Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

ІНТЕГРОВАНА СИСТЕМА КЕРУВАННЯ АДРЕСНИМ ПРОСТОРОМ IP-МЕРЕЖІ ПІДПРИЄМСТВА

Текстова частина до курсової роботи за спеціальністю „Інженерія Програмного Забезпечення” 121

Керівник курсової роботи
к.т.н., доц. Черкасов
_____
(підпис)
“____” _____ 2021р.
Виконав студент Накитняк В.І.
“____” _____ 2021р.

Київ 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ

к.т.н., доц. Черкасов

_____

(підпис)

“____” _____2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Накитняку Вадиму Ігоровичу факультету інформатики 3 курсу

Тема: Інтегрована система керування адресним простором IP-мережі підприємства

Зміст ТЧ до курсової роботи:

Вступ

1. Огляд існуючих рішень

2. Структурна розробка власного рішення

3. Детальна розробка компонента

Висновки

Джерела

Додатки

Дата видачі “___” _____ 2021 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Тема: Інтегрована система керування адресним простором IP-мережі підприємства.

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	18.10.2020	
2.	Вивчення предметної області та огляд технічної літератури за темою роботи	04.12.2020	
3.	Написання вступу курсової роботи	08.02.2021	
4.	Виконання аналізу сучасних методів	15.02.2021	
5.	Проєктування IPAM-системи	05.03.2021	
6.	Структурна розробка власного прототипу IPAM-системи	31.03.2021	
7.	Написання застосунку мовою програмування Python	30.04.2021	
8.	Розміщення готового застосунку на Apache сервері віртуальної машини Ubuntu	10.05.2021	
9.	Тестування готового застосунку	12.05.2021	
10.	Написання висновку та фінальне коригування текстової частини роботи	13.05.2021	
11.	Надсилання роботи для перевірки на відсутність плагіату	17.05.2021	

Студент _____

Керівник _____

“_____”_____

ЗМІСТ

Вступ

Актуальність теми

Комунікаційна мережа є базовою платформою будь-якої сучасної інформаційної інфраструктури. Від ефективності її функціонування значною мірою залежить успіх діяльності підприємства. Водночас мережа є складною інформаційною системою з великою кількістю компонентів, внаслідок чого важливу роль під час проєктування й експлуатації мереж відіграють засоби автоматизації.

Система автоматизації — це комплексна структура, яка допомагає абстрагуватись від налаштування конфігурацій пристроїв на низькому рівні і перейти до планування цілісної топології. Завдяки абстракції стає можливим налаштування всіх аспектів мережі з єдиного джерела інформації.

Одним з важливих завдань системних адміністраторів є керування адресним простором мережі. Засобами автоматизації розв`язанням цієї задачі є система управління розподілом адресного простору — IP Address Management(IPAM).

Завдяки цим засобам, адміністраторам надається можливість відстежувати стан топології в будь-який момент часу; якісно виділяти та редагувати IP-адреси; ділити та масштабувати мережу за допомогою **VLAN** та **VRF**; створювати звіти про пристрої, користувачів і пул зайнятих або вільних адрес у тій чи інший підмережі підприємства; групувати підмережі.

До переліку задач, з якими має діло системний адміністратор, завжди входило призначення IP-адрес та розбиття мережі на VLANи. З розвитком технологій кількість таких задач пропорційно зростала. Сервіс **DHCP** додав необхідність перевіряти, яку адресу призначено тому чи іншому пристрою. Актуальними стали перехід до нового протоколу третього(мережевого) рівня, а саме **IPv6**, а також поділ мережі на логічні сегменти за допомогою протоколу VRF. Для того, щоб полегшити цей процес, була розроблена система автоматизації, одним з основних компонентів якої якраз і є управління розподілом адресного простору.

Серед ризиків неефективного управління простором IP-мережі є простій мережі, втрачені компанією доходи, незадоволені клієнти, а також проблеми з безпекою.

Існує багато сценаріїв некоректної поведінки мережі, спричинених неправильними налаштуваннями конфігурацій на пристроях. Іноді, навіть маленька помилка, може призвести до зупинки роботи сервісів цілої корпорації. Неможливо позбутись помилок спричинених людським фактором, проте можливо зменшити їх кількість. Автоматика якраз і є тим механізмом, що дозволить нам здійснювати менше помилок. Ми маємо бути впевнені, що виділяючи нову IP-адресу для **loopback**(внутрішня петля) інтерфейсу комутатора, ми не зіштовхнемося з проблемою, що він уже комусь призначений. Або що один і той самий префікс було використано двічі в різних кінцях мережі[1].

Ще однією з задач системного адміністратора, є документування мережі. Добре написана документація може зберегти час та гроші. В довгостроковій перспективі це полегшить **troubleshooting**(налагодження) мережі і вам, і вашим колегам по роботі. IPAM робить мережу документованою, а також допомагає уникнути наслідків поганого дизайну.

Серед інших переваг таких засобів автоматизації:

- Аутентифікація, авторизація та облік користувачів;
- Розподіл обов`язків по управлінню мережею, надання прав користувачам;
- Зручний перехід до **IPv6**;
- Зручний інтерфейс програми;
- Звітування (можливість отримати потрібну інформацію по кожному пристрою у мережі);
- Зручна робота з даними (можливість експортувати/імпортувати, візуалізувати);

Просунутою версією IPAM є технологія **DDI** — DNS-DHCP-IPAM. DDI об`єднує всі переваги IPAM і додає інтеграцію з **DNS** і **DHCP** сервісами, а також

централізований репозиторій із резервним копіюванням, що автоматично підлаштовується під план IP-адресації. Це майже повністю усуває невідповідність даних на логічному та фізичному рівнях топології[2].

Постановка задачі

Серед задач, які ставить дана робота:

- Розглянути технологію автоматизації IPAM, як один з найкращих методів управління адресним простором мережі підприємств;
- Провести порівняльний аналіз існуючих рішень на ринку;
- Розглянути сучасні методи впровадження/інтеграції системи IPAM у існуючу комунікаційну мережу;
- Ознайомитись з трирівневою архітектурою, на якій базується більшість сучасних IPAM-систем;
- Провести структурну розробку власного рішення, з детальним описом архітектури, компонентів системи, їх характеристик, а також можливих платформ реалізації такої системи;
- Розробити окремий компонент IPAM-системи.

Основна мета

Основною метою роботи є дослідження ефективних методів управління адресним простором мережі підприємств.

1. Огляд існуючих рішень

Сьогодні на ринку існує безліч інструментів, що реалізують задачу автоматизації IP-простору. Перевагу надано тим, що поширюються на **open-source**(відкрите програмне забезпечення) основі, проте для чистоти експерименту розглянуто і комерційні. Функціоналу безкоштовних додатків зазвичай вистачає для домашнього використання або малого та середнього підприємства.

Також одним з головних факторів при виборі, була наявність документації, інструкції або тестової версії сайту/застосунку з акаунтом доступу адміністратора, що наочно демонструє інтерфейс і наповненість.

Розглянуто наступні застосування: NetBox, phpIPAM, SolarWinds. Ці IPAM-системи в тому чи іншому вигляді відображують поточний стан технології управління адресним простором мережі на поточний момент.

1.1. Базовий функціонал

Спочатку розглянемо перелік функцій, якими володіють більшість сучасних IPAM-систем[3].

- Імпортування та експортування таблиць у форматі **XLS/CSV**;
- Підтримка протоколу IPv6:
- Програмний інтерфейс RESTful API;
- Поділ користувачів на групи;
- Поділ підмереж на групи;

Для наочності було створено порівняльну таблицю:

Features\IPAM	NetBox	phpIPAM	SolarWinds	Infoblox
License	Apache	GPLv3	Commercial	Commercial
Latest release	2.10.3	1.4.1	4.6	8.3.2
IPv6 support	Yes	Yes	Yes	Yes
VRF support	Yes	Yes	No	Yes
Database support	PostgreSQL	MySQL,MariaDB	Microsoft SQL Server	Embedded real-time distributed database
Platform	Linux	Linux	Microsoft	Windows, Linux
DNS	No	PowerDNS	Yes	Yes
DHCP	No	No	Yes	Yes
NAT support	No	Yer	No	Yes
Authentication	LDAP authentication using an external server or local user	AD, LDAP, Radius	Radius	Windows AD, TACACS, or local user
Network discovery	No	Yes	Yes	Yes(NetMRI)
HTTP server support	nginx/Apache	nginx	Microsoft IIS	nginx/Apache
Import	XLS / CVS subnets import	XLS / CVS subnets import	XLS / CVS subnets import	CSV
Export	XLS / CVS subnets export	XLS / CVS subnets export	XLS / CVS subnets export	CSV
API	RESTful	RESTful	RESTful	RESTful

Рисунок 1 Порівняльна таблиця існуючих рішень

1.2. NetBox

1.2.1. Короткий огляд

NetBox – відкритий вебзастосунок для керування адресним простором IP-мережі й документування, розроблений компанією DigitalOcean. Основні принципи NetBox, за словами розробників: «Replicate the world, serve as source of truth, keep it simple[4].»

Поряд із сучасним вебінтерфейсом, NetBox зручний з точки зору організації мережі. Він допомагає документувати IP адресацію, роблячи основний наголос на пристроях, системній інфраструктурі та віртуальних машинах. Це дозволяє йому бути ідеальним вибором як **Source of Truth** для системи автоматизації.

1.2.2. Основні переваги

- **Open-source**;
- IPAM + DCIM;
- Віртуалізація;
- Секрети;
- RESTful API-інтерфейс;
- Підтримка Webhooks;
- Source of Truth;
- Активно підтримується(Slack, Github) та оновлюється;

1.2.3. Встановлення і тестування

В якості дослідження було встановлено NetBox на віртуальній машині Ubuntu 20.4 і запустив **nginx** сервер. З домашньої операційної системи шляхом **bridge adapter** приєднаємось до сервера. Ось як виглядає стартовий екран:

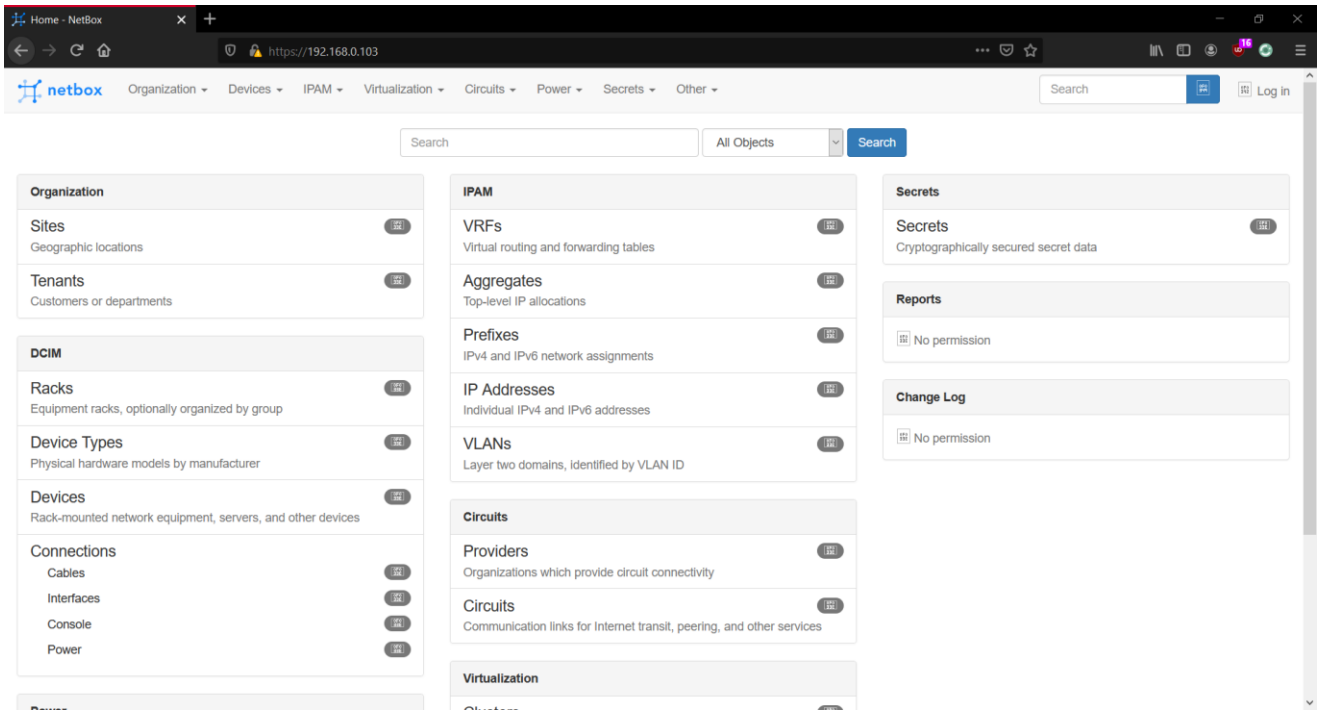


Рисунок 2 Вебінтерфейс IPAM застосунку NetBox

Інтерфейс програми є дуже мінімалістичний, але одночасно доволі функціональним. Для людини, знайомої з усіма технологіями, не виникне труднощів почати працювати з системою NetBox.

