UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
DEPARTAMENTO DE ENGENHARIA DE COMUNICAÇÕES
CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES
TRABALHO DE CONCLUSÃO DE CURSO

ANDERSON CLAUDIO RODRIGUES DA SILVA

UMA PROPOSTA PARA A IMPLEMENTAÇÃO DE CMDB'S NA UFRN COM FOCO NO GERENCIAMENTO DE CONFIGURAÇÃO DO SERVIÇO DHCP.

NATAL
2020

ANDERSON CLAUDIO RODRIGUES DA SILVA
anderson.acrs@gmail.com

UMA PROPOSTA PARA A IMPLEMENTAÇÃO DE CMDB'S NA UFRN COM FOCO NO GERENCIAMENTO DE CONFIGURAÇÃO DO SERVIÇO DHCP.

Defesa de Trabalho de Conclusão de Curso apresentado à banca examinadora do Curso de Engenharia de Telecomunicações (CETEL) da Universidade Federal do Rio Grande do Norte (UFRN), como requisito para obtenção do título de Engenheiro de Telecomunicações.

Orientador: Prof. Marcos Cesar Madruga Alves Pinheiro

NATAL
2020

AGRADECIMENTOS

Inicialmente agradeço a Deus por derramar suas bênçãos em minha vida, pela minha saúde e por me proteger todos os dias, desde o amanhecer até o anoitecer. Pelo sentimento de respeito e obediência que ele põe em meu coração. Baseado na história de Cristo, sei que sendo obediente a vontade do Pai, tendo determinação, e o mais importante, tendo a consciência tranquila, posso conseguir minhas vitórias sem passar por cima do caráter de ninguém. Acredito que pequenos gestos assim são a minha colaboração para mudar o mundo, tal como está escrito no livro de João 16:33; Salmos 51:10-12 e Gálatas 2:20.

Agradeço a minha família, minha esposa Rafaela Alves, por sempre estar presente em minha vida, me incentivando e sendo paciente pelos vários momentos nos quais eu precisava estar presente e não estava. Sem você em minha vida esse momento não seria possível. Agradeço a minha filha, Alana Gabriela, por ser o motivo que tenho para acordar todos os dias, incansavelmente, e fazer o melhor que eu puder.

Aos meus pais, Maria Alice e Claudio Rodrigues, por terem sacrificado muito de suas vidas para poder me fornecer uma educação de qualidade. Minhas irmãs, Andressa e Andréia, pelo apoio, carinho e respeito que sempre tiveram comigo.

Agradeço ao pessoal da Superintendência de informática da UFRN. A Denyson Falcão (que foi meu primeiro tutor e amigo na SINFO, desde a primeira bolsa de apoio técnico), a Luis Fhelipe, Edivaldo Cavalvante e Avando Campos pelo apoio. Agradeço especialmente ao funcionário Manoel Bezerra pela imensa colaboração neste trabalho. Ele que apadrinhou esse projeto como se fosse uma de suas demandas pessoais e cedeu seu tempo para me ajudar, inclusive finais de semana, muitas noites e algumas madrugadas. Ele que, apesar do seu cargo de trabalho não ser programador propriamente dito, aceitou estudar junto comigo e evoluir os conhecimentos para que esse trabalho tomasse forma e ser o pioneiro de outros projetos que estão por vir. Agradeço ainda ao Professor Marcos Madruga, diretor de redes da SINFO e professor do DIMAp/IMD, que me propôs esse desafio e me orientou bastante para finalizar este trabalho.

Agradeço aos meus amigos do POP-RN e CAIS-RNP pelo carinho e motivação. Meus agradecimentos a Nicole Rieckmann, Eduardo Rocha, Kleydson Wilbert, Thiago Dantas e Inácia Fernandes. Em especial aos professores Edson Moreira e Sérgio Fialho, por sempre contribuírem na minha formação, desde a bolsa de apoio técnico até este momento.

Agradeço aos professores da Escola de Ciência e Tecnologia, na pessoa do professor José Henrique Fernandez, que nunca deixou de me apoiar e sempre me incentivou a “olhar mais longe”. Aos professores do Departamento de engenharia de telecomunicações (CETEL/CT), que contribuíram para minha formação acadêmica. Em especial agradeço aos professores Vicente Ângelo, Márcio Eduardo, Claudio Muniz, Hertz Wilton e ao professor Antônio Campos, por cobrar exaustivamente de seus alunos extraindo o melhor de cada um deles para que se tornem bons profissionais no futuro.

Agradeço aos meus amigos do curso de engenharia de telecomunicações, pelos momentos bons e difíceis que passamos ao longo desses anos. Em especial a Helton Campos e Ana Roberta, por dividirem não só sua vida acadêmica mas também alguns momentos pessoais comigo.

A todos que contribuíram direta ou indiretamente para a construção deste trabalho, os meus mais sinceros agradecimentos. Sem o apoio e a colaboração de vocês eu não teria conseguido.

“Se consegui enxergar mais longe, foi porque me apoiei nos ombros de Gigantes”

(Isaac Newton)

“Prosperei com o suor do meu trabalho, me guardei, lutei sem buscar atalho..., minha cor não me atrapalhou, só me abençoou, quem falou que era moda, hoje felizmente se calou.

(Projota)

RESUMO

Este trabalho traz um estudo sobre a implementação de um CMDB na UFRN, para o uso dos setores da SINFO e POP-RN, bem como a proposta para a implementação de um modelo no gerenciamento de configuração do serviço DHCP através do desenvolvimento de um plugin. Este plugin é uma forma de customização para um CMDB desenvolvido com o auxílio do framework Django e com linguagem de programação Python. Ao final do trabalho o código do plugin será disponibilizado, por meio da instalação “pip”, para a comunidade para ser uma ferramenta de uso nas instituições que despertem o interesse.

Palavras-chave: CMDB; DHCP; Framework; Plugin; Python.

ABSTRACT

This work brings a study on the implementation of a CMDB at UFRN, for the use of the SINFO and POP-RN sectors, as well as a proposal for the implementation of a DHCP configuration management model through the development of a plugin. This plugin is a form of customization for a CMDB developed with the help of the Django framework and Python programming language. At the end of the work, the code of plugin will be made available, through the “pip” command, to the community to be a tool for use in institutions that awaken any interest.

Keywords: CMDB; DHCP; Framework; Plugin; Python.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ACL	Lista de Controle de Acesso
API	Interface de Programação de Aplicações
APIC	Interface Programática de Controle de Aplicações
CIM	Modelo Comum de Informação
CMDB	Banco de Dados de Gerenciamento de Configuração
DBR	Banco de Dados Relacional
DHCP	Protocolo de Configuração Dinâmica de Hosts
DMTF	Força Tarefa de Gerenciamento Distribuído
DNS	Serviço de Resolução de Nomes
GUI	Interface Gráfica do Utilizador
HTTP	Protocolo de Transferência de Hiper Texto
IC	Itens de Configuração
IETF	Grupo Internacional de Engenheiros da Internet
IP	Protocolo de Internet
IPAM	Gerenciamento de endereços IP
JSON	Notação de objeto em JavaScript
MOF	Formato de Objeto Gerenciado
NURA	Núcleo de Redes Avançadas
POO	Programação Orientada a Objetos
POP-RN	Ponto de Presença da RNP no Rio Grande do Norte
PSF	Fundação do Software Python
REST	Transferência Representacional de Estado
RFC	Pedido de comentários
RGM	Rede Giga Metropole
RNP	Rede Nacional de Ensino e Pesquisa
SaaS	Software As A Service
SDN	Rede Definida por Software
SINFO	Superintendência de Informática da UFRN
SOAP	Protocolo Simples de Acesso a Objetos
TI	Tecnologia da Informação
TIC	Tecnologia da Informação e Comunicações
UFRN	Universidade Federal do Rio Grande do Norte
UML	Linguagem de Modelagem Unificada
VPN	Rede Privada Virtual
XML	Linguagem de Marcação
XOS	Sistema Operacional Extreme

SUMÁRIO

1. INTRODUÇÃO

Uma problemática discutida atualmente em grande parte das instituições públicas e/ou privadas, no setor de telecomunicações, especificamente na área de redes de computadores é a forma de como são guardadas as informações de configuração dos ativos de rede, pois elas precisam ser confiáveis e representar o estado do seu parque de Tecnologia da Informação e Comunicações (TIC). Quando os administradores de rede realizam a configuração de seus Switches e/ou máquinas virtuais essas informações são guardadas em ferramentas de documentação Web ou Scripts de configuração ou no próprio arquivo de configuração salvo pelo dispositivo. Essas informações precisam ser armazenadas em um local apropriado, tal como um banco de dados. Com o fim de guardar todas as informações importantes de configuração de um parque tecnológico esse banco de dados torna-se um conceito de TI para o suporte eficiente de configuração e serviços, passando a ser chamado de banco de dados de gerenciamento de configuração (Configuration Management Data Base - CMDB).

Um CMDB pode ser definido como um repositório de informações que estão relacionadas aos ativos de rede de um sistema de informação, contendo detalhes da infraestrutura de rede e dos itens de configuração. Ou seja, o CMDB fornece um único ponto de referência, tornando-o o mecanismo definitivo para a tomada de decisões de um ambiente de TI, nas dependências entre processos de negócios, usuários, aplicativos e Infraestrutura de TI e serviços.

Além de serem fundamentais para sustentar o gerenciamento de serviços de TIC conforme recomendado pelas boas práticas definidas na ITIL V4(2019), os CMDB's também podem auxiliar na automatização da configuração remota dos ativos de rede, máquinas virtuais, atendimento de requisições, tratamento de incidentes e afins, em um ambiente de rede bem definido não obstante a um SDN. No que se refere em gerenciamento de serviços, um dos serviços mais utilizados nas instituições é o serviço de Dynamic Host Control Protocol (DHCP).

O DHCP é um protocolo de configuração dinâmica de hosts, que fornece parâmetros de configuração para hosts da Internet. O DHCP consiste em dois componentes: um protocolo para fornecer parâmetros de configuração específicos do host e um mecanismo para alocação de endereços de rede aos hosts (RFC 2131).

O DHCP é construído em um modelo cliente-servidor, onde hosts de servidores DHCP designados alocam endereços de rede e fornecem parâmetros de configuração para hosts configurados dinamicamente (RFC 2131). Esse serviço gera muita informação de entrada e saída como Mac Address, Prefix, endereços IP e afins. Isso pode dificultar o trabalho de um administrador de redes, caso ele não tenha uma documentação bem detalhada. Neste sentido o CMDB é uma ferramenta fundamental que auxiliará a gerenciar as informações das redes, advindas do DHCP.

No entanto, por mais que os CMDB's sejam muito bons, muitas vezes eles não se adequam 100% às necessidades da instituição e precisam ser customizados. Como no exemplo supracitado talvez o CMDB não traga, em sua construção, os parâmetros necessários para armazenar todas as informações de configuração de um serviço DHCP. Nesse sentido faz-se necessário buscar ferramentas que, aliada a uma interface amigável (GUI), busquem as informações necessárias dentro do CMDB e as apresentem de forma fácil ao administrador ou analista de redes, auxiliando no trabalho e fornecendo uma base de configuração sólida e que mais tarde possa se tornar a base para ferramentas de automação.

Isto posto, um dos objetivos deste trabalho é desenvolver e apresentar um modelo de gerenciamento das configurações de um serviço DHCP, de forma customizada e que atenda às necessidades de um levantamento de requisitos feito na SINFO, bem como servir como uma base sólida de configuração e uma base para automação de configuração futura.

1.1. Objetivo Geral.

Propor a customização de um CMDB para gerenciar o serviço de DHCP na UFRN.

1.2. Objetivos específicos.

Este trabalho tem o objetivo de comparar as diferentes formas de construção de um CMDB, bem como a utilização e customização de um CMDB que cumpra os requisitos definidos pela Superintendência de Informática da UFRN(SINFO) e pelo Núcleo de Redes Avançadas (NURA). Avaliar a pertinência da utilização de um CMDB para armazenar e gerenciar as informações do parque de TIC da UFRN e desenvolver uma ferramenta customizada (Plugin) para auxiliar no gerenciamento de tais informações.

1.3. Motivação e justificativa

Para poder gerenciar as configurações de um serviço DHCP é preciso que todas as informações estejam em um local que possa fornecer ao administrador de redes uma visualização rápida, disponibilidade e agilidade para tratar esses dados. Por esse motivo a utilização de ferramentas de documentação e configuração estáticas ficam cada vez mais inviáveis, tal como é o caso da ferramenta de documentação WikiDoku, que não dispõe de integração com outros sistemas e torna a visualização dos dados mais lenta, consequentemente a tomada de decisão diante de um problema também. Atualmente grande parte das informações de configuração dos Switches, Vm’s e serviços da UFRN estão em ferramentas como o WikiDoku e, nesse sentido faz-se necessário o estudo de novas tecnologias que nos garanta uma visualização mais fácil de informações, distribuídas em várias bases de dados, em um só local. Nesse sentido, ter um CMDB que possa integrar-se com ferramentas de monitoramento e ativos de Rede, por meio de uma interface de programação para aplicações baseada na transferência do seu estado representacional (API Rest), por exemplo, irá tornar o trabalho mais fácil e agregar valor na tomada de decisão dos administradores de rede.

1.4. Metodologia

Este estudo está dividido em três momentos distintos: o Primeiro consiste no estudo dos métodos e conceitos de um sistema de gerenciamento de banco de dados de configuração. Essa etapa consiste em fazer a instalação das ferramentas, em um ambiente de testes e desenvolvimento, alimentando-o com informações reais, ou seja,

que representem situações cotidianas de um administrador de redes, frente às configurações dos serviços e ativos de rede. Essa etapa visa analisar a versatilidade e estabilidade dos CMDB's.

O segundo momento consiste na elaboração de um checklist com os requisitos que atendem as necessidades da SINFO e do NURA e avaliar qual dos CMDBs atendem aos requisitos estabelecidos pelos setores supracitados.

