
34     texto = []
35     texto.append(now)
36     texto.append(' - Agendamento REALIZADO')
37     logfile.writelines(texto)
38     logfile.close()
39
40 def logDrop(self):
41     date = str(time.strftime("%Y-%m-%d"))  #
42     now = str(datetime.now())
43     logfile = open('%s-droplog.txt' % date,'w')  # Cria o arquivo de Log
44     texto = []
45     texto.append(now)
46     texto.append(' - drop REALIZADO')
```

```
47      logfile.writelines(texto)
48      logfile.close()
```

APÊNDICE F

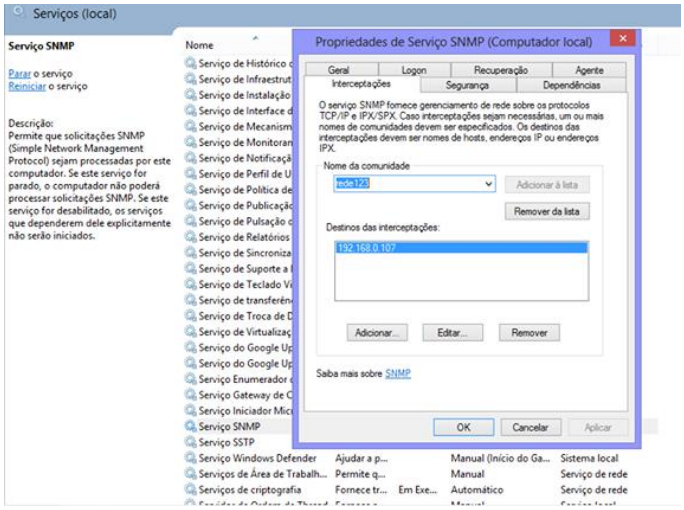

Figura 11- Configuração da comunidade

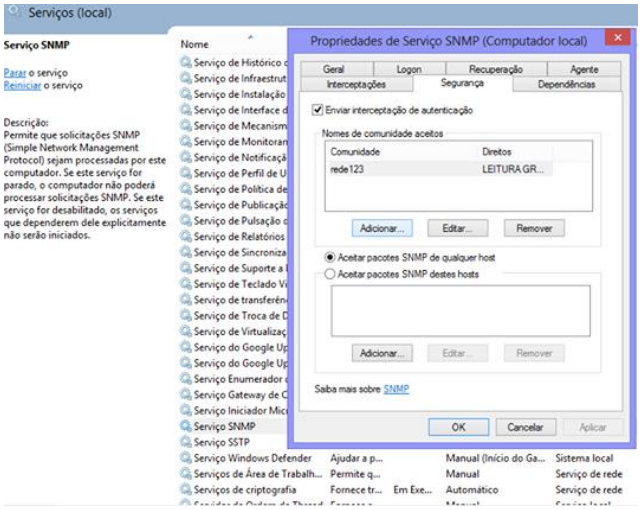

Figura 12 - Configuração do serviço SNMP

APÊNDICE G

A configuração no agente, foi realizado no Ubuntu, a ativação do SNMP no Linux vale tanto para o agente como para o gerente, se o serviço não estiver ativo no servidor não realizará o envio das mensagens. No Linux instale o serviço SNMP através do comando,

apt-getinstallsnmpd, apague o conteúdo do arquivo de configuração do serviço através do comando, # echo> /etc/snmp/snmpd.conf, edite o arquivo de configuração snmpd.conf utilizando o vi, # vi /etc/snmp/snmpd.conf.

Segue abaixo um arquivo "snmpd.conf" totalmente funcional:

rocommunity rede123

syslocationprojeto_final

sysContactprojeto_final<projeto.final@gmail.com>

Em nosso exemplo, foi utilizado o nome de comunidade rede123, a localização projeto_final e a pessoa de contato projeto_final.

Reinicie o serviço SNMP através do comando abaixo:

/etc/init.d/snmpdrestart

Instale e execute o utilitário rcconf, marque para que o serviço SNMP seja iniciado automaticamente durante o boot:

apt-getinstallrcconf

rcconf

O serviço SNMP foi instalado em seu sistema com sucesso.

APÊNDICE H

Segue a instalação do Python utilizando o pyenv.

A maioria das distribuições linux ja vem o o python instalado, mas a versão nem sempre é atualizada, pra instalar a nova versão voce pode utilizar o gerenciador de pacotes do Linux ou o pyenv, que é um gerenciador de versoes do Python, que permite que instale e gerencie varias versões do Python e manter essas versões isoladas umas das outras.

Para verificar a versão instalada do python digitar:

python -v

Para começar vamos atualizar o apt-get e instalar as dependências:

sudo apt-get update && sudo apt-get upgrade

$ sudo apt-get install -y make build-essential libssl-dev zlib1g-dev libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm libncurses5-dev git

Instaladas as dependências vamos passar a instalar o pyenv. Na página do projeto (https://github.com/yyuu/pyenv) existe um script de instalação que podemos chamar assim:

sudo curl -L https://raw.githubusercontent.com/yyuu/pyenv-installer/master/bin/pyenv-installer | bash

Ao final da instalação o script apresenta um aviso orientando a adicionar o pyenv ao load path editando seu arquivo ~/.bash_profile e inserindo as linhas apresentadas logo abaixo ao aviso. Podemos fazer isso no .profile ou no .bash_profile ou então no .bashrc, as linhas são:

export PATH="$HOME/.pyenv/bin:$PATH"

eval "$(pyenv init -)"

eval "$(pyenv virtualenv-init -)"

Abra seu editor de textos favorito e cole estas linhas no final do seu ~/.profile (ou um dos outros citados), depois salve e feche. No terminal vamos reiniciar, para ter efeito a mudança execute:

source ~/.profile ou source ~/.bash_profile ou source ~/.bashrc

Conforme o arquivo que você editou.

Para testar é só chamar o pyenv no terminal: pyenv

pyenv install 3.5.1 (Para instalar o Python na versão 3.5.1)

pyenv global 3.5.1 (Para tornar essa versão a principal)

APÊNDICE I

Esta seção descreve a instalação do PySNMP, Kivy e Alchemy.

A instalação da biblioteca PySNMP é simples, no terminal do Linux execute o comando:

$ Pip instalar pysnmp

No entanto, o "pip" no Windows geralmente falha para instalação no Windows requer que se instale o "PyCrypto", o link para download esta no site oficial do Python[4].

Estes são os comandos necessários para se instalar o Kivy no Linux:

Atualize o repositório:

$ sudo add-apt-repository ppa:kivy-team/kivy

$ sudo add-apt-repository ppa:kivy-team/kivy-daily

$ sudo apt-get update

Em seguida:

$ sudo apt-get install python3-kivy

Para instalação do SQLAlchemy:

$ sudo apt-get install python-sqlalchemy