1.2.4. IPAM

Розглянемо механізми адміністрування IP-простору в NetBox.

Підтримується багато видів фізичних інтерфейсів, в тому числі **LAG**(link aggregation link). Всі віртуальні інтерфейси об`єднані під єдиним типом – **Virtual**. Кожен інтерфейс прив`язаний до цільового пристрою, що дозволяє бачити які з інтерфейсів взаємно підключені. VLAN і VRF можуть бути прив`язані також і до локації, що особливо корисно для VLAN. При створенні VRF можна вказати, чи допускається перетин адресного простору з іншими VRF.

IP-префікси мають ієрархічну структуру і можуть належати VRF(якщо не належать, тоді - **Global**). Всі зайняті або вільні префікси наочно представлені у вигляді таблиці, з процентним співвідношенням зайнятих адрес.

Щодо IP-адрес, то вони можуть належать IP-префіксам, або існувати окремо. Основна їх функція – прив`язка до інтерфейсів пристроїв[5].

1.2.5. DCIM

DCIM - система інвентаризації пристроїв мережі. Це дозволяє зберігати всю інформацію про обладнання і його місце розташування. Для цього у NetBox розроблено вкладену систему з регіонами й сайтами, а кожен пристрій має роль, наприклад: **leaf**, **spine**, **edge**, **border**. Додатково можна організовувати стійки для пристроїв. Вони організовані за сайтами, групами та ролями.

1.2.6. Virtualization

Для представлення віртуалізації у NetBox існують сутності кластерів. Кластер - логічне угрупування фізичних ресурсів, що підтримують віртуальні машини. Віртуальні машини відображують віртуальні комп`ютери, прив`язані до кластера. Так само як і на фізичних пристроях, можливо призначати IP-адреси на інтерфейсах віртуальних пристроях

1.2.7. Secrets

Секрет являє собою один обліковий запис або інший конфіденційний рядок символів, який слід надійно зберігати. Кожен секрет може також зберігати необов’язковий параметр імені, який не шифрується. Це може бути корисно для зберігання імен користувачів.

Загалом, цей механізм надає захищеності системі, бо шифрує сховище облікових даних користувачів.

1.2.8. RESTful API

Як і більшість сучасних систем NetBox підтримує RESTful API інтерфейс. Однією з основних переваг RESTful API є його зручність для людини. Оскільки він використовує **HTTP** та **JSON**, тому дуже легко взаємодіяти з даними NetBox у командному рядку за допомогою загальних інструментів. API надзвичайно зручний, оскільки підтримує операції CRUD (створення, читання, оновлення, видалення) і повністю задокументований документацією Swagger.

1.2.9. Source of truth

Варто підмітити, що NetBox показує не реальний стан мережі, а цільовий. Іншими словами, NetBox не проводить сканування мережевих ресурсів і нічого не відвантажується напряму з мережі до NetBox. Таким чином, NetBox виступає як єдине джерело істини. Всі зміни на мережі мають бути ініційовані першочерговими змінами у NetBox. Адміністратори самостійно мають підтримувати дані системи у **up-to-date** стані[6].

1.2.10. Підтримка WebHooks

Webhooks - це спеціальний метод сповіщення вебсторінки за допомогою спеціальних зворотних викликів. Завдяки підтримці webhooks NetBox дозволяє інтеграцію з будь-якими онлайн сервісами.

Наприклад, ви можете налаштувати система на автоматичну систематизацію та генерацію звітів, які опісля надсилають HTTP запити в інший вебзастосунок, наприклад корпоративний чат у **Slack**. Або при створенні нового комутатора, автоматично надсилати webhook в систему автоматизації, яка запускає новий процес налаштування і введення в експлуатацію пристрою.

Так само можна дописати окремий модуль, або скрипт, який буде приймати webhooki з інших сервісів для керування програмним забезпеченням, наприклад через фреймворк Django оброблювати webhook запити з сайту Servicenow[7].

1.2.11. Недоліки

Одним з найбільших недоліків програми, є відсутність інтеграції з сервісами DNS, DHCP.

1.2.12. Перелік технологій

Функція	Компонент
HTTP сервіс	nginx або Apache(Оброблює HTTP запити)
WSGI сервіс	gunicorn або uWSGI (WSGI middleware)
Фреймворк	Python/Django (Високорівневий фреймворк для розробки вебсистем)

База даних	PostgreSQL 9.6+ (Реляційна база даних)
Черга задач/кешування	Redis/django-rq (розподілене сховище пар ключ-значення, для збереження в оперативній пам`яті)
Доступ до пристроїв	NAPALM (Python бібліотека для взаємодії з мережевими пристроями)

1.3. phpIPAM

1.3.1. Короткий огляд

phpIPAM – відкритий вебдодаток для управління IP-адресами. Перша версія проєкту побачила світ у 2014 році за публікації Міхи Петковшека, що фактично і є одноосібним розробником. Проєкт розпочався як особистий, проте наразі база дописувачів на сторінці GitHub перевищує 100 користувачів. Проєкт розробляється за моделлю вільного програмного забезпечення, а саме ліцензією **GNU 3**[8].

1.3.2. Основні переваги

- Open-source;
- Багато форм аутентифікації;
- RESTful API-інтерфейс;
- Network discovery;
- Інтеграція з PowerDNS;
- Source of Truth;

1.3.3. Тестування demo сайту

Попри відсутність повноцінної документації, сайт проєкту містить посилання на демо сервер. Пропонується вхід з двох акаунтів: адміністратора та звичайного користувача:

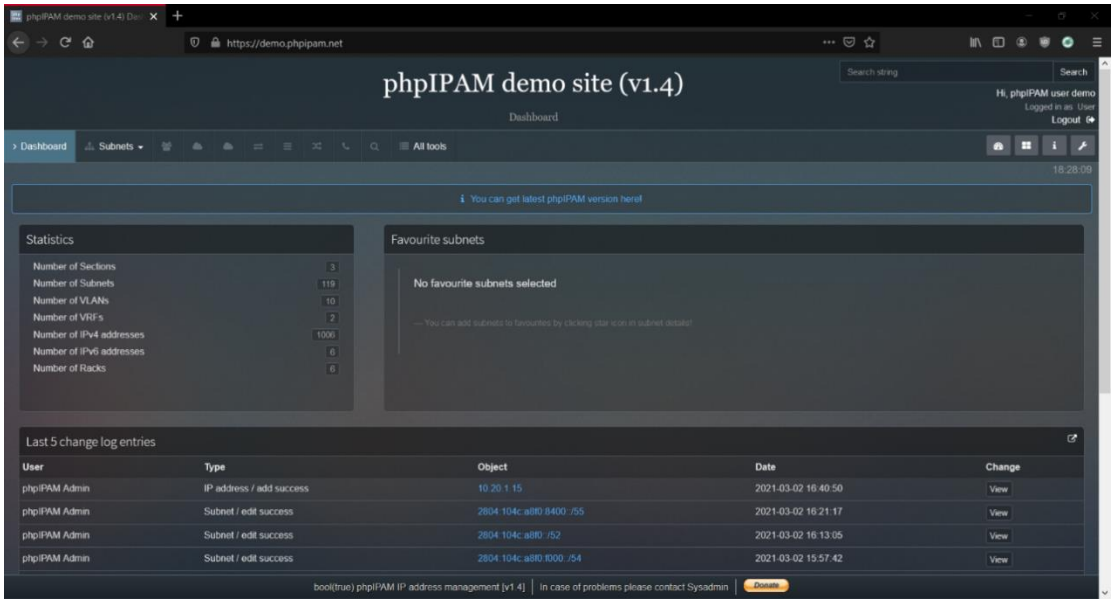


Рисунок 3 Вхід до phpIPAM з акаунту звичайного користувача

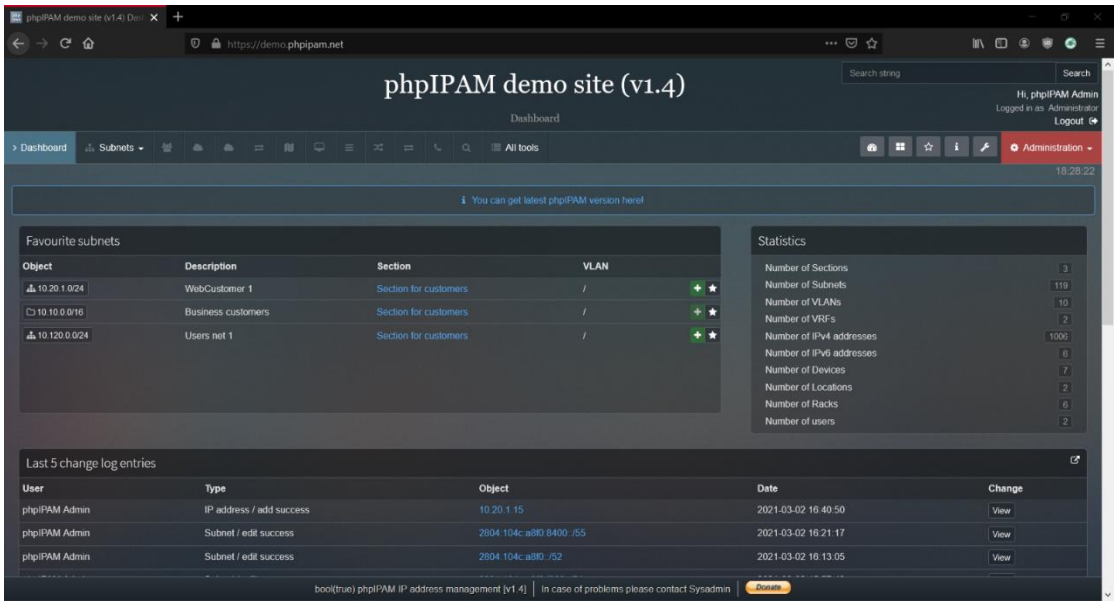


Рисунок 4 Вхід до phpIPAM з акаунту адміністратора

Основна відмінність в тому, що права користувача урізані, тому змінити або подивитись деякі модулі неможливо.

При вході на сайт, нас одразу зустрічає дошка оголошень, на якій агреговано загальну інформацію про поточний стан мережі:

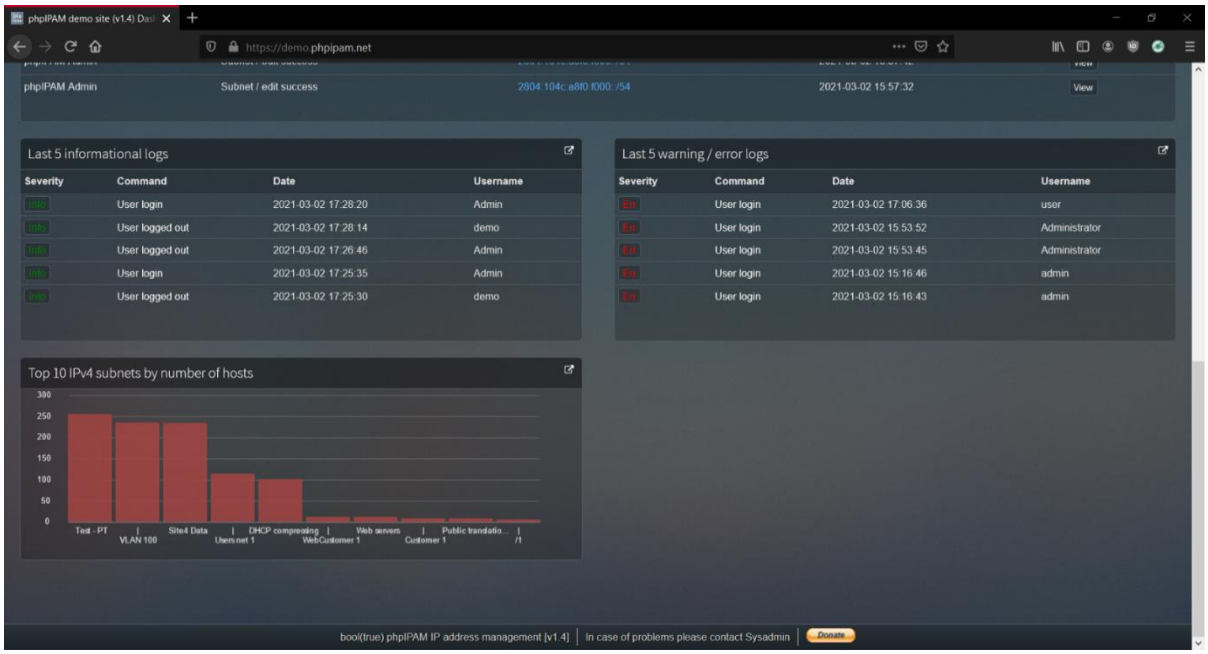


Рисунок 5 Поточний стан мереж в phpIPAM

В цілому інтерфейс доволі звичний. Серед усього іншого, виділяється лише вкладка “All tools” з переліком всіх утиліт застосунку, серед яких: калькулятор IP-адрес, сканування мережі, логування, стрічковий пошук та інші:

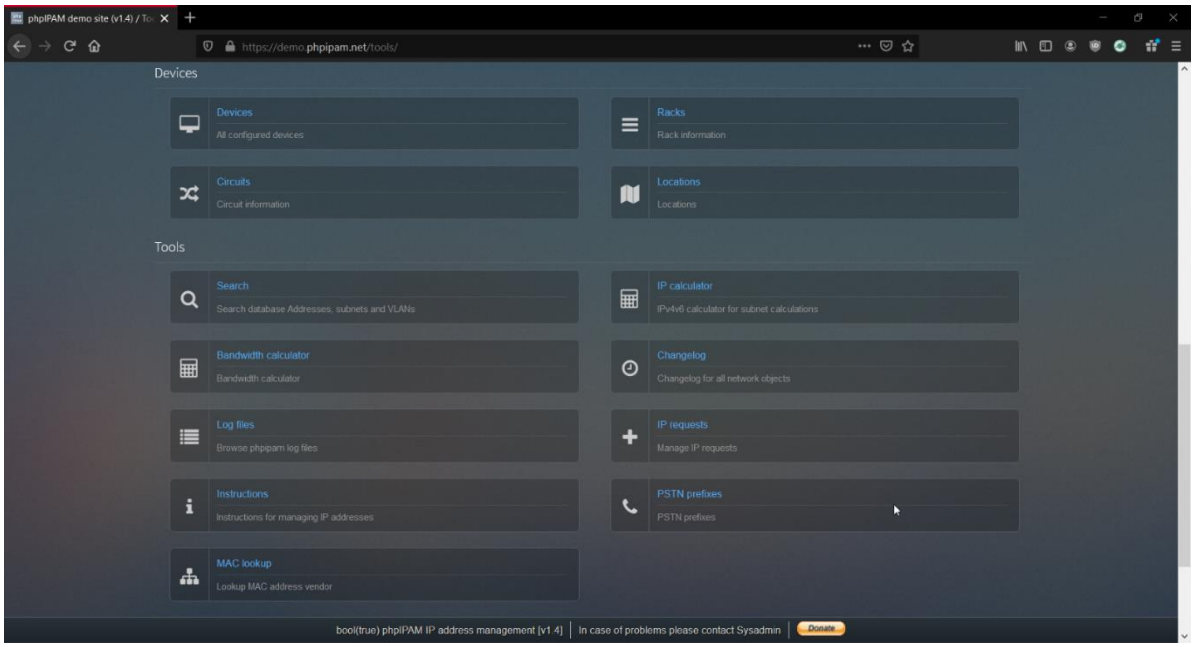


Рисунок 6 Неповний перелік інструментів phpIPAM

1.3.4. IPAM

Завданнями механізмів менеджменту IP-мережі у phpIPAM є відстежувати та зберігати інформацію в реляційній базі даних MySQL/MariaDB. Система підтримує збереження даних про наступні типи[9]:

- Пристрої та їх інтерфейси
- IP-адреси;
- Комп`ютерні стійки;
- Схеми (за допомогою Google Maps APIs);
- Дані про користувачів та групи користувачів;
- VLAN підмережі та групи мереж;
- Інформацію про NAT;
- Нотатки про пристрої;

1.3.5. Network discovery

З версії 0.9 в phpIPAM стала доступна опція сканування мережі. Система автоматично перевіряє статус сконфігурованих пристроїв на вибраних підмережах, а також виявляє нові хости. Цей процес здійснюється за допомогою протоколу ICMP. З появою pcntl (модуль підтримку управління процесів в PHP, що реалізує стиль управління процесами UNIX) стало можливим робити паралельні ICMP перевірки, що значно скоротило час на перевірку.

В налаштуваннях можна обрати інтервал пінгування, а також підмережі, що підлягають скануванню.

Процес сканування запускається **Unix**-утилітою **cron**[10].

1.3.6. PowerDNS

PowerDNS – це відкритий DNS-сервер для UNIX-подібних операційних систем. Він виступає в ролі альтернативи традиційній прив`язці імен DNS.

PowerDNS пропонує кращу продуктивність і менші витрати пам`яті. В якості бекенду може приймати як звичайні файли, так і структуровані бази даних (у випадку phpIPAM це MySQL).

Одна з ключових переваг PowerDNS, це підтримка DNS Security Extensions, а також зручний вебінтерфейс Poweradmin, який пропонує багато інструментів для менеджменту.

Вбудований модуль phpIPAM дозволяє вказати налаштування, такі як підключення до бази даних, а потім автоматично синхронізує всі зміни з сайтом Poweradmin[11].

1.3.7. Аутентифікація

phpIPAM має вбудований механізм аутентифікації, але також інтегрований зі службами **LDAP** або **Active Directory**. Ось повний перелік всіх можливих форм аутентифікації:

- Локальне сховище.
- Apache сервер.
- Active Directory (AD).
- Протокол LDAP.
- NetIQ.
- Radius сервер.
- SAMLv2.

Детальніше розглянемо один з наведених методів, а саме Active Directory.

Спочатку потрібно мати акаунт в Active Directory. Потім потрібно створити нового користувача в phpIPAM(бажано з тим самим ім`ям користувача), налаштувати його права і приналежність до групи. З переліку створюємо новий метод аутентифікації та вводимо дані з Active Directory акаунта. В налаштуваннях користувача обираємо новостворений метод аутентифікації і пробуємо увійти.[12].

1.3.8. Недоліки

Єдиним недоліком системи phpIPAM, є відсутність повноцінної документації. На головному сайті проєкту можна ознайомитись лише з переліком можливостей або інсталяційними гайдами. Проте це компенсується наявністю демоверсії сайту. Також, можливо задовгий час оновлення системи, через відсутності повноцінної команди розробників.

1.3.9. Перелік технологій

Функція	Компонент
HTTP сервіс	nginx або Apache
Фреймворк	PHP 5.4+
База даних	MySQL 9.6+/ MariaDB
Вебінтерфейс	jQuery, ajax, HTML5/CSS3

1.4. SolarWinds IPAM

1.4.1. Короткий огляд

SolarWinds IPAM – пропрієтарне програмне забезпечення для управління адресним простором мережі, що розробляється компанією SolarWinds. Діяльність компанії спрямована на автоматизацію IT процесів бізнесу, серед яких управляння комунікаційними мережами та інфраструктурою інформаційних технологій в цілому. SolarWinds IPAM входить в набір інструментів, що поставляється компанією SolarWinds на платформі Orion.

Ось як виглядає інтерфейс головного екрану:

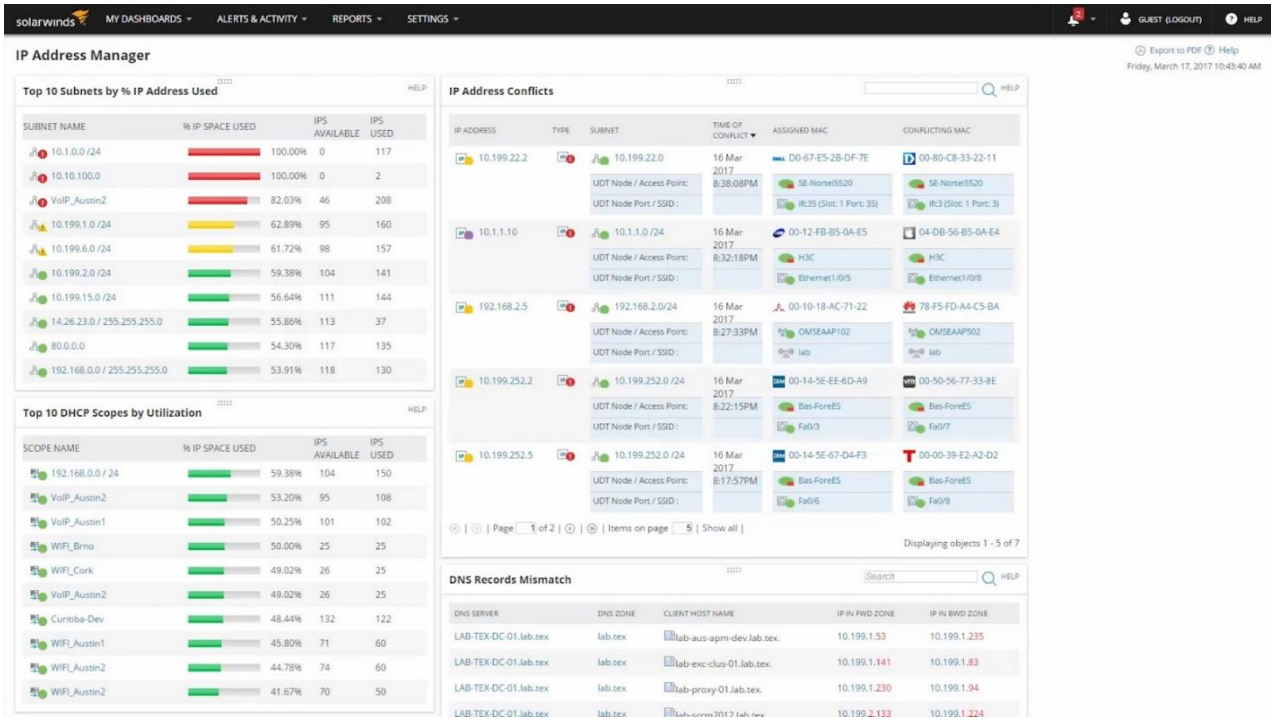


Рисунок 7 Вебінтерфейс IPAM системи SolarWinds

1.4.2. Основні переваги

- DDI;
- Виявлення мережі;
- Система оповіщень та звітування;
- Адміністрування на основі ролей;
- RESTful API-інтерфейс;
- Зручна документація;

1.4.3. DDI

IPAM працює з DHCP та DNS від різних постачальників. Отже, для роботи з ним не потрібно встановлювати додаткове пропрієтарне програмне або апаратне забезпечення. Зміни, які ви вносите в IPAM автоматично поширюються на відповідні сервіси, що дозволяє легко знаходити та налаштовувати IP-адреси з єдиної консолі управління.

Крім того, дзвінки **WMI**(інструментарій управління Windows) на сервери DHCP та DNS здійснюються для отримання детальної інформації про оренду та

сферу використання. Ви можете зробити резервування DHCP та записи DNS для бронювання IP-адрес одночасно з одного екрану.

1.4.4. Network Discovery

IPAM використовує ICMP і SNMP протоколи, а також **neighbor scanning** технологію, для того, щоб збирати інформацію з пристроїв у мережі, і використовує дані для відстеження і зображення використаних IP-адрес та автоматичного позначення IP-адрес, які більше не використовуються.

1.4.5. Система оповіщень та звітування

SolarWinds IPAM автоматично виявляє проблеми, пов`язані з IP-адресами, які в кінцевому результаті можуть призвести до порушення мережі. Основні механізми впровадження такої системи: безперервний моніторинг мережі, сповіщення про будь-які зміни IP адресації в мережі та доступ до даних про використання основних показників планування. Активний моніторинг та попередження допомагають превентивно усунути простої мережі через конфлікти IP-адрес або відсутність IP-адрес, спричинену заповненням підмережі.

1.4.6. Адміністрування на основі ролей

SolarWinds IPAM дозволяє відокремити управління DHCP, DNS та IP підмереж по групах користувачів. Кожна група може управляти власними сегментами, блоками адрес і DNS, DHCP сервісами не впливаючи на роботу колег. Такий підхід дозволяє підтримувати безпеку без обмеження делегування діяльності з управління IP-адресами[13].

1.4.7. Перелік технологій

Функція	Компонент
HTTP сервіс	Microsoft Internet Information Server
Фреймворк	C# .NET 4.6.2
База даних	SQL Server
Операційна система	Windows Server 2016

1.4.8. Недоліки

По-перше, зависока ціна. Ціна може варіюватись в залежності обраної ліцензії, вічної та річної, та починається з1774 або 947 євро за технічне обслуговування на рік відповідно. Також ліцензії відрізняються доступною кількістю IP-адрес мережі(1024, 4096, 16384 або безмежна).

По-друге, прив`язка до операційної системи Windows server.

По-третє, відсутність підтримки VRF.

1.5. Варті згадки

Окрім вже розглянутих IPAM-систем, варто згадати ще деякі:

- Microsoft IPAM – IPAM-система, що йде з коробки на операційних системах Windows Server;
- Infoblox IPAM – незмінний фаворит серед IPAM-систем, частка ринку якої становить 15% (найбільший показник серед конкурентів).

2. Структурна розробка власного рішення

Існує декілька архітектурних підходів до проєктування IPAM-системи та її інтеграції в мережу підприємства. Одним з таких підходів, є використання IPAM-системи як Source of Truth для всієї мережі підприємства.