O Terceiro momento, após eleger um CMDB, é customizá-lo de acordo as demais informações do levantamento de requisitos. Para isso faz-se necessário desenvolver um código a parte do código principal (core) no qual seja possível complementar os requisitos exigidos. Isso será possível por meio do desenvolvimento de um plugin, cujo qual é um programa de computador (módulo e/ou extensão) usado para adicionar dados e/ou funções a outros programas de maior porte, provendo funcionalidades especiais e bem específicas [Cambridge's Dictionary].

1.5. Estrutura do trabalho

O capítulo 2 define alguns conceitos utilizados neste estudo, tais como: o que é um CMDB's?; o modelo comum de informação e sobre a interface de programação de aplicação com base na representação de estados. O capítulo 3 mostra um estudo sobre alguns CMDB's estudados e como os mesmos atenderam aos pré-requisitos definidos. O capítulo 4 mostra o desenvolvimento da customização do CMDB que atendeu aos pré-requisitos, definidos no capítulo anterior, para gerenciar o serviço DHCP. Capítulo 5 apresenta os resultados da customização proposta e o capítulo 6 apresenta as conclusões e possibilidades de trabalhos futuros.

2. A CONSTRUÇÃO DE UM CMDB.

Este capítulo apresenta a definição de um CMDB e alguns itens de sua construção para o gerenciamento de ativos e serviços.

2.1. O CMDB

Assis (2020), define em seu artigo que um CMDB é um banco de dados (não no sentido técnico da palavra, mas um modelo de TI para o suporte eficiente dos serviços), que tem como objetivo organizar e gerenciar itens de configuração (IC). Ou seja, qualquer objeto de negócio de uma empresa é necessário para a entrega bem-sucedida de um serviço, como telefone celular, notebook, contrato, veículo etc. O CMDB armazena os nomes de todos eles, além de detalhes e atributos sobre os ICs. Ele também armazena informações sobre os relacionamentos entre vários ICs, serviços e processos.

Já O'Donnell (2009), cita em sua obra (2009), que o CMDB é um termo que está intimamente relacionado com a Biblioteca de Infraestrutura de TI (ITIL) e qualquer outro conceito de operações de TI. Na verdade, toda a noção de excelência operacional é uma panacéia sem um CMDB robusto servindo como sua base. A base de informações deve ser sólida e refletir fielmente as informações estão distribuídas. Essas informações são mais eficazes se estiverem disponíveis em uma fonte conhecida e confiável, que é definida como CMDB.

2.2. Modelo de arquitetura do CMDB

Esta sessão apresenta recursos técnicos que irão gerenciar os itens de configuração, atendidos por um sistema de gerenciamento de configuração, que são modelos lógicos para o gerenciamento de configurações e ativos.

2.2.1 CIM

O Common Information Model (CIM) é um modelo conceitual de informações que descreve entidades em ambientes de internet, incluindo modelos computacionais e modelo de negócios. Ele fornece uma definição sólida e uma estrutura de informações e gerenciamento utilizando técnicas de orientação a objetos. O CIM também inclui expressões para elementos comuns que devem ser apresentados de

forma clara aos aplicativos de gerenciamento. Dessa forma estão inclusas às classes, propriedades, associações e os métodos. Ele utiliza um conjunto de terminologia específica para o modelo da programação orientada a objetos (POO). Para definir esses elementos utiliza-se a extensão MOF (Managed Object Format), bem como a Unified Modeling Language (UML).

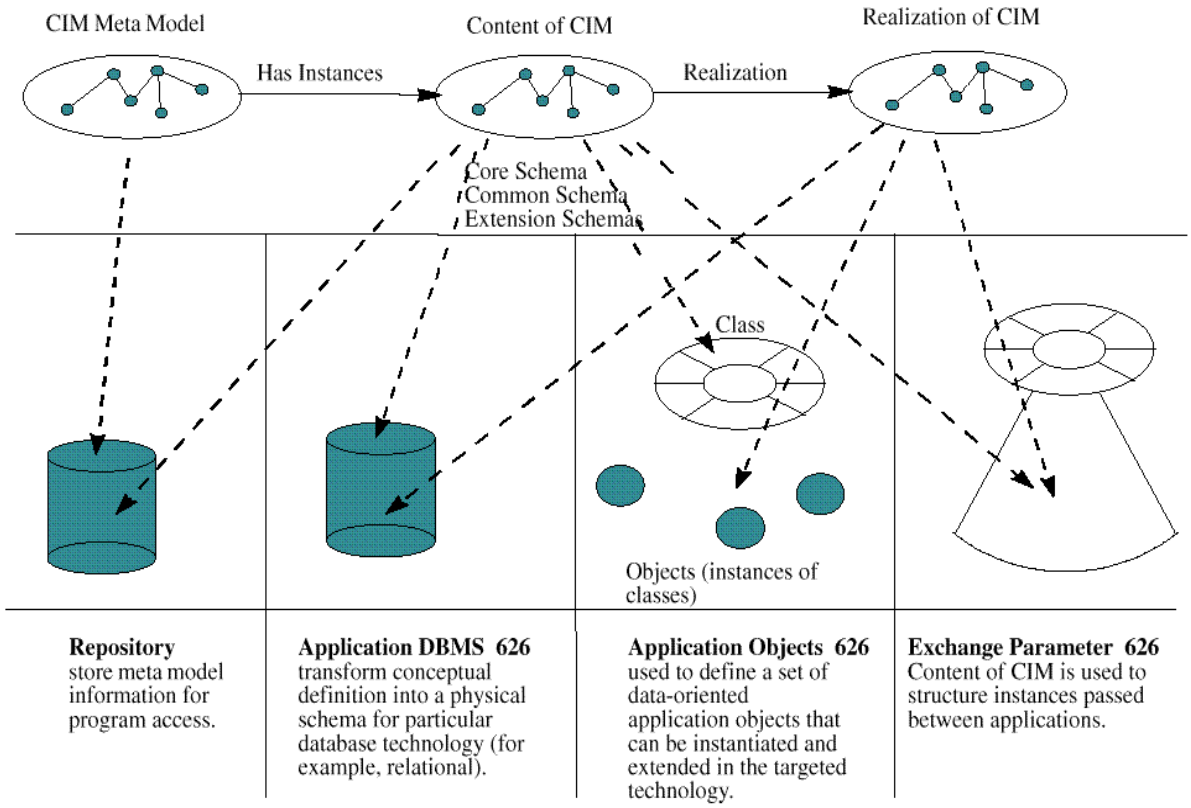
Sabendo dessas particularidades é importante reconhecer alguns aspectos da orientação a objetos, visto que a modelagem é diferente da programação orientada a objetos.

No que diz respeito à modelagem orientada a objetos:

- Instâncias são coisas.
- Propriedades são atributos.
- Relacionamentos são conjuntos de atributos.
- Classes são tipos de coisas.
- Subclasses são subtipos de coisas.

Percebe-se que o conceito de modelagem orientada a objetos não se limita aos elementos relacionados ao computador. Pode-se usar modelagem orientada a objetos para representar muitos tipos diferentes de coisas, estruturas, materiais orgânicos, edifícios físicos. No contexto do CIM a modelagem orientada a objetos é usada para modelar elementos de hardware e software DMTF (DMTF, 2019), conforme mostrado na Figura 1.

Figura 1 - Modelo conceitual de implementação CIM



Fonte.: https://pubs.opengroup.org/onlinepubs/9619599/chap1.htm

O CIM é uma arquitetura hierárquica e orientada a objetos que torna relativamente simples rastrear e descrever as interdependências e associações frequentemente complexas entre diferentes objetos gerenciados. Essas interdependências podem incluir conexões lógicas de rede e dispositivos físicos subjacentes, ou aquelas de uma transação de comércio eletrônico e os servidores da Web e de banco de dados dos quais depende.

2.3. API REST

Vincent [2020] cita que a representação de transferência de estado (REST) é uma arquitetura proposta no ano 2000 por Roy Fielding, em sua dissertação. É uma abordagem para a construção de API, no topo da web, ou seja, no topo do protocolo HTTP. Qualquer API Rest deve, no mínimo, obedecer a três princípios: 1) não ter estado, tal como o HTTP; 2) suportar os métodos comuns do HTTP (GET, POST, DELETE e PUT); 3) retornar os dados em ambos os formatos (JSON e XML).

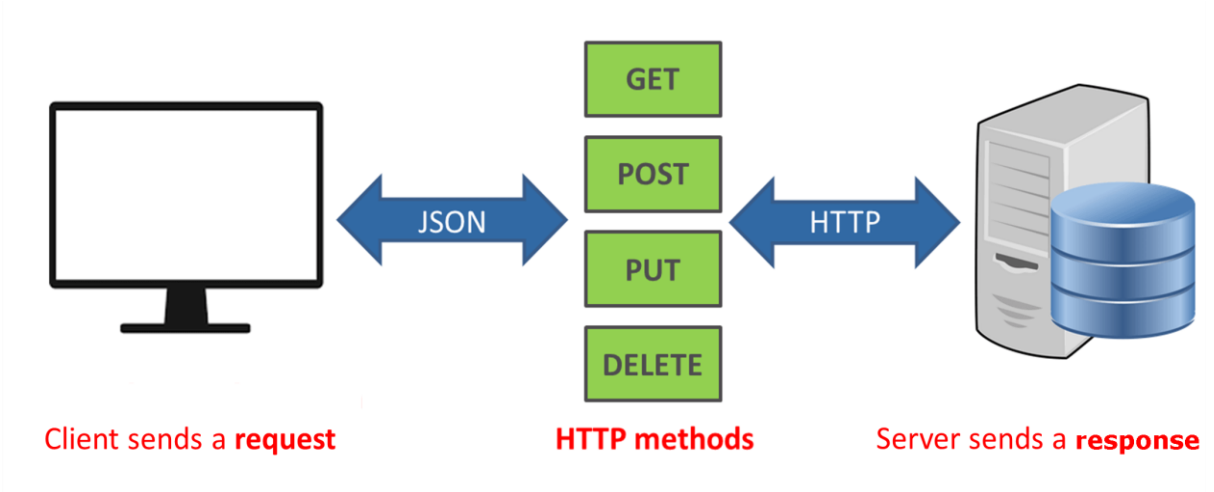
Com a evolução da tecnologia novas ferramentas baseadas na Web estão surgindo, muitas delas baseadas (REST).

Uma interface de programação de aplicativo REST (API REST) é um tipo de servidor da web que permite a um cliente, operado pelo usuário ou automatizado, acessar recursos que modelam os dados e funções de um sistema. (Massé, 2011). Segundo um artigo da empresa RedHat [1993-2020], uma API Rest, também chamada de API RestFul, é uma interface de programação de aplicações que segue a conformidade de restrições da arquitetura REST, que significa transferência representacional de estado.

Uma API é um conjunto de definições e protocolos para criar e integrar softwares de aplicações. REST não é um protocolo padrão, mas sim um conjunto de princípios de arquitetura.

Os desenvolvedores de API podem implementar a arquitetura REST de maneiras distintas. Quando uma solicitação é feita por meio de uma API RestFul ela transfere uma representação do estado do recurso do solicitante. Essa informação, ou representação, é fornecida utilizando vários formatos possíveis via HTTP: Java Script Object Notations (Json), HTML, XLT ou texto simples. O formato Json é o mais utilizado porque, apesar de seu nome, é independente de qualquer linguagem e pode ser lido por máquinas e humanos. A Figura 2 representa a comunicação entre a requisição do cliente e a resposta do servidor através dos métodos HTTP e Json.

Figura 2: Comunicação entre cliente e servidor



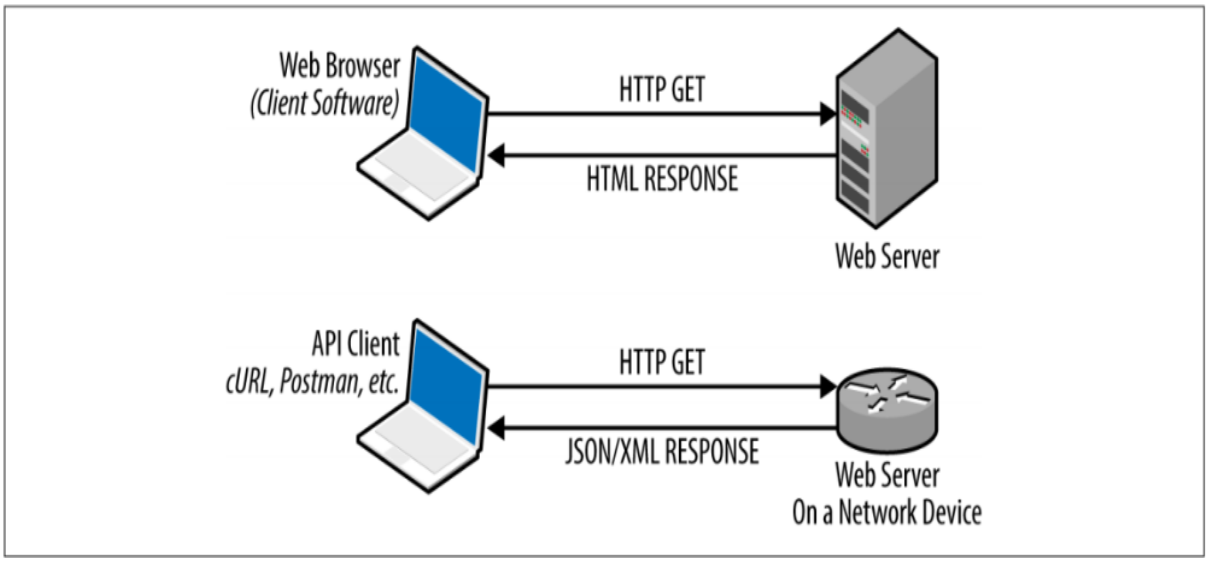
Fonte.: https://www.devopsschool.com/blog/restful-web-services-with-php-and-laravel/

Tanto a RESTfullness quanto a capacidade de codificar dados como JSON tornam o REST uma opção muito atraente para desenvolvedores de aplicativos.
Edelman [2018], cita que alguns exemplos de restrições abordadas no trabalho do Roy são descritas a seguir, na qual uma interface deve estar em conformidade para ser considerada RESTful.