2.1. Схема інтегрованої IPAM-системи

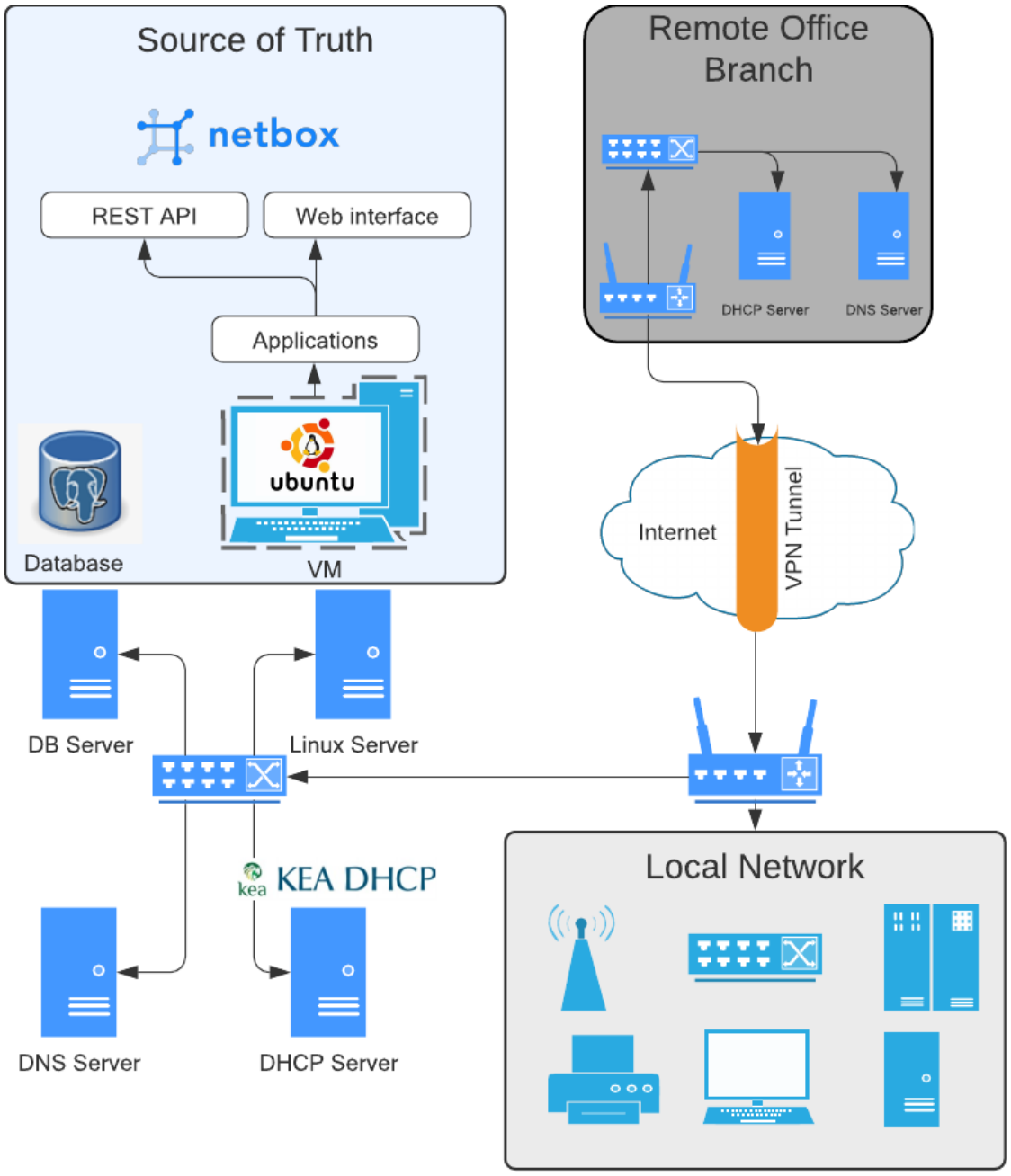


Рисунок 8 Схема IPAM системі в мережі підприємства

На указаній схемі зображено використання IPAM-застосування(на прикладі NetBox) в мережі підприємства.

Для наочності схему розділено на декілька блоків Source of Truth, Local Network та Remote Branch Office.

Source of Truth (SOT) – єдине джерело істини, один із способів проєктування інформаційних систем. Він полягає у тому, що дані про кожний елемент системи консолідовано в єдиному місці. В нашому випадку, це інформація про призначення IP адрес та DNS імен на пристроях.

Local Network – локальна мережа підприємства. Тут зазвичай розміщуються пристрої рівня розподілення, кінцеві пристрої(принтери, персональні комп`ютери), точки доступу, тощо.

Remote Branch Office – віддалений офіс. Зазвичай мережа підприємства не обмежується парою кімнат офісної будівлі. Це може бути велика **WAN**-мережа, що поєднує в собі декілька офісів з різних кінців одного міста, або навіть з різних міст. Для того, щоб об`єднати декілька віддалених мереж в одну, використовують технологію **VPN**. Створюється так званий тунель через мережу інтернет, що передає дані в зашифрованому вигляді між роутерами різних офісів. Таким чином, відбувається імітація того, ніби всі знаходять в єдиній локальні мережі.

2.2. Організація Source of Truth

А тепер детальніше про підхід до організації **SOT**. Всі зміни на пристроях попередньо мають бути занесені в базу даних системи. **SOT** показує бажаний стан системи, а не робочий. Дані, створені таким чином, після цього перевіряються адміністратором для забезпечення цілісності. Після затвердження, цю базу вже можна використовувати для наповнення систем моніторингу(напр. **Nagios**) та систем забезпечення мережі (**network provisioning system**, напр. **Cisco DNA**), і надалі для внесення змін на пристрої[14].

Таким чином, всі зміни мають централізовано відбуватись через програмне забезпечення з єдиного джерела. Всі елементи системи синхронізуються з **SOT**, і після цього, з джерела можна дізнатись поточний стан мережі.

Для організації **SOT** потрібен Linux Apache/nginx сервер з встановленим програмним забезпеченням NetBox, доступ до якого здійснюється по 80 порту одночасно на два інтерфейси - програмний та веб.

На іншому сервері встановлюємо реляційну базу даних **PostgreSQL**, з якою буде спілкуватись **NetBox**.

Тепер перейдемо до інтеграції з **DNS** та **DHCP**.

2.3. Інтеграція з DNS

Для інтеграції з сервісами **DNS**, нам потрібен робочий **DNS**-сервер. Для забезпечення найкращої працездатності, копії DNS-сервера(те саме стосується і DHCP) мають бути встановлені в усіх офісах.

Для того, щоб зв`язати нашу IPAM-систему з цими серверами, нам потрібен конектор – плагін або скрипт, який здійснює спілкування IPAM з DNS. Для NetBox є такий, під назвою **netbox-ddns** – динамічний DNS-з`єднувач для NetBox. Цей плагін дозволяє визначати DNS-сервери, що підтримують протокол **RFC3007**(протокол динамічного оновлення DNS). Для кожного серверу ви вказуєте за які домени та зворотні DNS-домени він відповідає. Після цього, NetBox автоматично надсилатиме оновлення цим серверам кожного разу, як змінюватиметься DNS-ім`я IP-адреси в Netbox. Для відстеження прогресу цих оновлення можливо переглянути консоль з увімкненим логуванням фреймворку Django, або переглянути фонові завдання в режимі адміністратора NetBox. Цей плагін підтримує версії NetBox (2.8, 2.9 та 2.10)[15].

2.4. Інтеграція з DHCP

З **DHCP** все складніше. Інтеграцію NetBox з **DHCP** продемонстровано на прикладі **KEA DHCP**-серверу.

Як вже було сказано, за принципом **SOT**, все визначається у NetBox: пристрої, віртуальні машини, інтерфейси з MAC-адресами, призначаються IP-адреси на інтерфейси. Це забезпечує достатньо інформації для налаштування

DHCP-сервера, який передаватиме пристроям адреси, визначені в NetBox, при запиті IP-адреси пристроями.

KEA – це сучасний DHCP-сервер, який надає API для його налаштування та управління. Є можливим використати цей програмний інтерфейс для того, щоб напряму надсилати команди з NetBox, проте модуль з цієї можливістю доступний лише як платний сервіс.

Іншим варіантом, є скрипт, що на основі REST API NetBox, генерує файли у форматі JSON з даних, про призначення IP-адрес на інтерфейси, і надсилає їх як конфігурацію на **DHCP**-сервери. Це працює, завдяки директиві <?include?> в DHCP KEA.

Цей скрипт повинен спрацьовувати кожного разу, коли змінюється інформація про інтерфейси в NetBox. Для цього в NetBox існують webhooки(більше про це у розділі “Порівняльний аналіз”)[16].

2.5. Альтернативи

В цілому, NetBox чудова відкрита платформа для IPAM-системи, яку можливо розширювати під свої потреби. Звичайно, існують простіші варіанти організації такої системи, проте, зазвичай це платні сервіси. Можна скористатися готовими рішеннями для автоматизації мережі, такими як SolarWinds Network Automation Manager/ SolarWinds Network Configuration Manager, Red Hat Ansible Automation Platform.

3. Детальна розробка IPAM-компонента

Основу IPAM-системи, наведеної в схемі, складає програмне забезпечення NetBox, запущене на HTTP-сервері. Для демонстрації його функціонування, було розроблено спрощений аналог.

Основна мета – розробити програму, що являє собою бекенд повноцінної IPAM-системи.

3.1. Основний функціонал

- Створення, видалення та редагування даних про пристрої в мережі підприємства;
- Створення, видалення та редагування даних про інтерфейси пристроїв;
- Створення, видалення та редагування даних про IP-префікси мережі, виведення інформації про використання IP-адрес префіксів;
- Створення, видалення та редагування даних про призначення IP-адрес на інтерфейси пристроїв;
- Створення, оновлення та редагування даних про віртуальні підмережі – VLAN;
- Створення, оновлення та редагування даних про сервіси 4-го рівня, запущені на пристроях;
- Створення, оновлення та редагування даних про локації, та прив`язку IP-підмереж до них;
- Створення, оновлення та редагування даних про стійки;
- Створення, оновлення та редагування даних про функціональні ролі;
- Створення, оновлення та редагування акаунту користувача системи;
- Реалізація аутентифікації/авторизації/обліку користувачів:
    - Аутентифікація користувачів в систему;
    - Авторизація на основі груп та прав доступу окремих груп до певних ресурсів, обмеження на додавання, редагування, видалення даних;

- Логування дій користувачів, пов`язаних з додаванням, редагуванням та видаленням даних про пристрої, інтерфейси і т.д. в системі.

- Вивід розширеної інформації про призначення IP-адрес в мережі: вивід процентного співвідношення зайнятих адрес, вивід вільних пулів адрес, тощо;
- Надання адміністратору інтерфейсу створення груп користувачів, призначення користувачів до цих груп, надання цим групам прав доступу;
- Надання адміністратору право передивлятись всі дії користувачів;

3.2. Особливості застосунку

- IPAM + DCIM, поєднання засобу керування призначенням IP-адрес та інвентаризацію мережі;
- Програмний(API) та вебінтерфейси;
- Документований API інтерфейс, по якому легко орієнтуватись;
- Зручний інтерфейс адміністратора;
- Вивід інформації про поточний стан мережі;
- Легке розгортання на сервері;

3.3. Стек технологій

Щодо технологій реалізації, я було використано мову програмування **Python** разом з каркасом **Django**, а в якості реляційної бази даних, раніше зазначену **PostgreSQL**[17].

Функція	Компонент
HTTP сервіс	Apache
Фреймворк	Python/Django
База даних	PostgreSQL 9.6+
Вебінтерфейс	HTML, css, AJAX(бібліотека для мови javascript), Bootstrap5(бібліотека стилів)

Переваги мови програмування Python та каркасу Django:

- Модульність;

- Наявність бібліотек для підключення по протоколу **SSH**(NAPALM, Netmiko);

- Наявність бібліотек для полегшення маніпуляції мережевих адрес(netaddr, ipaddress, pythonping);

- Наявність вбудованого шаблонізатору в Django(це надає змогу легко і швидко проєктувати вебінтерфейс програми на основі бекенду, тобто відсутня потреба запускати виділений для фронтенду програми сервер, що є ідеальним вибором для підприємства);

- Вбудована в Django технологія **ORM**(об`єктно-реляційне відображення);

- **Django REST** пакунок, який дозволяє реалізовувати RESTful API програмний інтерфейс на Django;

- Вбудований компонент аутентифікації.

Для ведення розробки було використано систему контролю версій **Git**, платформу **GitHub**, і середовище інтегрованої розробки **PyCharm**.

3.4. Структура застосунку

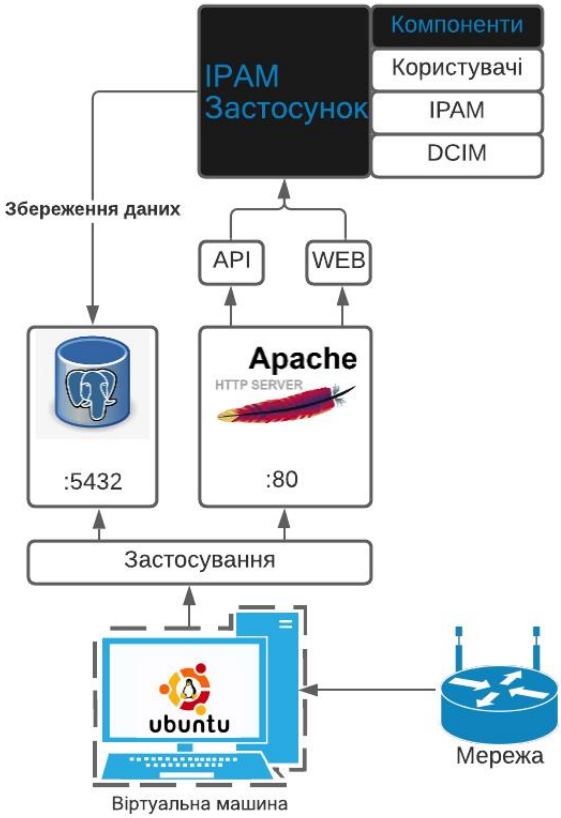


Рисунок 9 Схема структура власного IPAM застосунку

Структурно компонент складається з трьох частин: “Користувачі”, “IPAM”, “DCIM”. Доступ до цих компонентів здійснюється програмним шляхом, через RESTful API, або через вебінтерфейс сторінки браузера по порту 80. Дані з цих модулів зберігаються в базі даних, в нашому випадку, запущеній на віртуальній машині на порті 5432. Віртуальна машина напряму під`єднана до мережі, тому є можливість пінгувати пристрої в мережі та перевіряти їх доступність.

3.5. Інтерфейси застосунку

Застосунок складається з двох інтерфейсів:

1. Вебінтерфейс, побудований на формах і мові розмітки HTML;
2. Програмний інтерфейс, RESTful API, для керування програмою з консолі використовуючи команду **curl**, або утиліту командного рядка **httpie**; напряму через браузер; написанням скриптів автоматизації;

Моделі та бізнес-логіка для двох інтерфейсів спільні.

3.6. Опис функціональних компонентів

Проєкт складатиметься з трьох компонентів: “Користувачі”, “IPAM”, “DCIM”. Кожен компонент, своєю чергою, поділяється на окремі модулі. Модулі містять моделі.

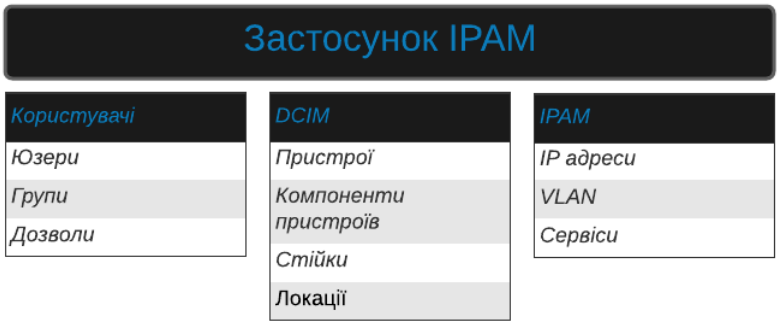


Рисунок 10 Компоненти та модулі власного IPAM застосунку

3.6.1. Пояснення до схем ER-моделей

При побудові схем **ER-моделей**(модель сутність-зв`язок) використовувалася нотація “вороняча лапка”. На прикладі зв`язків між “Група стійок” та “Стійка” розглянемо це.

Рисунок 11 Приклад нотації "вороняча лапка"

У цьому прикладі показано необов’язковий зв’язок між групою стійок та стійкою; символи, найближчі сутності “Стійка”, представляють "нуль, один або багато", тоді як стійка належить "одній і тільки одній" групі. Тому перше читається як “Група стійок” може мати “нуль, один або багато” “Стійок”.

Також в схемі присутній циклічний зв`язок. На прикладі сутності “Регіон” розглянемо це:

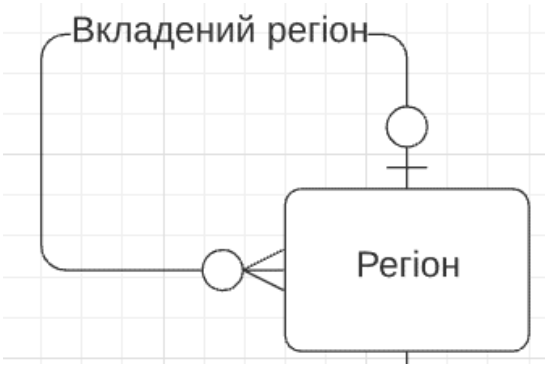


Рисунок 12 Приклад рекурсивного зв`язку в ER-моделі

У сутності “Регіон” існує зв`язок з самим собою. Він читається як: “Кожен регіон необов`язково може належати батьківському регіону. Кожен регіон має нуль, один або багато вкладених регіонів”

3.6.2. Компонент “Користувачі”

Це стандартний компонент, який відповідає за автентифікацію, авторизацію та акаунтинг.

Юзери

Для того, щоб почати роботу з IPAM, потрібно залогінитись. Для нових користувачів передбачено функцію реєстрування(також, адміністратор має можливість створювати нові акаунти). Щойно зареєстрований користувач має стільки ж прав, як і анонімний, тому для початку роботи, потрібно попросити адміністратора додати його в групу доступу, з правами на читання даних, або редагування, в залежності від позиції, яку людина займає в підприємстві.

Кожен користувач може змінювати контактні дані про себе.

Адміністратор

Адміністратор, це суперкористувач, який має доступ до інтерфейсу адміністратора, а також за замовчуванням має всі права.

Адміністратору надається повний перелік можливостей для ефективного управління користувачами. Він може створювати, призначати користувачів в групи, блокувати користувачів, переглядати історію внесених змін в систему обраного користувача.

Моделі

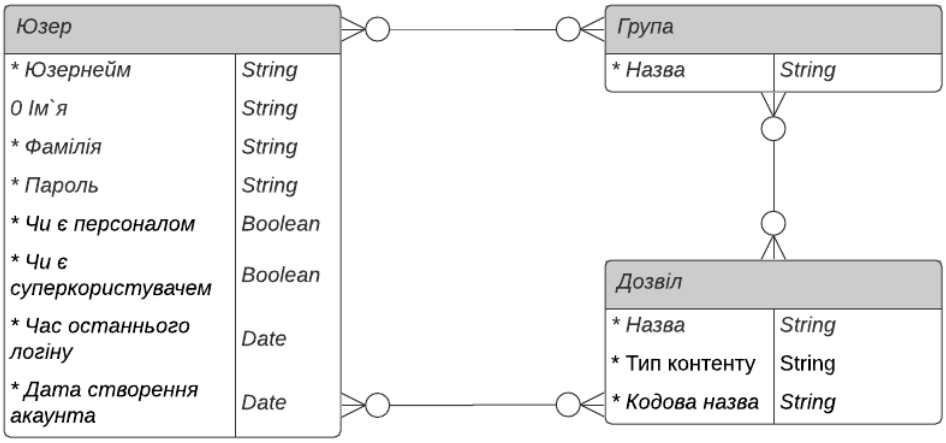


Рисунок 13 ER-модель модуля "Користувачі"

* - обов`язковий атрибут;

0 - необов`язковий атрибут;

Для реалізації я буду використовувати модуль фреймворку Django “**User authentication**”, що містить моделі “Юзери”, “Групи”.

Модель “Юзери”

Юзери є ядром системи аутентифікації. Вони, як правило, представляють людей, які взаємодіють з сайтом, і використовуються для увімкнення таких речей, як обмеження доступу, реєстрація профілів користувачів, асоціювання вмісту з творцями і т. д. У системі автентифікації Django існує лише один клас користувачів, тобто "суперкористувачі" або "персонал адміністратора" - це просто користувачі із встановленими спеціальними атрибутами, а не різні класи користувачів.

Якщо користувач одночасно є і персоналом і суперкористувачем, то він вважається адміністратором. Адміністратор має найвищі права в системі[18].

Атрибути:

- Юзернейм
- Ім`я (необов`язково)
- Фамілія (необов`язково)
- Пароль

- **Групи**
- Чи є персоналом
- Чи є суперкористувачем
- Дозволи
- Час останнього логіну
- Дата створення акаунта

Модель “Групи”

Групи - це загальний спосіб класифікації користувачів, щоб ви могли застосувати до них дозволи чи іншу мітку. Користувач може належати до будь-якої кількості груп.

Користувач у групі автоматично отримує дозволи, надані цій групі. Наприклад, якщо редактор сайту групи має дозвіл **can_edit_smth**, будь-який користувач у цій групі матиме такий дозвіл.

Крім дозволів, групи - це зручний спосіб класифікувати користувачів, щоб надати їм якусь мітку або розширену функціональність. Наприклад, ви можете створити групу "Спеціальні користувачі", і ви можете написати код, який може, скажімо, надати їм доступ до частини вебсайту, призначеного лише для учасників[19].

Атрибути:

- Назва
- Дозволи

3.6.3. Компонент “DCIM”

Компонент, що відповідає за інфраструктуру підприємства. Головним чином, серед його функцій – занесення даних про пристрої, які беруть участь в мережі: персональні комп`ютери, маршрутизатори, комутатори, роутери, тощо.

Стійки

Всі додані пристрої прив`язуються до стійок. Але це виконує роль не так фізичних стійок, як логічного групування пристроїв за їх функціональним призначенням.

Локації/Регіони

Кожна стійка має належати локації. Локації своєю чергою належать регіонам. Локація – це по суті і є офіс. Також можливий циклічний зв`язок між регіонами коли в батьківському регіоні існує декілька підрегіонів (напр. Україна -> Київ -> Поділ). Це потрібно для зручного управління всіма елементами мережі підприємства.

Спрощена ER-модель "DCIM/IPAM"

Це спрощена ER-модель, що показує зв`язки між сутностями (моделями) об`єднаних модулів. Атрибути сутностей спеціально опущені, задля легшого сприйняття схеми.

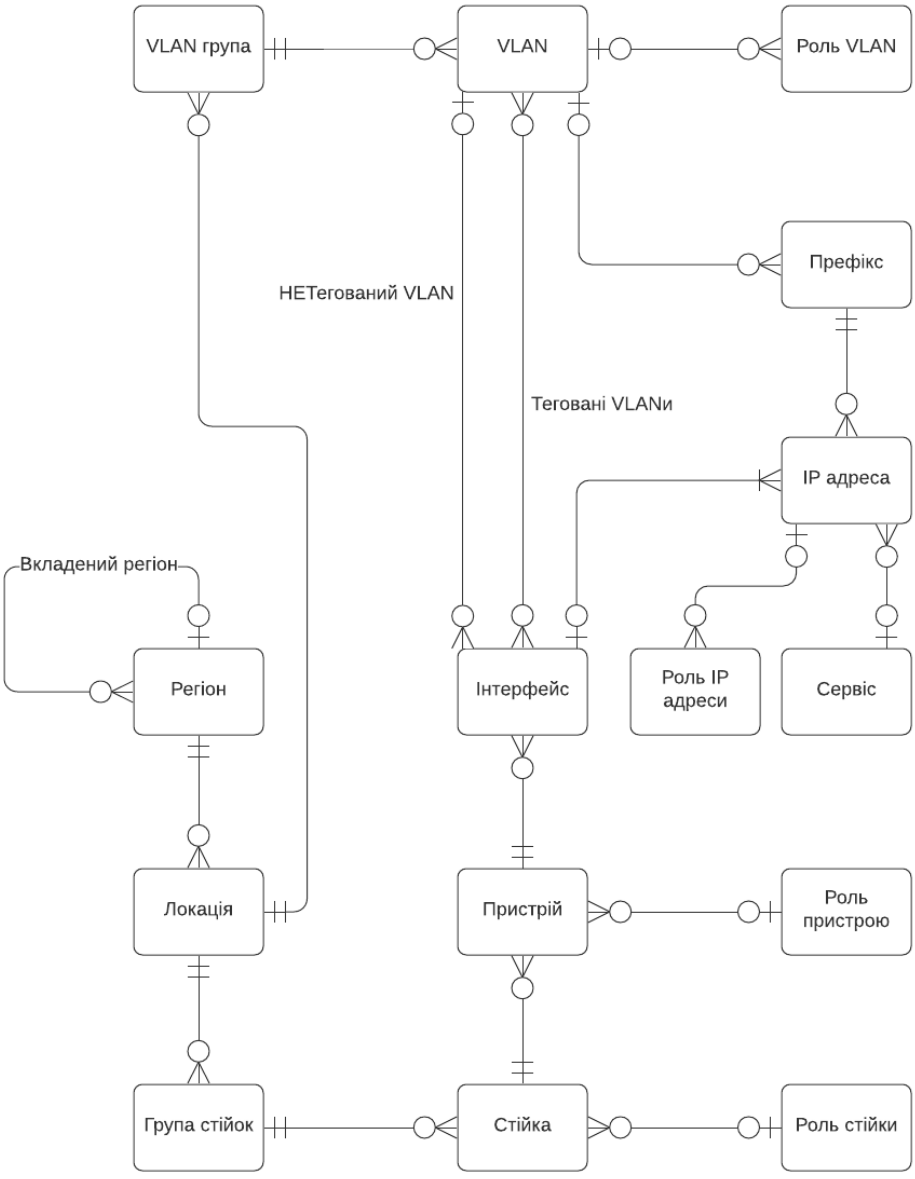


Рисунок 14 Спрощена ER-модель модулів DCIM/IPAM

Модуль “Пристрої”

Модель “Пристрій”

Пристрій являє собою фізичне обладнання, встановлене на стійці.