- Cliente-Servidor: Este é um requisito para melhorar a usabilidade dos sistemas, simplificando os requisitos do servidor. Ter uma arquitetura cliente-servidor permite a portabilidade e a mutabilidade dos aplicativos clientes sem que os componentes do servidor sejam alterados. Isso significa que é possível ter diferentes clientes de API (interface da web, CLI) que consumam os mesmos recursos do servidor (API de back-end) [Edelman, 2018].
- Stateless: A comunicação entre o cliente e o servidor deve ser sem estado. Os clientes que usam formas de comunicação sem estado devem enviar todos os dados necessários para o servidor entender e executar as operações solicitadas em uma única solicitação. Isso contrasta com interfaces como SSH, nas quais há uma conexão persistente entre um cliente e um servidor [Edelman, 2018].
- Interface uniforme: Recursos individuais no escopo em uma chamada de API são identificados nas mensagens de solicitação HTTP. Por exemplo, nos sistemas baseados em HTTP RESTful, a URL usada faz referência a um recurso específico. No contexto da rede, o recurso é mapeado para uma construção de dispositivo de rede, como um nome de host, interface, configuração de protocolo de roteamento ou qualquer outro recurso existente no dispositivo. A interface uniforme também afirma que o cliente deve ter informações suficientes sobre um recurso para criar, modificar ou excluir um recurso [Edelman, 2018].

Na Figura 3, Roy demonstrou como seria a relação de uma aplicação Cliente-servidor para a recuperação de dados de um site e recuperação de dados de um dispositivo de rede por meio de uma API RESTful. Nos dois casos, uma solicitação HTTP GET é enviada ao servidor da web.

Figura 3 - Relação cliente servidor com Restconf



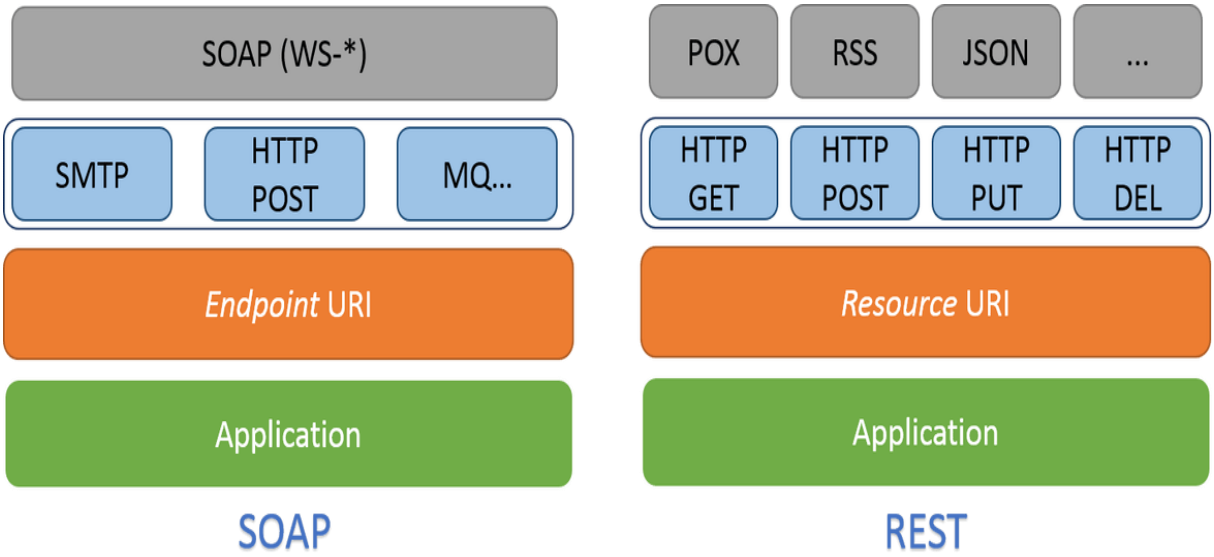
Fonte.: Edelman (2018)

Para que uma API seja considerada RestFul, ela precisa estar em conformidade com os critérios a seguir:

1. Ter uma arquitetura Cliente/Servidor formada por clientes, servidores e recursos, com solicitações gerenciadas por meio de HTTP.
2. Realizar comunicação cliente/servidor stateless. Isso significa que cada solicitação é separada e não conectada com outras, e que nenhuma informação do cliente é armazenada entre elas.
3. Armazenar dados em cache para otimizar as interações entre cliente e servidor.
4. Ter uma interface uniforme para que as informações sejam transferidas em um formato padronizado.
5. Ter um sistema em camadas que organiza os tipos de servidores envolvidos na recuperação das informações solicitadas em hierarquias que seja transparente ao cliente.

Embora uma API REST precise estar em conformidade com os critérios acima, ela é considerada mais fácil de usar que um protocolo prescrito, como um Protocolo Simples de Acesso a Objetos (SOAP). A Figura 4 apresenta uma comparação entre os modelos de construção e comunicação SOAP e REST.

Figura 4 – Método SOAP vs REST



Fonte.: https://www.thistechnologylife.com/soap-vs-rest/

2.4. YANG

O Yet Another Next Generation - YANG, cuja tradução é algo semelhante a “mais uma nova geração” é uma linguagem de modelagem de dados, definida na RFC 7950 (2016), para tratar a forma com que os dados são enviados, por meio de protocolos de gerenciamento de rede, como o NETCONF e o RESTCONF. A linguagem de modelagem de dados YANG é mantida pelo grupo de trabalho NETMOD na IETF e foi publicada com a RFC 6020 em outubro de 2010. Essa linguagem pode ser usada para modelar dados de configuração e/ou estado. Além disso o YANG pode ser usado para definir o formato das notificações de eventos emitidas pelos elementos de rede e permite que os modeladores de dados definam a assinatura de chamadas de procedimento remoto (RPC) que podem ser chamadas nos elementos de rede pelo protocolo NETCONF. A linguagem, pode ser implementada de forma independente ao protocolo de codificação (XML ou Json).

2.5. NETCONF

Este protocolo é definido pela IETF para "instalar, manipular e excluir a configuração dos dispositivos de rede" e define um mecanismo simples no qual os aplicativos de gerenciamento de rede, agindo como clientes, podem invocar operações nos dispositivos, que agem como servidores. A especificação NETCONF

define um pequeno conjunto de operações, mas evita quaisquer requisitos sobre os dados transportados nessas operações, preferindo permitir que o protocolo transporte quaisquer dados (Claise & Clark 2019).

O protocolo base NETCONF foi publicado oficialmente como protocolo de configuração RFC 4741 NETCONF no final de 2006 [RFC 4741]. O grupo de trabalho da IETF que produzia o padrão também produziu RFCs de suporte para vários mapeamentos de transporte, incluindo:

- RFC 4742 usando o protocolo de configuração NETCONF sobre Secure SHell (SSH);
- RFC 4743 usando NETCONF sobre o Simple Object Access Protocol;
- RFC 5539 NETCONF sobre TLS (Transport Layer Security).

As operações NETCONF são realizadas na parte superior de uma camada RPC (Remote Procedure Call) usando uma codificação XML e fornece um conjunto básico de operações para editar e consultar a configuração em um dispositivo de rede [RFC 4741].

Em suma, o NETCONF é um protocolo que funciona com RPC e realiza seus acessos por meio do SSH, SOAP além de realizar o transporte dos dados com o TLS. Ou seja, o NETCONF é definido para a configuração segura de transações de dispositivos. Isso significa que cenários como a configuração inicial de uma variedade de dispositivos, alterando ACLs e adicionando VPNs.

2.6. RESTCONF

RESTCONF é um equivalente funcional muito próximo do NETCONF. Em vez do SSH, o RESTCONF depende do HTTP para interagir com os dados de configuração e o estado operacional do dispositivo de rede e codifica todos os dados trocados em XML ou JSON. O RESTCONF empresta a ideia de operações CRUD em recursos do REST e os mapeia para modelos e Data Stores YANG. Há um relacionamento direto entre operações NETCONF e os métodos HTTP RESTCONF. A Figura 5 apresenta a relação entre os métodos Restconf e Netconf.

Figura 5 – Relação entre os métodos Restconf e Netconf.

HTTP VERB	NETCONF OPERATION
POST	create
PUT	replace
PATCH	merge
DELETE	delete
GET	get/get-config

Fonte.: Edelman (2018)

2.7. Diferença entre estilos de arquitetura

A Tabela 1 faz a comparação entre as arquiteturas, comparando características comuns entre elas.

Tabela 1 – Comparativo entre arquiteturas

	REST API	NETCONF API	RESTCONF API
Métodos de dados definidos pelo Fabricante	Utiliza métodos Http (Get, Delete, Patch, Post and Put)	Usa mensagens RPC & operações.	Utiliza métodos Http (Get, Delete, Patch, Post and Put)
Representação dos dados	Suporta dados codificados em XML ou Json.	Suporta codificação em XML	Suporta dados codificados em XML ou Json.
Modelo de Dados	Utiliza modelos de dados YANG, XSD, JSD ou um modelo de linguagem customizado.	Usa modelo de dados Yang ou XSD	Usa o modelo de dados YANG
Transporte	Usa HTTP/HTTPS para se comunicar	Usa SSH/TLS para a comunicação	Usa HTTP/HTTPS para se comunicar
Consumo da API	Utiliza os comandos cURL, request python library, postman (API Client)	Utiliza ncclient python tool e SSH do CLI	Utiliza os comandos cURL, request python library, postman (API Client)

Fonte.: https://yasseraudalab.com/2020/07/24/rest-vs-netconf-vs-restconf-apis/

2.8. Fabricantes e modelos de switches com suporte ao Rest API

Considerando o exposto do item anterior definiu-se optar pela API Rest, pois é uma arquitetura mais nova que o Netconf e o Restconf, herdando características de ambos. Possui uma comunidade de desenvolvedores em ascensão, ou seja, o suporte a problemas fica mais simples de ser resolvido. Três fabricantes de ativos de rede, existentes na UFRN, foram escolhidos para serem estudados, os quais possuem suporte a API Rest a partir de versões específicas do Firmware dos seus dispositivos e são detalhados a seguir:

2.8.1 ARUBA

O HPE Aruba OS-Switch REST API foi introduzido nos dispositivos a partir da versão 16.08 e conta com um suporte de comunicação HTTP, HTTPS. A estrutura do documento guia para realizar os procedimentos REST está bem documentada e é de fácil compreensão [HP 1939-2020], o exemplo de acesso aos modelos a seguir foi retirado do manual de utilização da HP.

```
WorkStation# curl --noproxy10.100.167.104 -X POST
http://10.100.167.104:80/rest/v1/login-sessions -d '{"userName":"test",
"password":"test"}'
```

Response from the switch when user validation is successful:

```
{
"uri": "/rest/v1/login-sessions","cookie":
"sessionId=09CG1bRuT5hkCPzI97mmDjpn4uLtsmgkBsAaWUr9h7GxlkbsiASak1PEyj7Ov3n"
}
```

Fonte.: https://support.hpe.com/hpesc/public/docDisplay?docId=a00057978en_us

2.8.1.2 Plataformas suportadas

Os modelos de Switches a seguir possuem suporte a API REST:

- Aruba 2530 Switch Series (J9772A, J9773A, J9774A, J9775A, J9776A, J9777A, J9778A, JL070A, J9853A, J9854A, J9855A, J9779A, J9780A, J9781A, J9782A, J9783A)
- Aruba 2540 Switch Series (JL354A, JL355A, JL356A, JL357A)
- Aruba 2620 Switch Series (J9623A, J9624A, J9625A, J9626A, J9627A)
- Aruba 2920 Switch Series (J9836A, J9726A, J9727A, J9728A, J9729A)

- Aruba 2930F Switch Series (JL253A, JL254A, JL255A, JL256A, JL258A, JL259A, JL260A, JL261A, JL262A, JL263A, JL264A)
- Aruba 3800 Switch Series (J9573A, J9574A, J9575A, J9576A, J9584A)
- Aruba 3810 Switch Series (JL071A, JL072A, JL073A, JL074A, JL075A, JL076A)
- Aruba 5400R zl2 Switch Series (J9821A, J9822A, J9850A, J9851A, JL001A, JL002A, JL003A, JL095A)

2.8.2 EXTREME

A implementação REST nos dispositivos com o Extreme XOS segue o que foi definido na RFC 8040, sendo adicionado a partir da versão EXOS 22.4.1[Extreme, 2018]. Na documentação do Extreme XOS, a EXTREME disponibiliza gratuitamente scripts, na linguagem Python, para que seja possível que os programadores implementarem suas soluções com o conjunto de produtos da Extreme. No entanto, os scripts são apenas disponibilizados e não são suportados nem mantidos. Para contornar essa situação alguns programadores realizaram um Fork do código, os quais podem ser acessíveis por meio repositório Github [github – extremeapp].

2.8.2.1 Plataformas Suportadas

Os modelos de Switches a seguir possuem suporte a API REST:

- Switches Summit X450-G2, X460-G2, X670-G2, X770
- Extreme Switching, Switches das séries X440-G2, X870, X620, X690.

2.8.3 CISCO

A partir do IOS XE Everest 16.6.1, esse recurso foi introduzido nos roteadores de serviços de agregação ASR 1000-ASR1001-HX e ASR1002-HX, roteador de serviços em nuvem da série CSR 1000v e roteadores de serviços integrados da série Cisco 4000 (ISRs) [NetWorkop].