Атрибути:

- **Тип**
- **Роль** (необов`язково)
- **Платформа** (необов`язково)
- Назва
- Серійний номер
- **Стійка**
- Статус (offline, active)
- Мак-адреса (необов`язково)

Модель “Роль пристрою”

Пристрої поділені за функціональними ролями, наприклад, “Leaf Switch” або “DHCP Server”.

Атрибути:

- Назва
- Колір(вказується для наочного відображення пристроїв за ролями на стійках)
- Опис

Модель “Виробник”

Виробник представляє компанію, що виробляє пристрої. Наприклад **Cisco** або **Dell**.

Атрибути:

- Назва
- Опис

Модель “Тип пристрою”

Тип пристрою представляє виробника і модель.

Атрибути:

- **Виробник**
- Модель

Модель “Платформа”

Платформа відповідає за програмне забезпечення, або прошивку, що залучене на пристрої. Наприклад “**Cisco IOS-XR**”.

Атрибути:

- Назва
- **Виробник** (необов`язково)
- Опис

Модуль “Стійки”

Модель “Стійка”

Пристрої розміщено в стійках.

Атрибути:

- Назва
- **Група**
- Статус(reserved, available, active)
- **Роль** (необовязково)
- Серійний номер (необовязково)
- Тип стійки (2-post-frame, 4-post-frame, необов`язково)

Модель “Група стійок”

Стійки можна групувати як підмножини всередині локації. Сфера застосування групи варіюється в залежності від місця прив`язки.

Наприклад, якщо локація охоплює кампус, тоді група може позначати кожну окрему будівлю в межах цього кампусу. Якщо ж локація репрезентує будівлю, тоді група – окрему кімнату, або цілий поверх.

Атрибути:

- Назва
- **Локація**

- Опис

Модель “Роль стійки”

Стійкам можна присвоїти певну функціональну роль.

Атрибути:

- Назва
- Колір
- Опис

Модуль “Локації”

Модель “Локація”

Локація вказує географічне положення мережі; зазвичай це будівля або кампус.

Атрибути:

- Назва
- Статус (active, planned, staging)
- **Регіон**
- Часова зона (необовязково)
- Опис
- Фізична адреса (необовязково)
- Широта (необовязково)
- Довгота (необовязково)
- Контактний телефон (необовязково)
- Контактна пошта (необовязково)

Модель “Регіон”

Місця можуть групуватись за географічними регіонам.

Атрибути:

- **Батько** (необов`язково)
- Назва
- Опис

3.6.4. Компонент “IPAM”

Головний компонент застосунку, що відповідає за внесення даних про IP-простір мережі.

Префікс

Існує 3 типи префіксів: Контейнер, пул і звичайний.

Контейнер може мати безліч підмереж, які не перетинаються. Кожна підмережа також може бути контейнером. Тобто, можливо зробити безліч вкладених контейнерів підмереж.

По суті являє собою логічну групу підмереж.

- Контейнеру не можна призначати VLAN.
- Контейнер не обов`язково може мати локацію
- Контейнер не може бути пулом адрес
- До контейнера не можна призначати адреси
- Контейнерам заборонено вказувати локацію

Пул – це різновид префіксу, який симулює пул адрес для **NATy**(Network Address Translation), кожна адреса з якого дійсна і може бути назначена.

Якщо прапорець “пул” порожній, то вважається що префікс звичайний. У звичайного префікса відсутні перша та остання адреси: відповідно адреса самого префікса і **broadcast**(широкомовна) адреса.

IP

Є можливість створити нову адресу, або обрати наступну вільну в обраному префіксі. Префікси, є зручним способом групування адрес. Також їм можна призначати функціональні ролі. Кожна IP-адреса може бути призначена лише на один сервіс, але на один сервіс може бути призначено безліч адрес. IP-адреса має бути пов`язана з інтерфейсом саме того пристрою, на якому запущено даний сервіс.

VLAN

Ще одним способом групування адрес є створення віртуальних підмереж. Кожна підмережа, своєю чергою, може групуватись у групи VLANів. Кожному VLANу також можна призначити функціональну роль.

Service

Сервіси відповідають запущеним на пристроях застосуванням. Їм можна призначити IP-адреси, протокол(UDP/TCP) та порти, на яких воно функціонує.

Модуль "IP-адреси"

Модель "IP адреса"

Зображує IPv4 або IPv6 адресу та маску. Може бути призначена Інтерфейсу або сервісу. Інтерфейс/сервіс може мати нуль або декілька адрес призначених йому.

Атрибути:

- Адреса
- Статус (active, reservered, deprecated, dhcp)
- Роль (необов`язково)
- DNS назва (необов`язково)
- Опис
- Призначений інтерфейс (необов`язково)
- Призначений сервіс (необов`язково)

Модель "Префікс"

Відображає мережу IPv4 або IPv6, включаючи довжину маски.

Атрибути:

- Префікс
- Чи є контейнером (може мати підпрефікси)
- Чи є пулом (всі адреси з цього пулу вважаються доступними)
- **Контейнер батько** (необов`язково)
- **Локація** (необов`язково)
- **VLAN** (необов`язково)
- Статус (active, reserved, deprecated)
- Роль

- Опис

Модель “Роль адреси”

Показує функціональну роль префіксу або VLAN, наприклад, “Користувачі”, “Менеджмент”.

Атрибути:

- Назва
- Опис

Модуль “VLAN”

Модель “VLAN”

VLAN – домен другого рівня, що ідентифікується 12-бітним числом(1-4094). Кожен VLAN має належати місцю, але VLAN id необов`язково мають бути унікальними всередині місця. Також VLAN може бути призначений групі, і в межах цієї групи, VLANId має бути унікальним.

Як і в префіксах, кожен VLAN може мати роль. VLAN може мати нуль або декілька префіксів, призначених йому.

Атрибути:

- **Група**
- VID
- Назва
- Статус(active/reserved/deprecated)
- **Роль**
- Опис

Модель “Група VLANів”

Довільна колекція VLANів, де кожен VLAN id має бути унікальним.

Атрибути:

- Назва
- Опис
- **Локація**

Модель “Роль VLAN”

Відображає функціональну роль VLAN, наприклад, “Менеджмент”, “Сервери”.

Атрибути:

- Назва
- Опис

Модуль “Сервіси”

Модель “Сервіс”

Являє собою службу 4-го рівня(напр. HTTP або SSH), запущену на пристрої. Сервіс може бути прив`язаний до однієї або декількох IP-адрес, що належать пристрою.

Атрибути:

- Назва
- **Пристрій**
- Протокол
- Порти
- Опис

Модуль “Компоненти пристроїв”

Модель “Інтерфейс”

Мережевий інтерфейс пристрою.

Атрибути:

- Увімкнений
- **Пристрій**
- Назва
- Опис
- Тип(virtual, ethernet, wireless, serial)
- **Нетегований VLAN** (необов`язково)
- **Теговані VLAN** (необов`язково)

3.7. Детальна розробка компоненту “IPAM” на прикладі моделі “IPPrefix”

Для демонстрації роботи програми, я обрав модель “IPPrefix”. Код з лістингу (Додаток А) показує визначення моделі у фреймворку Django. Також вказано додатковий метод з атрибутом “@property”, що дає змогу дізнатись версію префікса(IPv4/IPv6).

Розглянемо метод моделі “IPPrefix”, що вертає процентне співвідношення вільних адрес у ньому. Вираховується діленням зайнятих адрес префікса на кількість всіх можливих адрес префікса. У випадку префікса-контейнера, сумуються всі поточні IP-адреси його нащадків:

```
def get_utilization(self):
    """
    Determine the utilization of the prefix and return it as a percentage.
    For container, calculate utilization based on subnet prefixes.
    For all others, count child IP addresses.
    """
    if self.is_container:
        child_count = calc_ipaddress_children(self.subnets.all())
        return round(float(child_count) / self.prefix.size * 100, 2)
    else:
        child_count = IPSet([ip.address.ip for ip in
self.ip_addresses.all()]).size
        prefix_size = self.prefix.size
        if self.prefix.version == 4 and self.prefix.prefixlen < 31 and not
self.is_pool:
            prefix_size -= 2
        return round(float(child_count) / prefix_size * 100, 2)
```

Допоміжна функція, що рекурсивно підраховує кількість зайнятих адрес у всіх нащадків:

```
def calc_ipaddress_children(subnets):
    result = 0
    for subnet in subnets:
        if subnet.is_container:
            result += calc_ipaddress_children(subnet.subnets.all())
        else:
            child_count = IPSet([ip.address.ip for ip in
subnet.ip_addresses.all()]).size
```

```
            if subnet.prefix.version == 4 and subnet.prefix.prefixlen < 31 and not
subnet.is_pool:
                child_count += 2
            result += child_count
    return result
```

Наступний метод, вертає всі вільні адреси префікса. Для префікса контейнера підраховується кількість вільних адрес його нащадків. Для префікса-пула, всі можливі адреси, а для звичайного – всі, окрім першої та останньої:

```
def get_available_ips(self):
    """
    Return all available IP addresses within this prefix as an IPSet.
    """
    prefix = IPSet(self.prefix)

    # If the prefix is container and has children,
    # count available children addresses
    if self.is_container and self.subnets:
        available_ips = IPSet([])
        for subnet in self.subnets.all():
            available_ips.update(subnet.get_available_ips())
        return available_ips

    prefix_ips = IPSet([ip.address.ip for ip in self.ip_addresses.all()])
    available_ips = prefix - prefix_ips

    # If the prefix is IPv6 protocol, is a pool or has prefix length 31
    # return all the addresses from the prefix
    if self.family == 6 or self.is_pool or self.prefix.prefixlen == 31:
        return available_ips

    # If the prefix is not pool, omit first and last addresses
    available_ips -= IPSet([
        netaddr.IPAddress(self.prefix.first),
        netaddr.IPAddress(self.prefix.last),
    ])

    return available_ips
```

3.8. Встановлення готового застосунку на віртуальній машині UbuntuVM

Для впровадження готового застосунку в мережу підприємства, його потрібно розгорнути. Для локального використання ідеально підійде віртуальна машина на основі linux, в цьому випадку UbuntuVM.

При запуску сервера використовувався конфігураційний файл Apache2 **ipam_project.conf** (Додаток Б).

Переконаємося у працездатності сервера. За допомогою утиліти curl відправимо на сервер GET-запит “http://192.168.0.120/api/dcim”. У відповідь отримали всі кінцеві точки API для компонента “DCIM”. Те саме можна побачити й через вебінтерфейс.