A API REST do Application Policy Infrastructure Controller (APIC) é uma interface programática que usa a arquitetura REST. A API aceita e retorna mensagens HTTP (não habilitado por padrão) ou HTTPS que contêm documentos JavaScript Object Notation (JSON) ou Extensible Markup Language (XML). O programador pode

usar qualquer linguagem de programação para gerar as mensagens e os documentos JSON ou XML que contêm os métodos de API ou descrições de objeto gerenciado (MO). [CISCO, 1984-2020]

2.9. PYTHON

A linguagem Python foi escolhida para este trabalho por ser uma linguagem de programação de alto nível, multiplataforma, orientada a objetos e de tipagem dinâmica e forte, que tem se mostrado compacta, legível, versátil, possuir compatibilidade com outras linguagens e ferramentas web e por ter sua curva de aprendizagem suave [Linge, 2016]

Segundo Zuliani [2019], a Linguagem Python foi concebida no fim dos anos 80, pelo matemático Guido van Rossum, enquanto ele trabalhada na CWI (Centrum Wiskunde & Informática, Centro de Matemática e Ciência da Computação) em Amsterdã, Holanda. Ele trabalhava no desenvolvimento da linguagem ABC, mas com o fim da linguagem, Guido foi transferido para um outro grupo que trabalhava em um sistema operacional chamado Amoeba. Percebeu então que precisava de uma linguagem para escrever programas intermediários, algo entre o C e o Shell Script. Em 1989 a linguagem Python teve seu início, tendo como base a linguagem ABC e em 1990 já possuía uma versão mínima e operacional, sendo que no final daquele ano, Python já seria mais utilizada que a linguagem ABC no CWI.

Em março de 1993 surgiu a primeira comunidade Python e em 1994 a comunidade, comum certo receio da linguagem Python desaparecer, pois até então era mantida por “um homem só”, decidiram que era necessário criar uma organização responsável pelo Python.

Em dezembro de 2008 é lançada a versão 3 do Python, mas ainda manteve muitos adeptos na versão 2, tanto que é possível trabalhar até hoje com a versão 3 (**3.9.0** é a última versão utilizada neste trabalho) como na versão 2 (**2.7.18**), apesar desta última não ser mais recomendada para novos projetos.

2.9.1 Licenciamento

Sua licença foi renomeada para Python Software Foundation License. Todo código, documentação e especificação desde o lançamento da versão alfa da 2.1 é propriedade da Python Software Foundation (PSF), uma organização sem fins lucrativos fundada em 2001, um modelo tal qual da Apache Software Foundation.

2.10 O DJANGO

Django é um framework web Python de alto nível que permite o rápido desenvolvimento de sites seguros e de fácil manutenção. Construído por desenvolvedores experientes, o Django cuida de grande parte do trabalho de desenvolvimento web, para que seja possível se concentrar em escrever seu aplicativo sem muito esforço. É gratuito e de código aberto, tem uma comunidade próspera e ativa, ótima documentação e muitas opções de suporte gratuito e pago. [MDN Web Docs, 2020]

Django foi inicialmente desenvolvido entre 2003 e 2005 por um time de desenvolvedores web que era responsável por criar e manter sites de jornal. Depois de criar um número de sites, o time começou a fatorar e reutilizar muitos de seus códigos comuns e padrões de design. Esse código comum evoluiu para um framework genérico de desenvolvimento web, que foi lançado como um projeto de código aberto nomeado "Django" em Julho de 2005.

Django continua a crescer e aprimorar, desde seu lançamento (1.0) em setembro de 2008 até a versão mais estável lançada em 2017, a Versão 2.0. Cada lançamento adicionou novas funcionalidades e consertou falhas, variando entre suportar novos tipos de banco de dados, mecanismos de template e caches, até a adição de funções view "genéricas" e classes (que reduzem a quantidade de código que os desenvolvedores tem que escrever para um número de tarefas de programação [MDN Web Docs, 2020].

3. CMDB’s estudados

Este capítulo apresenta um estudo de alguns CMDB’s que atenderam Pré-requisitos definidos:

3.1 Levantamento de pré-requisitos

No início deste estudo alguns CMDB's foram avaliados, tais como: iTOP, CMDBuild, OneCMDB, e Netbox. Os CMDB's deveriam satisfazer os seguintes critérios para serem analisados:

- Ser Open source.
- Ser customizável.
- Programabilidade e escalabilidade.
- Utilizar REST/JSON API.
- Utilizar DBR.
- Documentação acessível.
- Integração com outras bases de dados.
- Possuir GUI "amigável".
- Suportar o desenvolvimento e módulos.
- Comunidade e suporte.
- Objetividade.
- Segurança.
- Curva de aprendizado suave.

As definições, vantagens e desvantagens de cada CMDB são explanadas a seguir.

3.2. iTOP

iTop significa Portal Operacional de TI. O iTop é um aplicativo da Web de código aberto para as operações diárias de um ambiente de TI. O iTop foi projetado com as melhores práticas da ITIL em mente, mas não determina nenhum processo específico; o aplicativo é flexível o suficiente para se adaptar aos seus processos, se você deseja processos informais e pragmáticos ou um comportamento estrito alinhado à ITIL.

No centro do iTop está o CMDB, esta é originalmente a primeira parte do iTop que foi desenvolvida. Em seguida foram desenvolvidos os tickets e todos os processos derivados.

A crença por trás do iTop é que um CMDB deve ser uma ferramenta operacional. A única maneira de um CMDB ser preciso e atualizado é ser usado diariamente pelas equipes de TI (agentes de suporte, engenheiros de TI etc) [ProgSoft].

3.2.1. Vantagem.

O iTOP possui suporte e condições para ter uma Implantação em nuvem, possui o conceito de Software como um Serviço (SaaS), totalmente baseado em Web.

3.2.2. Desvantagem.

Possui um módulo pronto (pago) para o IP Address Management (IPAM). O ITOP possui uma versão gratuita, mas não oferece versão de teste grátis. Recursos como: Descoberta automática de dispositivos, gestão de licenças, mapeamento de relacionamentos, monitoramento de desempenho e visualização de dados são recursos inexistentes na versão free, estando apenas na versão paga.

Fonte.: [https://www.capterra.com.br/software/163429/itop]

3.3. CMDBuild

O CMDBuild é um ambiente da web de código aberto para a configuração de aplicativos personalizados para o Gerenciamento de ativos. Por um lado, fornece mecanismos nativos para o administrador, implementados em um código "central" que foi mantido separado da lógica de negócios, para que o sistema possa ser configurado com todos os seus recursos [CMDBuild, 2020].

Por outro lado, gera dinamicamente uma interface da web para os operadores, para que eles possam manter a situação do ativo sob controle e sempre conhecer suas características, configuração, dependência, relações funcionais e como eles são atualizados, a fim de gerenciar seu ciclo de vida em um ambiente. O administrador do sistema pode criar e estender seu próprio CMDB (daí o nome do projeto), modelando o CMDB de acordo com as necessidades da empresa; uma interface adequada permite adicionar progressivamente novas classes de itens, novos atributos e novas relações. Filtros podem ser definidos, "visualizações" e permissões de acesso limitadas a linhas e colunas de todas as classes. Usando editores visuais externos, o administrador pode projetar fluxos de trabalho, importá-los para o CMDBuild e colocá-los à disposição dos operadores, para que possam executá-los de acordo com o

automatismo configurado. De maneira semelhante, usando editores visuais externos, o administrador pode criar vários relatórios sobre impressões de dados CMDB (gráficos, etiquetas de código de barras etc.), importá-los para o sistema e colocá-los à disposição dos operadores. O administrador também pode configurar alguns painéis compostos por gráficos que mostram imediatamente a situação de alguns indicadores no sistema atual. Um gerenciador de tarefas incluído na interface do usuário do Módulo de Administração permite agendar várias operações (início do processo, recebimento e envio de e-mail, execuções de conectores) e vários controles nos dados do CMDB (eventos síncronos e assíncronos). Com base nas descobertas, ele envia notificações, inicia fluxos de trabalho e executa scripts. Graças aos sistemas de gerenciamento de documentos que suportam o padrão CMIS (Content Management interoperability Services), entre os quais também há a solução de código aberto *Alfresco*, que é sistema de Gestão de conteúdo empresarial (em inglês ECM "Enterprise Content Management"). Há também um agendamento, que pode ser fornecido automaticamente ao preencher um cartão de dados e manualmente. Esse agendamento gerenciará prazos únicos ou recorrentes relacionados, por exemplo, a certificações, garantias, contratos com clientes e fornecedores, procedimentos administrativos etc. O CMDBuild permite utilizar os recursos GIS para georreferenciar e exibir ativos em um mapa geográfico (serviços de mapas externos), ou em mapas vetoriais (GeoServer local e banco de dados espacial PostGIS) e recursos de Building information Modeling (BIM) para visualizar modelos 3D (formato IFC) [CMDBuild, 2020].

O sistema também inclui um serviço da web REST, para que os usuários do CMDBuild possam implementar soluções de interoperabilidade personalizadas com sistemas externos [CMDBuild, 2020].

3.3.1. Vantagens.

Possui Licença AGPL, customização das características, mobile app.

É fácil criar formulários com tabelas de banco de dados relacionadas. Além disso, a relação entre as tabelas pode ser definida de forma fácil. O software é flexível e personalizável. Possui código aberto, é construído em JAVA, possui conectores para sistemas externos em um único produto e o mais importante, a solução pode rastrear o ciclo de vida completo do ativo.

3.3.2. Desvantagens.

Algumas características (aplicativo móvel, estrutura GUI completa, etc.) estão disponíveis apenas na compra da assinatura anual.

A Documentação é sempre um problema. Embora alguma documentação esteja disponível, faltam instruções concretas sobre como fazer com que algumas utilidades mais complexas funcionem. Há um escopo de melhoria na interface do usuário.

Sua curva de aprendizado é acentuada. No CMDBuild configurar uma solução customizada sem o suporte de uma documentação concisa não é uma boa prática.

O CMDB foi desenvolvido para fins administrativos e deixa a desejar na questão do gerenciamento de infraestrutura.

Fonte.:https://www.capterra.com.br/software/149326/cmdbuild#deployment

3.4. OneCMDB

OneCMDB é um CMDB voltado para pequenas e médias empresas. Ele pode ser usado como um CMDB independente para controlar os ativos de software e hardware e suas relações, graças à sua API Rest. O OneCMDB é fácil de instalar e preencher com dados, seja manualmente ou com outras fontes de dados . Ele possui um modelo de dados especificado pelo usuário que pode ser alterado e aprimorado sem programação. A proposta do CMDB é poder criar seu modelo de dados CMDB, sem escrever uma única linha de código preencher o banco de dados, por meio da descoberta automática de sua rede. Realiza importação e exportação de fontes externas. Importa/exporta informações de configuração de rede de para o NAGIOS Network Monitoring System OneCMDB. Além de ser escrito em Java, sua interface de serviços da Web permite que aplicativos escritos em outras linguagens, além de Java, se comuniquem facilmente com o OneCMDB. Ele também permite que os aplicativos acessem o OneCMDB remotamente pela rede.

3.4.1. Vantagens.

A nova versão apresenta uma função de descoberta automática aprimorada. A função de descoberta navega na rede do usuário e preenche o modelo CMDB com dados básicos sobre o equipamento de rede que encontra. No equipamento encontrado, ele também verifica as portas abertas para descobrir quais aplicativos estão sendo executados, semelhante à solução IMC da HP.

3.4.2. Desvantagem.

O projeto foi descontinuado no ano de 2014, segundo informações do site do projeto [onecmdb].

3.5. NETBOX

O NetBox é um sistema que une a capacidade de DCIM (Data Center Infraestructure Management) e IPAM (IP Address Management), ou seja, uma ferramenta web de código aberto desenvolvido por Jeremy Stretch, funcionário da Digital Ocean para ajudar a gerenciar e documentar redes de computadores. Com ele é possível cadastrar Sites, PoPs, Racks, Dispositivos, interfaces dos dispositivos, associando o uso dos endereços IP em cada um deles (vinculando às interfaces), além das conexões entre eles. Também é possível cadastrar VLANs, Circuitos, Senhas entre outras informações úteis para documentar uma rede e fazer inventário. Ideal para provedores que estão crescendo e precisam ter controle sobre a sua rede [Hoisel, 2016].

O NetBox foi projetado com alguns princípios, dentre os quais destacam-se: Replicar o mundo real, cuja consideração cuidadosa foi dada ao modelo de dados para garantir que ele possa refletir com precisão uma rede do mundo real. Por exemplo, os endereços IP são atribuídos não a dispositivos, mas a interfaces específicas conectadas a um dispositivo, e uma interface pode ter vários endereços IP atribuídos a ele; Servir como uma "fonte de verdade", pois o NetBox pretende representar o estado desejado de uma rede versus seu estado operacional. Como tal, a importação automatizada do estado da rede, ao vivo, é fortemente desencorajada. Todos os dados criados no NetBox devem primeiro ser examinados por um Analista de TI ou Administrador de Redes para garantir sua integridade. O NetBox pode então ser usado para preencher sistemas de monitoramento e provisionamento com um alto grau de confiança; manter a simplicidade, Quando é possível escolher entre uma solução relativamente simples e uma solução completa muito mais complexa, a primeira será tipicamente favorecida. Isso irá garantir uma base de código enxuta com curva de aprendizagem suave [Netbox, 2020]. O Netbox é desenvolvido na estrutura do Django Python e utiliza um banco de dados PostgreSQL, ou seja, uma linguagem e um banco de dados relacional que está em ascensão, com documentação acessível e uma comunidade assídua nos problemas do cotidiano.

O NetBox foi desenvolvido especificamente para atender às necessidades dos engenheiros de rede e infraestrutura [Netbox, 2020]. Além dos requisitos supracitados o Netbox abrange os seguintes aspectos do gerenciamento de rede:

- Gerenciamento de endereço IP (IPAM) - redes e endereços IP, VRFs e VLANs.
- Suportes para equipamentos - Organizados por grupo e site.
- Dispositivos - tipos de dispositivos e onde eles estão instalados.
- Conexões - conexões de rede, console e energia entre dispositivos.
- Virtualização - Máquinas virtuais e clusters.
- Circuitos de dados - circuitos e provedores de comunicações de longo curso.
- Segredos - Armazenamento criptografado de credenciais confidenciais.

3.5.1. Vantagens.

O netbox é um CMDB voltado para um ambiente de infraestrutura de TI; Totalmente customizado, suportando Webhooks e Custom Links; Possui comunicação direta com o estado dos ativos de rede que suportam Napalm [Nalpalm, 2017]; Consegue ser integrado a ferramentas de monitoramento de rede, tal como o Zabbix [Horst, 2015]; por ser desenvolvido em Python e ser desenvolvido de forma modular, suporta a adição de plugins.