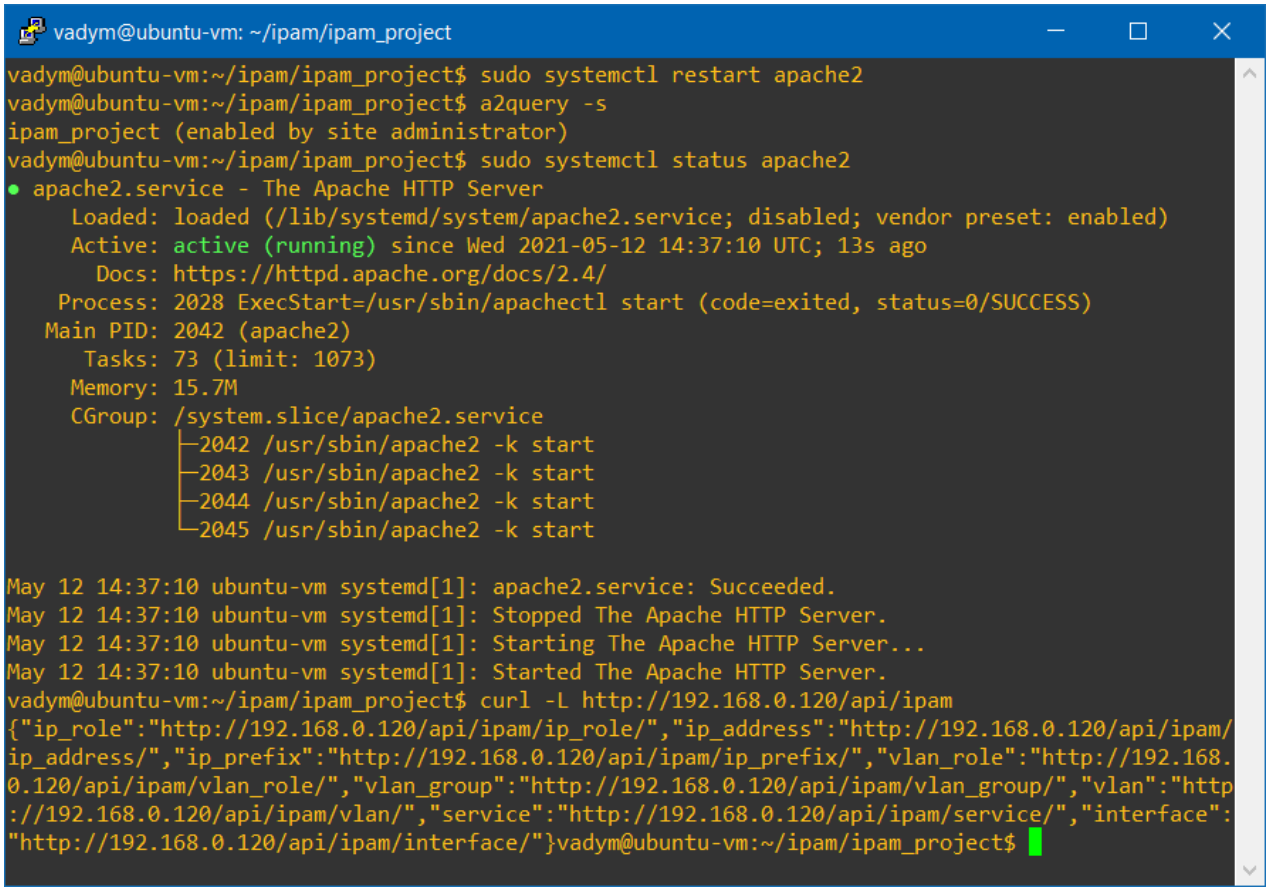


Рисунок 15 Сервер apache2 на ubuntuVM з встановленим IPAM

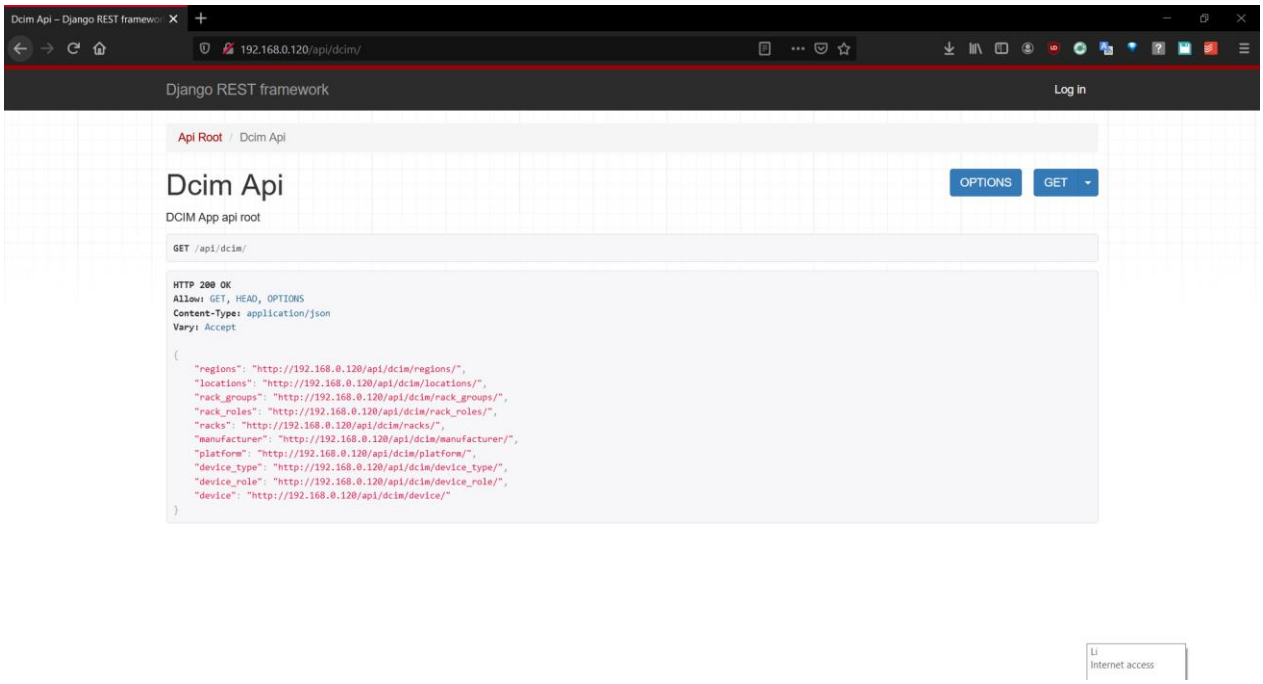


Рисунок 16 Сторінка документованого API компоненту DCIM

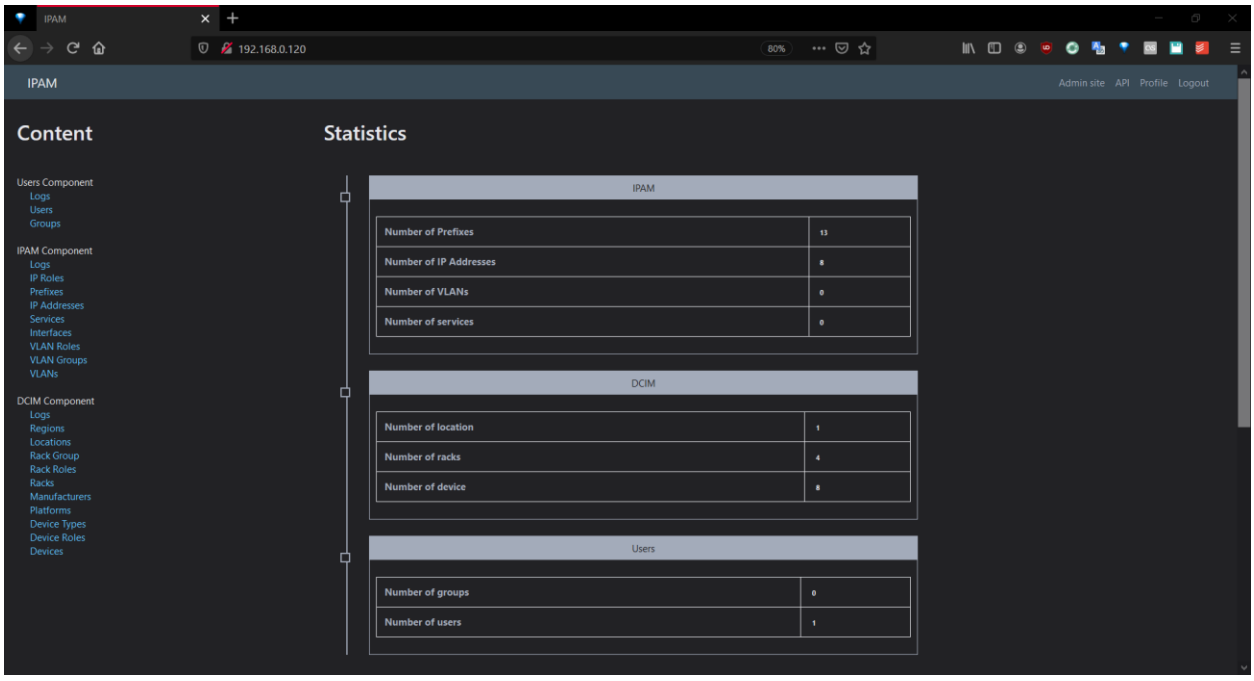


Рисунок 17 Головна сторінка застосунку

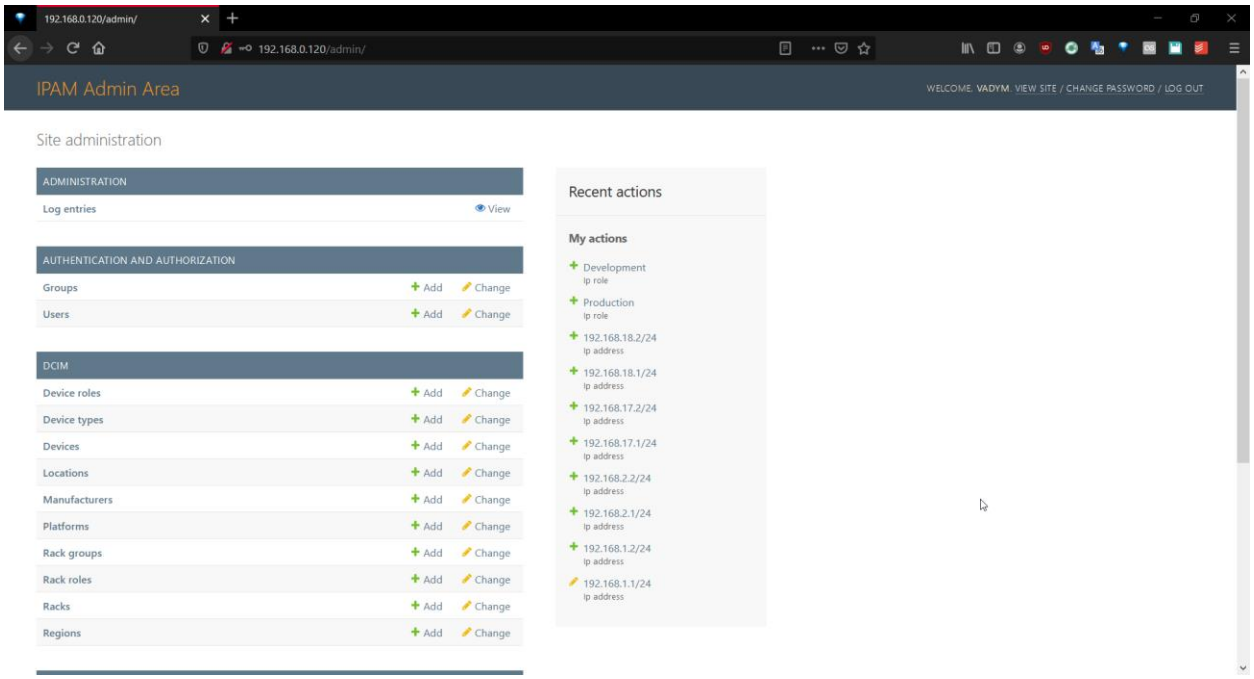


Рисунок 18 Головна сторінка вебінтерфейсу адміністратор

3.9. Тестування застосунку

Для тестування власного рішення, було побудовано невелику мережу підприємства і на її основі продемонстровано наповненість IPAM системи.

Для побудови топології мережі використано програму **Packet Tracer**:

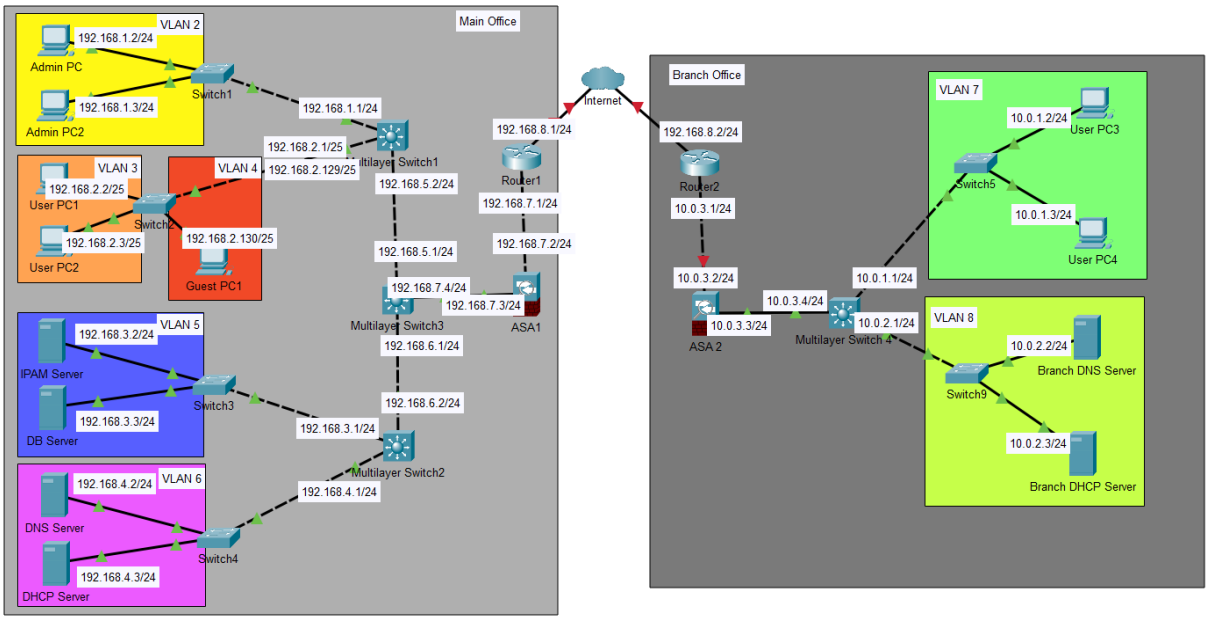


Рисунок 19 Топологія тестової мережі підприємства, що складається з двох локацій

Мережа підприємства складається з локальних мереж двох офісів: головного і філіалу. На головний офіс виділено підмережу 192.168.0.0/16, а на

філіал - 10.0.0.0/16. Кожна локальна мережа має декілька VLANів, позначених прямокутниками різних кольорів. Зі сторони кожного пристрою, звідки виходить **ethernet** кабель, я позначив IP-адресу інтерфейсу(або IP-адресу віртуального інтерфейсу, у випадку з **L-3** комутатора).