3.5.2. Desvantagens.

Embora o NetBox se esforce para cobrir muitas áreas do gerenciamento de rede, o escopo de seu conjunto de recursos é necessariamente limitado. Isso garante que o desenvolvimento se concentre na funcionalidade principal e que a fluência do escopo esteja razoavelmente contida. Para esse fim, pode ser útil fornecer alguns exemplos de funcionalidade que o NetBox não fornece:

- Monitoramento de rede.
- Servidor DNS.
- Servidor RADIUS.
- Gerenciamento de instalações.

3.6. A escolha do CMDB a ser customizado

Após a análise dos requisitos iniciais, os CMDB's foram sendo descartados do estudo por diversos motivos, tais como:

- iTop: a Versão free é muito limitada e os módulos adicionais são pagos.
- CMDBuild: curva de aprendizado bastante acentuada e não possui documentação clara a respeito das funcionalidades, bem com é um CMDB com construção voltada à administração e não a Infraestrutura de rede, propriamente dita.
- OneCMDB: O projeto foi descontinuado em 2014.

O Netbox foi o CMDB que mais atendeu aos requisitos iniciais exigidos e por esse motivo será o objeto de estudo neste trabalho.

4. CUSTOMIZAÇÃO DO CMDB

Este capítulo explica como a customização do CMDB foi feita, detalhando os requisitos exigidos, a definição do diagrama de classes, regras de negócios e apresentação da ferramenta.

4.1. Análise de Negócio e análise de Requisitos

Em reunião com a diretoria de redes da SINFO foram discutidas duas possibilidades de estudo e desenvolvimento: a primeira seria desenvolver um código para o gerenciamento do DNS (que a documentação do Netbox deixa claro que não está desenvolvido); e a segunda seria desenvolver um código para o gerenciamento do serviço DHCP. Foi proposto customizar o serviço DHCP, por meio de um plugin. O levantamento de requisitos e desenvolvimento do plugin do DNS ficou sob responsabilidade de um dos funcionários da SINFO, bem como os requisitos e responsabilidade do desenvolvimento do plugin relacionado ao DHCP ficaram a cargo de serem mostrados neste trabalho de conclusão.

Para o desenvolvimento do Plugin DHCP uma lista de requisitos foi levantada:

- Configurações de Prefix.
- Configurações de Vlan.
- Configurações de IP de Gerência.
- Identificação do IP do servidor de DNS.
- Identificação e associação IP do servidor responsável pelo serviço DHCP.
- Identificação do Gateway.
- Definição do Pool do endereçamento IP.
- Identificação e associação do Site.
- Tipo de endereçamento a ser configurado (IPV4 ou IPV6).
- Identificação e configuração da Opção (43 ou 60, específico).
- Identificação e associação do número de chamado.
- Identificação e associação do Mac address.
- Configuração de host.
- Associação de IP Fixo no Prefix para determinado host.
- Campo para comentários diversos.

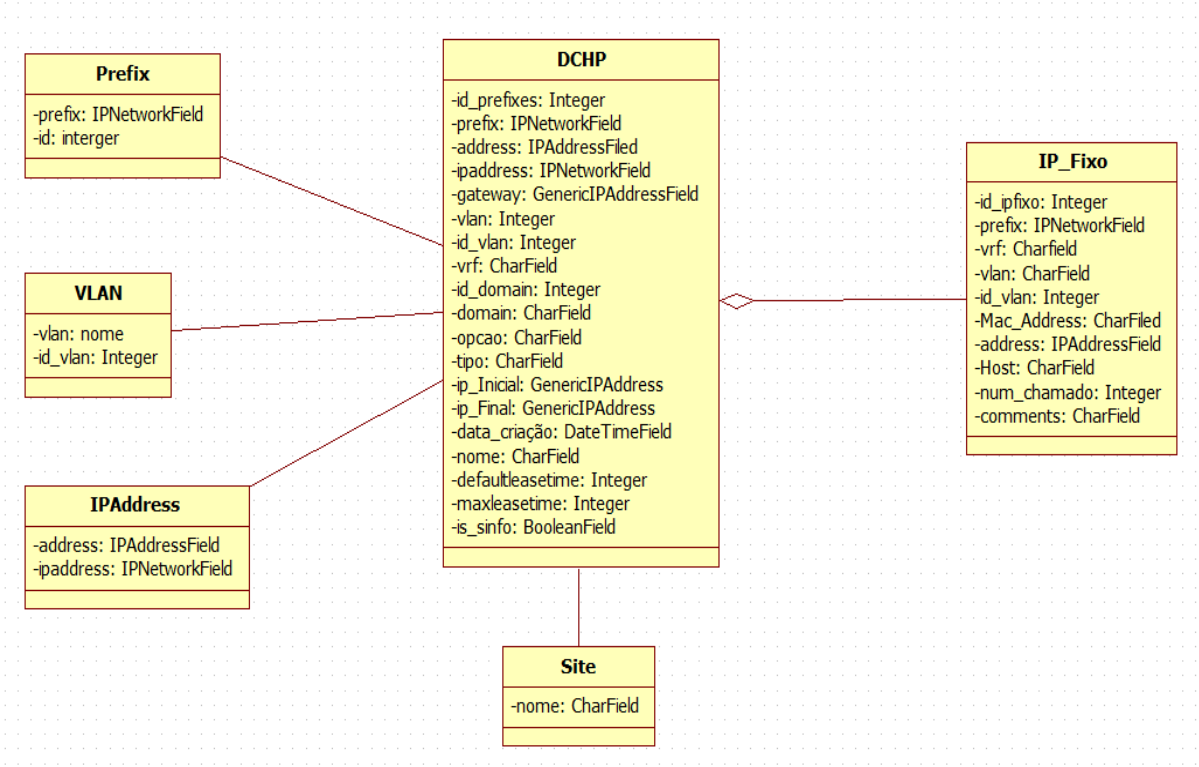
4.2. Tratamento dos dados e normalização

De posse das informações do item supracitado foi preciso fazer uma análise para identificar quais informações seriam classificadas de forma primária e secundária. As informações a serem tratadas foram:
Prefix, Vlan, Address, IPaddress, Service, Site, Protocol, Port number, Gateway, VRF, Domain, Option, Tipo, Pool, Date Created, Default Lease Time, Max Lease Time, Responsible, Mac Address, Host, Number of Call, comments.

A nível de Banco de dados foi percebido que seriam utilizadas formas de normalização de dados. França e Junior [2015] dizem que uma forma normal é uma regra que deve ser seguida para que uma tabela seja bem avaliada. A forma normal sujeita o esquema de relação a uma cadeia de avaliação para garantir que ele satisfaz a forma normal. Esse processo de avaliação segue o estilo top-down, na qual cada relação é avaliada sob os critérios das formas normais. Uma tabela está na 1FN se não possuir atributo multivalorado ou atributo composto, esse procedimento elimina tabelas aninhadas. Uma tabela está na 2FN se estiver na 1FN e não possuir dependência funcional parcial. Uma dependência parcial ocorre quando os atributos “não chave” não dependem de toda chave primária composta, ou seja, se um atributo é multivalorado é gerada uma nova tabela e sua chave estrangeira fica na tabela mais fraca.

Após realizar a normalização foi percebido era preciso relacionar algumas tabelas do banco de dados do Netbox com as novas tabelas a serem criadas para o desenvolvimento do plugin. Com a normalização os campos puderam ser organizados e relacionados com as tabelas já existentes. Foram criadas duas tabelas com características específicas ao serviço DHCP. A primeira delas é a tabela “DHCP”, que conta com informações de configurações que um servidor precisa; a segunda tabela gerada é a tabela “IP_Fixo", que fornece as configurações necessárias para que um host possa adquirir um endereço IP após realizar uma requisição com o servidor. As tabelas do banco de dados foram definidas de acordo com o diagrama UML, mostrado na Figura 6.

Figura 6 – Normalização e diagrama de classes



Fonte: o Próprio autor

4.3 Dependência de dados

Na forma na qual as classes foram definidas há uma dependência lógica na criação de um serviço DHCP, como exemplo, é preciso saber que para associar um serviço DHCP a um Prefix, o mesmo deve estar criado, juntamente com uma Vlan. A partir daí, criar um IP address, ou um pool de endereços, que estejam disponíveis e acessíveis às classes desenvolvidas. Após a criação dessas dependências é possível associar um serviço DHCP a um Prefix. Da mesma forma, para ser criado um endereço específico a um Host na rede é preciso que existam as mesmas informações.

4.4. Regras de negócio

As regras de negócio são declarações sobre a forma na qual o Plugin é gerido. Com elas é possível limitar o escopo do projeto, auxiliar na tomada de decisão, estabelecer quais operações são permitidas e quais não, bem como resguardar o programador frente ao cliente final. No desenvolvimento da ferramenta algumas regras de negócio foram definidas, tais como:

- Importação de dados – será por meio do protocolo Rest e arquivo .csv
- Exportação de dados – será por meio de um arquivo .sql gerado pelo próprio Netbox.
- Não é possível ter duas Vlans correspondentes a um mesmo prefixo.
- O Prefix não está amarrado ao Mac address do Host, ou seja, caso o Prefix seja deletado do banco de dados o host associado a ele não será deletado, concomitantemente.
- Não é possível ter dois hosts com Mac Address iguais no mesmo Prefix, ou seja, isso pode gerar um conflito de IP's

4.5. Ambiente de desenvolvimento

O desenvolvimento do plugin foi feito em um ambiente Docker, ou seja, em um ambiente "conteinerizado". O ambiente Docker foi escolhido por ser leve, versátil e não consumir tantos recursos de um host, comparado com as instalações convencionais [Vitalino e Castro, 2016]. No ambiente de desenvolvimento foram instaladas algumas ferramentas, tais como:

- VScode – um editor de código fonte.
- Git - um versionador de projetos.
- o Python e o Django – linguagem de programação e framework, respectivamente.

O ambiente de desenvolvimento do NetBox foi iniciado e foram feitas configurações iniciais, seguindo os passos a seguir:

Passo 01.:
Criar um diretório no linux chamado "Netbox" e fazer o clone do projeto do site github (https://github.com/netbox-community/netbox-docker).

Passo 02.:
Definir as configurações de ambiente no arquivo docker-compose.yml, tais como porta de acesso, configurações de banco de dados e afins. A Figura 7 mostra um exemplo do arquivo citado:

Figura 7 – exemplo do arquivo “docker-compose.yml”

```yaml
version: '3.4'
services:
  netbox: &netbox
    image: netboxcommunity/netbox:${VERSION-latest}
    depends_on:
    - postgres
    - redis
    - redis-cache
    - netbox-worker
    env_file: env/netbox.env
    user: '0'
    volumes:
    - ./startup_scripts:/opt/netbox/startup_scripts:z,ro
    - ./initializers:/opt/netbox/initializers:z,ro
    - ./configuration:/etc/netbox/config:z,ro
    - ./reports:/etc/netbox/reports:z,ro
    - ./scripts:/etc/netbox/scripts:z,ro
    - netbox-nginx-config:/etc/netbox-nginx:z
    - netbox-static-files:/opt/netbox/netbox/static:z
    - netbox-media-files:/opt/netbox/netbox/media:z
  netbox-worker:
    <<: *netbox
    depends_on:
    - redis
    entrypoint:
    - python3
    - /opt/netbox/netbox/manage.py
    command:
    - rqworker

  # nginx
  nginx:
    command: nginx -c /etc/netbox-nginx/nginx.conf
    image: nginx:1.19-alpine
    depends_on:
    - netbox
    ports:
      - 8000:8080
    volumes:
    - netbox-static-files:/opt/netbox/netbox/static:ro
    - netbox-nginx-config:/etc/netbox-nginx/:ro

  # postgres
  postgres:
    image: postgres:12-alpine
    env_file: env/postgres.env
    volumes:
    - netbox-postgres-data:/var/lib/postgresql/data

  # redis
  redis:
    image: redis:6-alpine
    command:
    - sh
    - -c # this is to evaluate the $REDIS_PASSWORD from the env
    - redis-server --appendonly yes --requirepass $$REDIS_PASSWORD ## $$ because of docker-compose
    env_file: env/redis.env
    volumes:
```

Fonte: o próprio autor.

Passo 03.:

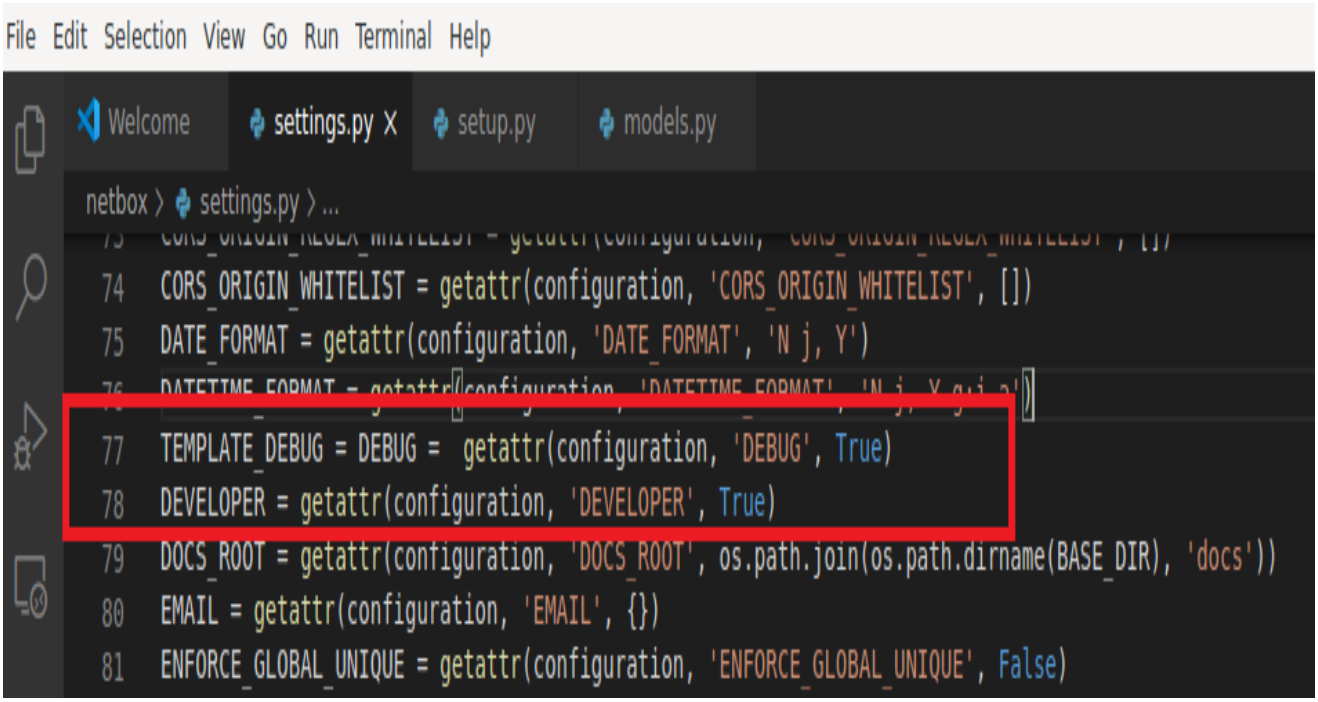
Iniciar o ambiente Docker, realizando um pull das imagens necessárias.

Após o ambiente Docker ser iniciado, deve-se realizar os passos a seguir dentro do container do Netbox:

Passo 04.:

Acessar ao arquivo settings.py e modificar as linhas 77 e 78, conforme a Figura 8:

Figura 8 – modificação do arquivo para habilitar a função de desenvolvedor



Fonte: o próprio autor

Passo 05.:

Criar um diretório de projeto e, paralelo a esse diretório, criar um arquivo “setup.py”, no qual deverá conter os dados da versão, autor, descrição e os dados de instalação do plugin, bem como outras informações. A Figura 9 mostra a configuração do arquivo setup.py.

Figura 9 – exemplo de arquivo “setup.py”

plugins > Dhcp > setup.py > ...

```python
import os
from setuptools import setup, find_packages


here = os.path.abspath(os.path.dirname(__file__))
readme = ""
if os.path.exists('README.md'):
    with open('README.md', 'r') as f:
        readme = f.read()

setup(
    name='dhcp',
    version='1.0.1',
    description='A Netbox plugin for dhcp service management',
    long_description=readme,
    long_description_content_type='text/markdown',
    url='https://github.com/anderson-acrs/dhcp',
    author='anderson claudio',
    license='GNU General Public License v3.0',
    package_data={
        '': ['LICENSE'],
    },
    install_requires=[],
    packages=find_packages(),
    include_package_data=True
)

```

Fonte: o próprio autor

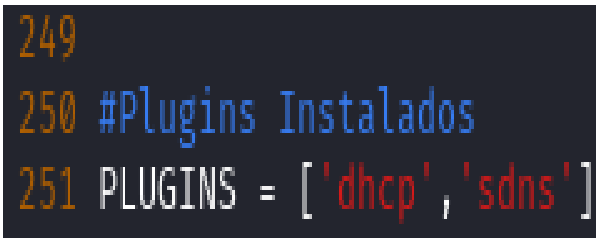
Os exemplos de definição dos modelos do banco de dados, exemplos de Views, Forms e demais arquivos de configuração e funcionamento do plugin foram seguidos conforme padronização e definição da documentação do Netbox, disponível em: https://netbox.readthedocs.io/en/stable/plugins/development/

Passo 06:

No linux, é preciso acessar ao diretório no qual se encontra o arquivo de configurações adicionais do projeto NetBox. Para habilitar o plugin é preciso adicionar as linhas ao arquivo configuration.py, conforme representado na Figura 10.

```
root@dev:/Netbox/netbox-docker# vim configuration/configuration.py
```

Figura 10 – adição de linhas no arquivo “configuration.py”

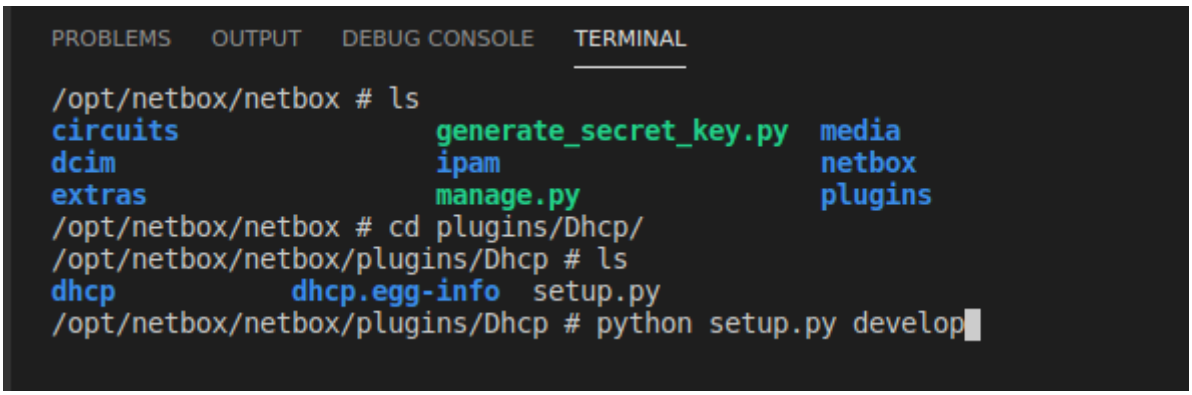


Fonte: o próprio autor

Passo 07:

Para realizar a instalação no container e desenvolver o plugin é preciso digitar os comandos no local abaixo, conforme mostrado na Figura 11:

Figura 11 – Instalação do plugin na versão desenvolvedor



Fonte: o próprio autor

```
/opt/netbox/netbox/plugins/Dhcp # python setup.py develop
```

Passo 08:

Desenvolver o seu plugin de acordo com as especificações propostas pela documentação do Netbox (mostrado no passo 5).

4.6. Instalação do Plugin

O plugin após desenvolvido passou por um processo de empacotamento e foi disponibilizado no site https://test.pypi.org/ . Para proceder com a instalação em outro ambiente Docker/Netbox basta habilitar o plugin, conforme mostrado no passo 06 e proceder com a instalação, via comando pip no terminal do conteiner do NetBox.

```
pip install -i https://test.pypi.org/simple/ dhcp
```

4.7. Segurança do modelo

Como o Netbox é escrito em Django esse framework possui ferramentas prontas para que a questão da segurança possa ser pensada. Analisando outros programas de computador foi verificado que alguns deles continham falhas que permitiam o acesso às funcionalidades do código, mesmo sem a necessidade de realizar um login, ou seja, sem os protocolos de segurança de Authentication, Authorization e Accounting (AAA). Desse modo o desenvolvimento fica inseguro e compromete a integridade do banco de dados por não registrar nenhum tipo de acesso em log. Utilizando as classes corretas de programação no Netbox, ao criar um usuário, o Django fornece uma chave Token de 128bits para que esse usuário possa inserir, acessar e/ou consultar as informações presentes no CMDB. Nesse sentido, na construção da View de acesso foi implementada a classe "PermissionRequiredMixin". Essa classe verifica se o usuário já realizou o login com suas devidas credenciais e, baseada nelas, libera (ou não) as Views do código, além de realizar o registro em Log. A Figura 12 exibe o uso dessa classe.

Figura 12– exemplo de uso da classe "PermissionRequiredMixin"

plugins > Dhcp > dhcp > views.py > render

```
You, 7 days ago | 1 author (You)
from django.shortcuts import render, get_object_or_404, get_list_or_404      You, 7 days ago •
from django.contrib.auth.mixins import PermissionRequiredMixin
from django.contrib.auth.decorators import permission_required, login_required
from .models import Dhcp, Ipfixo
from .forms import DhcpForm, DhcpFilterForm, DhcpCSVForm, IpfixoForm, IpfixoFilterForm
from .filter import DhcpFilter, IpfixoFilter
from .tables import DhcpTable, IpfixoTable
from django.views import View
from django.views.generic.base import TemplateView
from .utils import get_token, get_user, get_unit, get_server
from  utilities.views import ObjectListView, ObjectEditView, ObjectDeleteView, BulkDeleteView,

# Create your views here.

#Bloco DHCP
"""Agrupa as Views da classe DHCP"""
You, 7 days ago | 1 author (You)
class DhcpView(PermissionRequiredMixin, View):
    permission_required = 'dhcp.dhcp_view'
    def get(self, request, pk):
        dhcp = get_object_or_404(Dhcp.objects.filter(id_prefixes=pk))
        return render(request, 'dhcp/dhcp.html', {
            'dhcp': dhcp
        })

```

Fonte: o próprio autor.

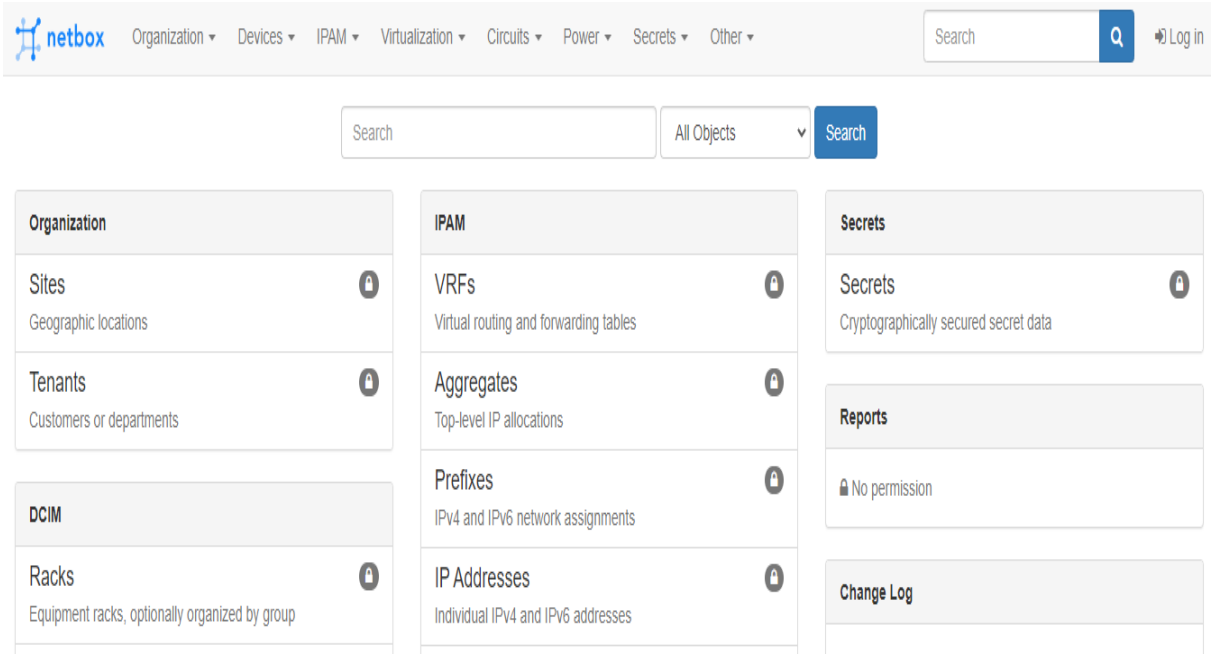
5. RESULTADOS

Este capítulo apresenta os resultados do modelo de configuração do serviço DHCP, desenvolvido através de um plugin. O desenvolvimento atendeu aos requisitos e as regras de negócio definidas, bem como o comportamento do modelo que mostrou-se estável na alimentação das informações iniciais.

5.1. Sem o plugin instalado

Quando um ambiente NetBox é acessado pela primeira vez é possível visualizar que a barra de Menus do Netbox possui 08 opções de menus de acesso. A barra de Menus é composta pelos seguintes campos: Organization, Devices, IPAM, Visualization, Circuits, Power, Secrets, Other. A Figura 13 apresenta o ambiente inicial do Netbox.

Figura 13 – visualização da página inicial do Netbox



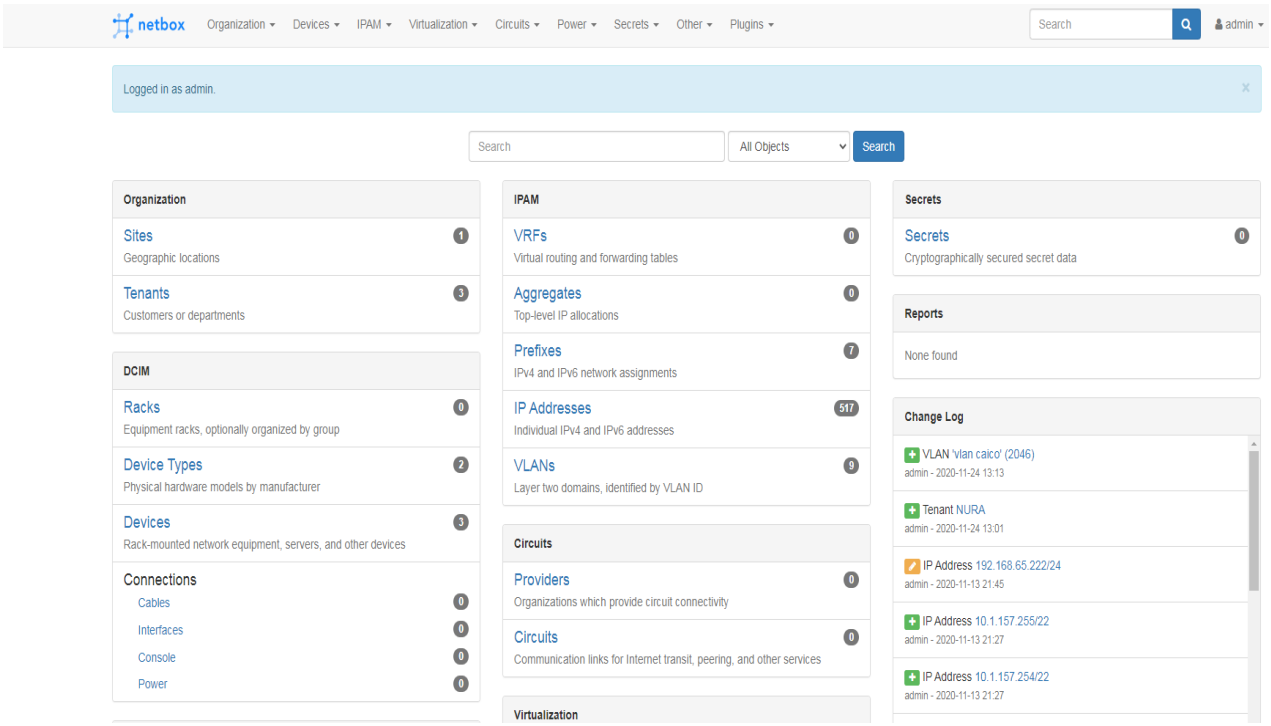
Fonte: o próprio autor

5.2. Com o plugin instalado

Após habilitar o plugin no arquivo “configuration.py” e realizar a instalação do plugin “dhcp”, via comando “pip” no container, é possível visualizar a adição de mais um menu na barra. O menu “Plugins”. Nele estão todas as opções de plugins que seguiram as orientações da documentação do Jeremy Strech, com o auxílio da documentação do Netbox. A Figura 14 e Figura 15 referem-se a habilitação do menu Plugins e as opções de escolhas de views após clicar no menu. Também é possível

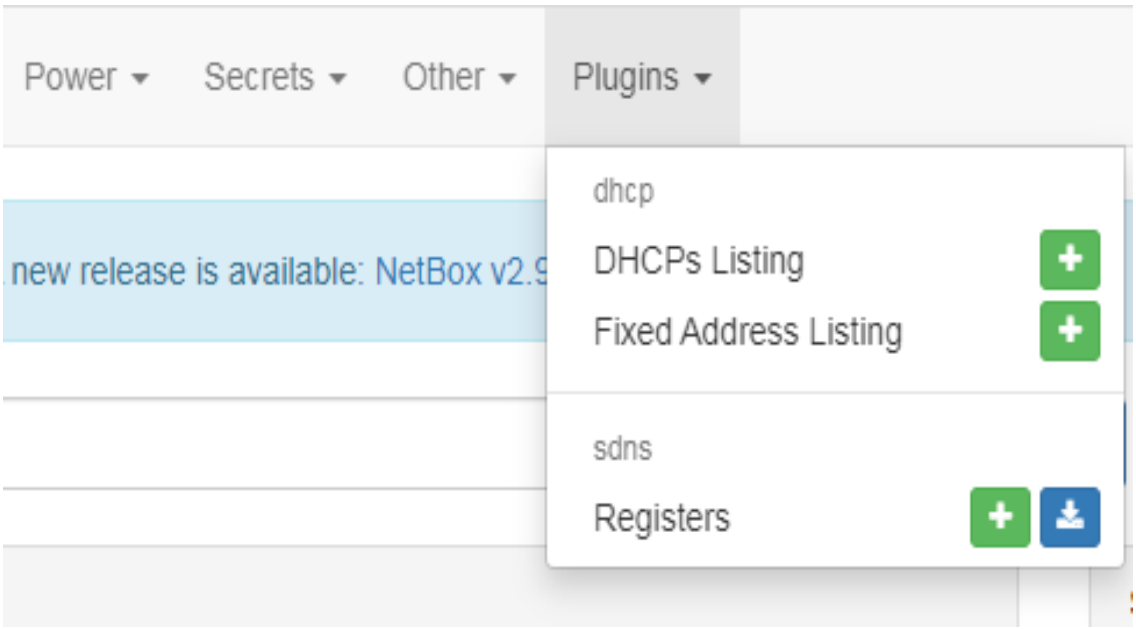
observar a coexistência de dois plugins desenvolvidos: O plugin DHCP e o plugin SDNS, respectivamente.

Figura 14 – visualização do menu “Plugins”



Fonte: o próprio autor

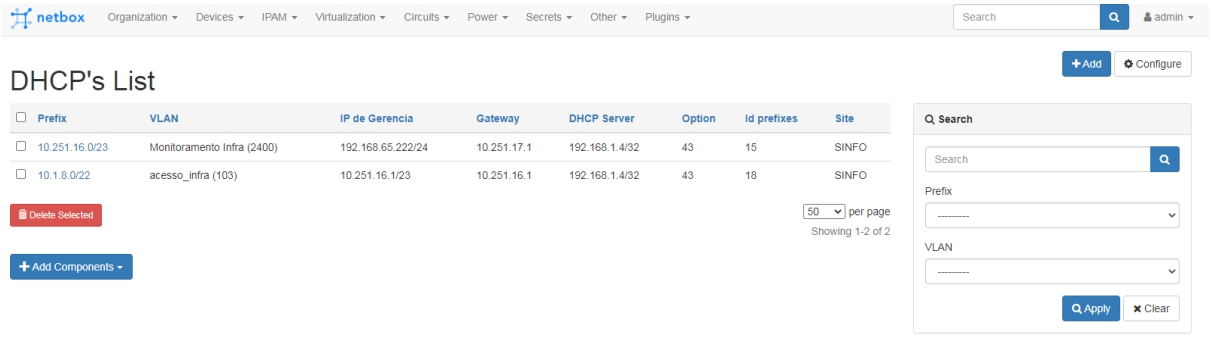
Figura 15 – visualização das opções do menu “Plugins”



Fonte: o próprio autor

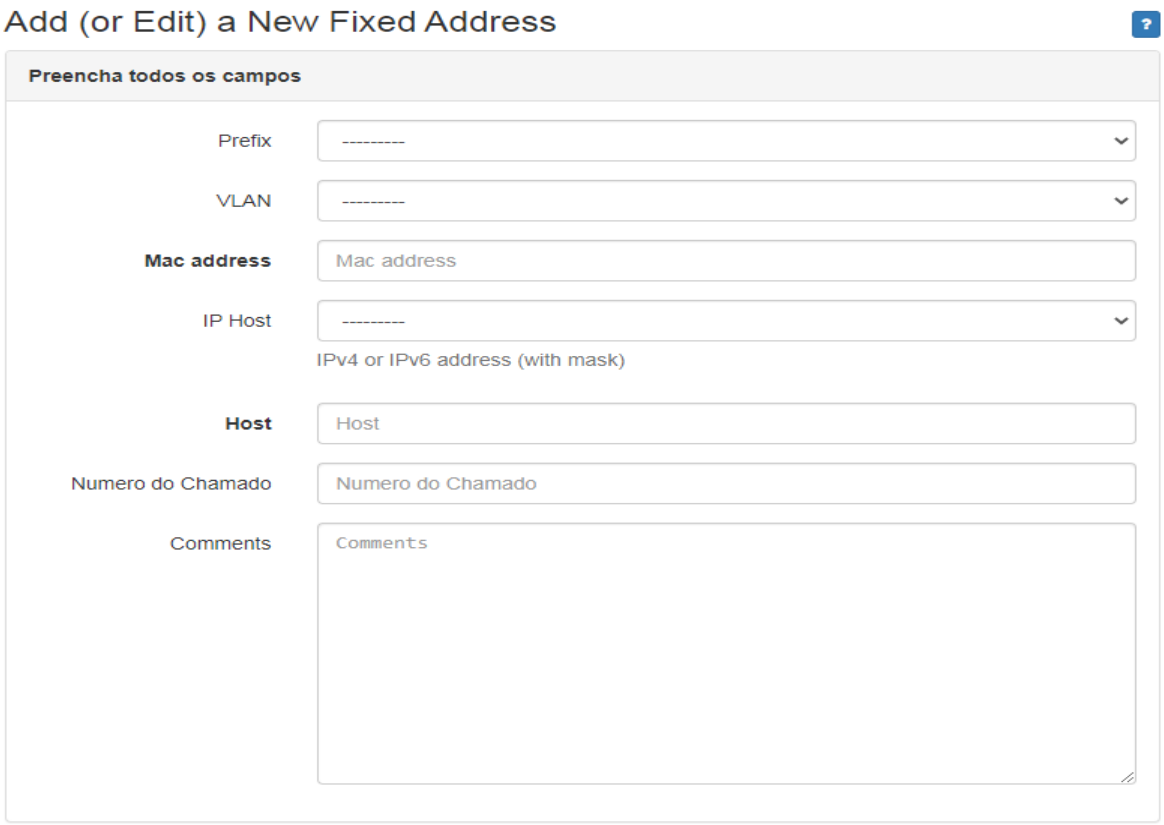
Com o plugin instalado é possível visualizar, na view “DHCP’s List”, a lista de DHCP’s referentes a cada Prefix, conforme definido na lista de requisitos. É possível observar que o Plugin herdou o padrão das Views do Netbox, bem como os botões, estilos de página, operações CRUD, links internos e afins, como mostrado na Figura 16. A Figura 17 apresenta a View de adição de um novo “Fixed Address” para um novo host na rede, conforme número de chamado recebido pela SINFO.

Figura 16 – visualização da lista de DHCP’s



Fonte: o próprio autor

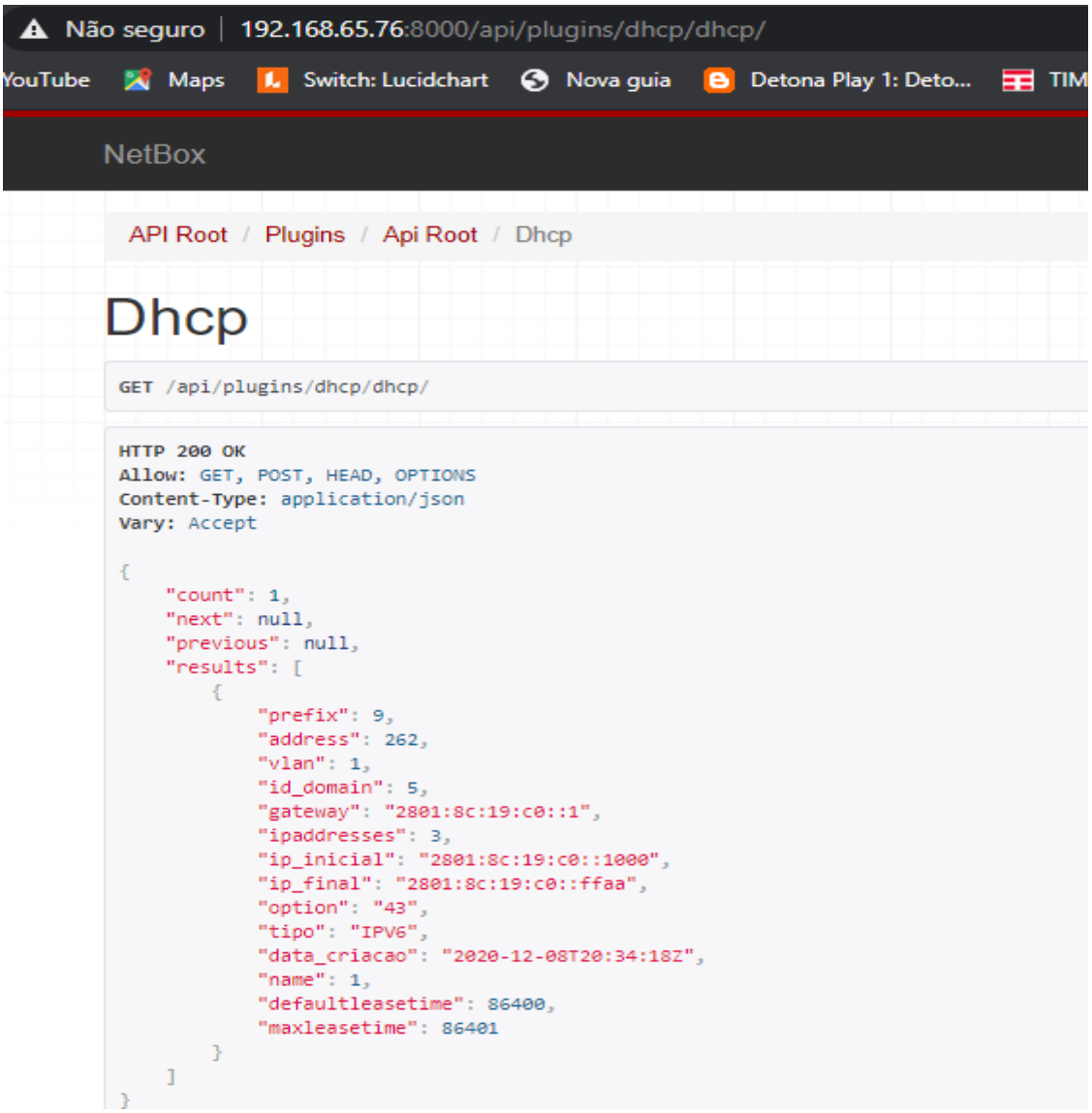
Figura 17 – visualização para adição de um novo host



Fonte: o próprio autor

Um outro requisito atendido foi desenvolver a interface REST para que os ativos de rede que tivessem suporte, pudessem ser consultados pelo administrador de redes ou pelo CMDB Netbox. O cumprimento desse requisito permitirá duas possibilidades importantes: inserir informações do gerenciamento das configurações referentes ao DHCP sem o auxílio da interface web, ou seja, por meio de comandos em linha de códigos; servirá como base para possíveis implementações de automatização futura. Para cumprir esse requisito foi preciso criar uma classe na qual seus elementos fossem serializados e dispostos de forma lógica para que fosse possível a inserção no banco de dados, tal como mostra a Figura 18.

Figura 18 – visualização da classe de serialização



Fonte: o próprio autor

Como exemplo desta possibilidade, a inserção das informações para o gerenciamento de configurações, pode-se destacar o exemplo do POP-RN, que possui em uma de suas várias atribuições a gerência do serviço DHCP do projeto GigaMetropole. O projeto Giga Metropole é uma rede de comunicação de dados de alta velocidade, que utiliza tecnologia óptica para prestar serviços de conectividade física a instituições localizadas na Região Metropolitana de Natal. A rede Giga Natal atende cerca de 360 escolas publicas (estaduais e municipais) e é operada e mantida pelo PoP-RN, através de seu Centro de Operação de Redes (NOC). A Rede Giga Metrópole foi inaugurada oficialmente em 19 de junho de 2017 e é uma ampliação da Rede Giga Natal, que se encontra em operação desde 2008, atendendo a diversas instituições na capital. Ela interliga os campi do IFRN e da UFRN localizados nos municípios da região da Grande Natal. A nova rede elevou a extensão de 40 km de fibra, inicialmente instalada para aproximadamente 260 km nos trechos de backbone, atingindo nove dos onze municípios dessa região. Com os recursos disponibilizados pela UFRN, foram implantados 120 km para construir o denominado Anel Norte, que atende aos municípios de Parnamirim, Macaíba, São Gonçalo do Amarante, Extremoz e Ceará Mirim, e 80 km para construir o Anel Sul, que atende aos municípios de São José do Mipibu, Vera Cruz e Monte Alegre [IMD, 2020].

O NOC do POP-RN realiza o gerenciamento do serviço DHCP das escolas atendidas pela RGM através de uma solução de firewall chamada pfSense. O pfSense é um dos mais conceituados firewalls do mercado e, dentre suas funcionalidades, possui API REST.

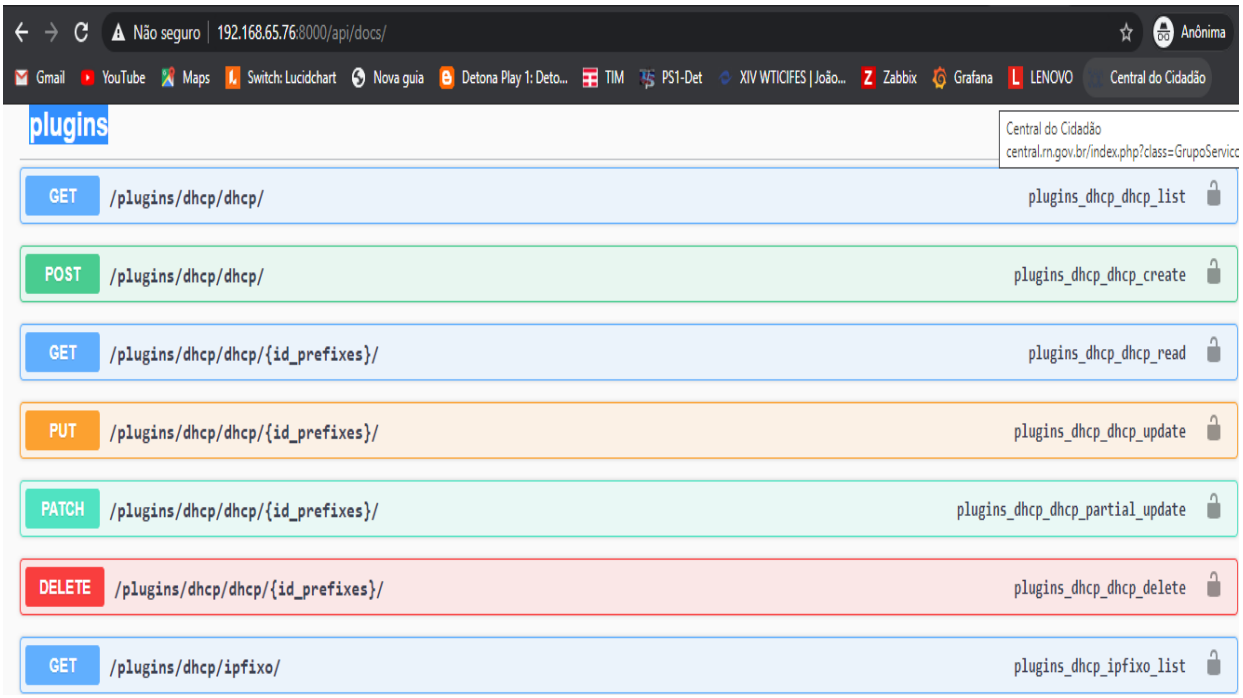
No primeiro momento foi feita a comunicação entre o pfSense e o Netbox para que, por meio da APT REST, fosse possível o firewall alimentar o Netbox com as informações contidas nele, para que o Netbox fosse um espelho do armazenamento de configuração, um backup. Lógicamente a construção destas ferramentas, o Netbox e o pfSense, possuem formas de organização e inserção interna dos dados bem distintas e , por esse motivo faz-se necessário verificar a documentação da API REST de ambas ferramentas para saber a ordem na qual as informações serão inseridas.

Em um segundo momento será possível pensar em uma solução na qual, a partir das informações inseridas no CMDB, o netbox possa acessar e consultar o pfSense e configurá-lo da mesma forma na qual o Netbox está configurado atualmente.

Para isso é necessário que a documentação da API REST esteja acessível e todas as opções de CRUD estejam funcionais e registradas de forma fácil. Nesse sentido o Django, a partir da classe de serialização cria um diretório chamado de “docs”, que auxilia o desenvolvedor ou administrador de redes a enteder como a inserção de dados poderá ser feita.

A Figura 19 apresenta a parte do CMDB referente ao “Docs”, um diretório que contém todas as URL’s, bem como todos os acessos por linha de comando, estrutura nos quais os dados são inseridos e apresentados ao usuário, bem como modelos e exemplos de como utilizar os métodos de Get, Post, Put, Delete e Path.

Figura 19 – visualização da API Docs



Fonte: o próprio autor

6. CONCLUSÃO E TRABALHOS FUTUROS

Para uma instituição do porte da UFRN, que possui um grande campus central e vários campi adjacentes, um CMDB é uma ferramenta indispensável de armazenamento e gerência de informações do parque de TIC e na tomada de decisões dos problemas do cotidiano. Foi visto que o serviço DHCP gera bastante informação para ser armazenada em um banco de dados (tais como: informação de Vlan, Prefix, nome de Hosts e afins), e que o Netbox, no papel de banco de dados de gerenciamento de configuração, consegue gerir essas informações sem grandes problemas. O desenvolvimento do Plugin cumpriu com os requisitos exigidos e é um modelo de gerenciamento do serviço DHCP que pode ser adotado, melhorado e utilizado na SINFO e no POP-RN. O estudo de tecnologias como a API Rest serve como base para a comunicação dos ativos de rede, máquinas virtuais e outros dispositivos afins, que se comuniquem com o CMDB, bem como outros sistemas de monitoramento. O desenvolvimento do modelo de gerenciamento com o framework Django facilitou a atividade, pois é uma documentação bastante atualizada e ativa pela comunidade de desenvolvedores. O framework possui funções de autocompletar a escrita e dá sugestões sobre a construção do código, além de possuir integração com o repositório Git.

Isto posto, um próximo passo para futuros trabalhos é o aprimoramento do Plugin DHCP, podendo ser desenvolvidos webhooks e webservices que possam, aliados ao CMDB, fornecer alguma automação na configuração dos ativos de rede. Para o POP-RN o uso do Netbox tem sido, além do início do armazenamento das configurações do DHCP por meio da comunicação entre o Netbox e o PFsense, iniciar o estudo de um projeto piloto no gerenciamento das máquinas virtuais. Um outro projeto, em conjunto com a SINFO, é desenvolver um validador de configurações de ativos de rede e máquinas vituais, no qual o principio é verificar esporadicamente se as configurações atuais estão condizentes com o que está armazenado no CMDB. Caso essas informações não estivejam em conformidade o Netbox deve iniciar um procedimento de atualização ou um retrocesso dessas configurações (roll-back). O primeiro passo é ter uma base de informações/configuração sólida e atualizada constantemente.

Em suma, os requisitos deste trabalho foram cumpridos, gerando uma ferramenta de gestão do serviço DHCP Open Source e que será utilizada na UFRN e POP-RN e está disponível para a toda a comunidade que desejar ter um código base para auxiliar nos seus problemas de TIC. Também foi possível explorar outras áreas de conhecimento e aprender sobre novas ferramentas, que não foram tão abordadas durante o curso.

Referências

API Restful: Conceitos, princípios e como criar, disponível em: https://www.hostgator.com.br/blog/api-restful/ , acesso em 25/11/2020.

Artigo RedHat - “o que é uma API Rest”, disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api , Acesso em 01/12/2020.

ASSIS, M. Baeta - Afinal, o que é um CMDB e como pode ser usado, disponível em: https://cio.com.br/tendencias/afinal-o-que-e-cmdb-e-como-pode-ser-usado/, acesso em 01/12/2020.

BRANDIZZI, L – SQl Vs NoSql – “Qual a melhor Opção para armazenamento de Grafos?”, Artigo da SERPRO, disponível em: https://www.serpro.gov.br/menu/noticias/sql-vs-nosql-2013-qual-a-melhor-opcao-para-armazenamento-de-grafos , acesso em 20/11/2020.

Cambridge’s Dictionary, disponível em: https://dictionary.cambridge.org/pt/, acesso em 30/12/2020.

CIM Management Schema, disponível em: https://pubs.opengroup.org/onlinepubs/9619599/chap1.htm, acesso em 30/11/2020.

CISCO APIC Rest API configuration Guide, disponível em: https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/2-x/rest_cfg/2_1_x/b_Cisco_APIC_REST_API_Configuration_Guide.html, acesso em 11/11/2020.

CLAISE. B, Clark. J. - Network Programmability with YANG: The Structure of Network Automation with YANG, NETCONF, RESTCONF, and gNMI, Addison-Wesley Professional; 1ª edição (10 maio 2019).

CMDBuild Technical Manual, Version 3.3, disponível em: https://www.cmdbuild.org/file/manuali/technical-manual-in-english, acesso em 12/02/2020.

Como surgiu o Python, disponível em: https://blog.vulpi.com.br/python-como-surgiu/, acesso em 01/12/2020.

DMTF - CIM Model, disponível em: https://www.dmtf.org/standards/cim, acesso em 12/02/2020.

EDELMAN, Jason; LOWE, Scott; OSWALT, Matt (Comp.). Network Programmability and Automation. 2018

Extreme XOS release notes(2018), disponível em: https://documentation.extremenetworks.com/release_notes/ExtremeXOS/22.4.1-Patch1-2/EXOS_Release_Notes/22.4.1-Patch1-2/c_rest_api_using_restconf.shtml, acesso em 11/11/2020.

FRANÇA, C. T. P. Lima – Júnior, J. Celestino – Banco de dados 2º Ed. Editora UECE, 2015. Disponível em: https://educapes.capes.gov.br/bitstream/capes/177824/2/Livro_Computacao_Banco%20de%20Dados.pdf, acesso em 22/11/2020.

Github – extremenetworks/exos_apps, disponível em: do link: https://github.com/extremenetworks/EXOS_Apps , acesso em 11/11/2020.

HPE Aruba Os - Hewlett-Packard Companny. Disponível em: https://support.hpe.com/hpesc/public/docDisplay?docLocale=en_US&docId=emr_na-c05373669 , acesso em 11/11/2020.

HOISEL. D, - “Netbox, um DCIM + IPAM livre feito em Django”, 2016, disponível em.: http://daniel.hoisel.com.br/2016/10/09/netbox-um-dcim-ipam-livre-feito-em-django/#more-486, acesso em 02/02/2020.

HORST, A. S. PIRES, A. DÉO, A. Luis B. De A a Zabbix, Novatec, 2015.

IETF - Dynamic host Configuration Protocol - RFC 2131 - pagina 12, disponível em: https://tools.ietf.org/html/rfc2131 , acesso em 15/11/2020.

IETF – Netconf Configuration Protocol – RFC 4741 – página 5,10,74, disponível em: https://tools.ietf.org/html/rfc4741, acesso em 15/11/2020.

IETF – The YANG 1.1 Data Modeling Language – RFC 7950 – pagina 01, disponível em: https://tools.ietf.org/html/rfc7950, acesso em 01/12/2020.

IETF – YANG A Data Modeling Language for the network configuration protocol – RFC 6020 – , disponível em: https://tools.ietf.org/html/rfc6020, acesso em 01/12/2020.

Infraestrutura – Gigametropole, disponível em: https://www.imd.ufrn.br/portal/rede-giga, acesso em 11/12/2020.

Information Technology Infrastructure Library (ITIL V4), The ITIL 4 Foundation certification is designed as an introduction to ITIL 4 - Pag.188 and 236, 2019

Introduction to YANG programming and RESTCONF on cisco IOS XE, disponível em.: https://networkop.co.uk/blog/2017/02/15/restconf-yang/, acesso em 22/11/2020.

LINGE. Svein, Langtangen. Hans P. - Programming for Computations - Python: A Gentle Introduction to Numerical Simulations with Python. ED. Springer Open, 2016.

MASSÉ M. – Rest API: Design Rulebook, Ed. O'Reilly, 2011.

MDN Web Docs, 2020 - Introdução ao Django, Mozilla Developer Network, disponível em.: https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/Django/Introdu%C3%A7%C3%A3o, acesso em 18/11/2020.

Napalm, disponível em.: https://napalm-automation.net/, acesso em 01/12/2020.

Netbox Documentation, disponível em: https://netbox.readthedocs.io/en/stable/, acesso em 12/02/2020.

NETCONF OVERVIEW, disponível em: https://www.tail-f.com/what-is-netconf, acesso em 30/11/2020.

Networkop – Introduction to YANG Programming and RESTCONF on Cisco IOS XE, disponível em.: https://networkop.co.uk/blog/2017/02/15/restconf-yang/, acesso em 11/11/2020.

O'Donnell. G, Casanova. C - The CMDB Imperative, 2009, Ed. Pearson.

OneCMDB, disponível em: http://www.onecmdb.org/wiki/index.php?title=Main_Page, acesso em 28/11/2020.

Progsoft, iTOP, disponível em: https://progsoft.net/pt/software/itop,m acesso em 21/11/2020.

VINCENT, Willian S. – Django for APIs Ed. Welcome toCode, 2020.

VITALINO, J. F. N. ; CASTRO, M. A. Nunes – Descomplicando o docker, Brasport, 2016.

ZULIANI, E. – O que é python? Parte3: Quando surgiu o Python, disponível em: http://emersonzuliani.com.br/o-que-e-python-parte-3-quando-surgiu-o-python/, acesso em 01/12/2020.