Перейдемо до розгляду вебінтерфейсу.

Спершу розглянемо інтерфейс додавання IP-адрес:

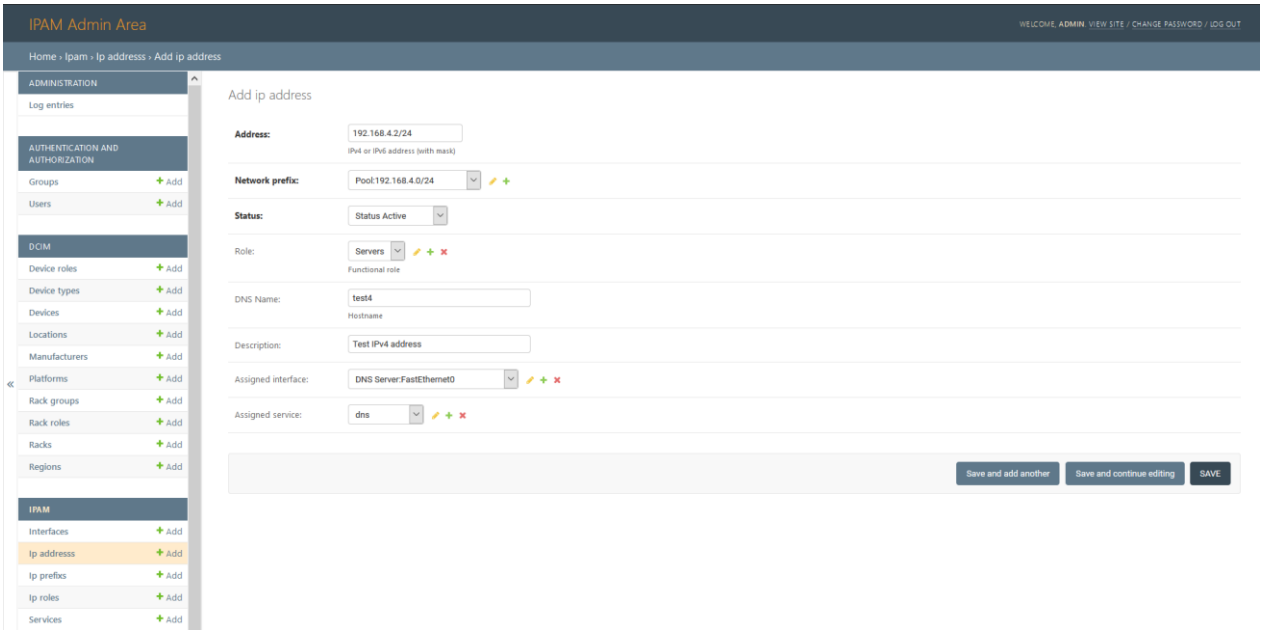


Рисунок 20 Інтерфейс додавання IPv4-адреси

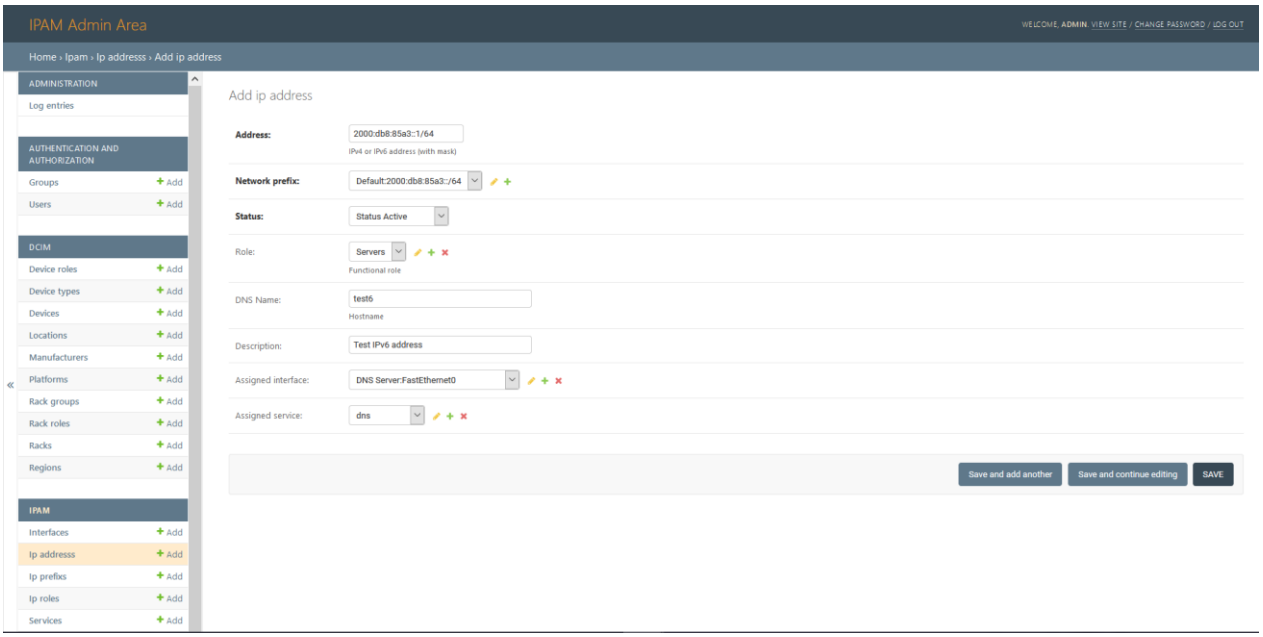


Рисунок 21 Інтерфейс додавання IPv6-адреси

На головній сторінці можна побачити статистику мережі підприємства:

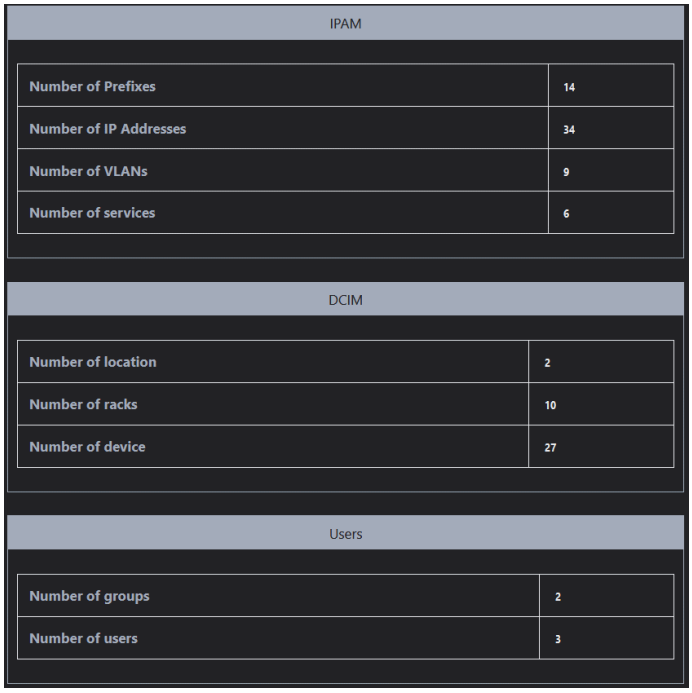


Рисунок 22 Статистика мережі підприємства

Статистика доступна також окремо по кожній локацій(наприклад “Головного офісу”):

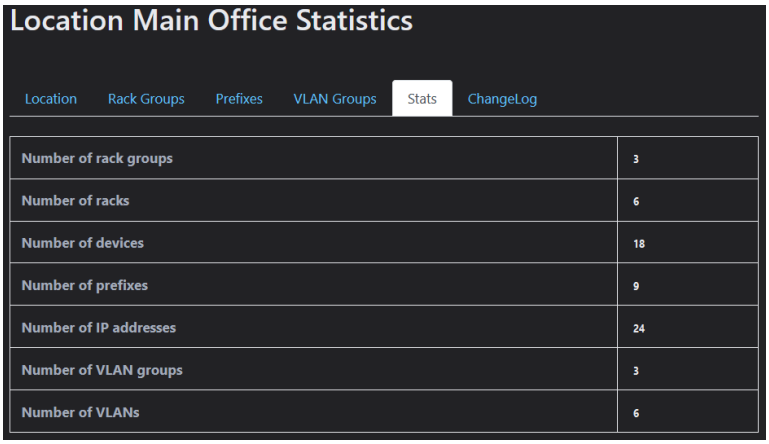
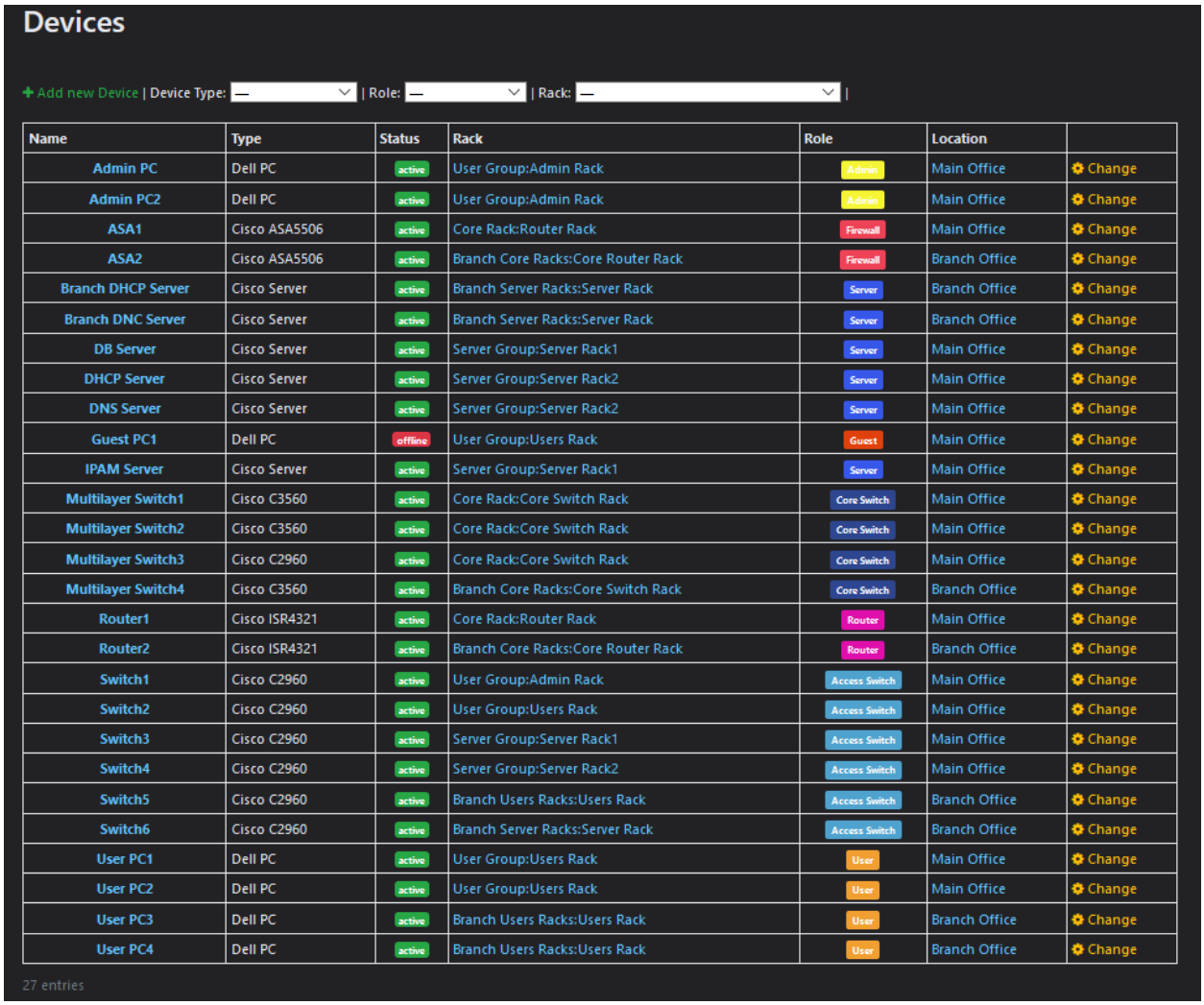


Рисунок 23 Статистика по локації "Головний офіс"

Тепер перейдемо до розгляду окремих компонентів та їх модулів. Переважна частина даних представлена у вигляді таблиць, з посиланнями на інші таблиці, можливістю додати новий об`єкт, або змінити обраний:



Devices

+ Add new Device | Device Type: — | Role: — | Rack: — |

Name	Type	Status	Rack	Role	Location	
Admin PC	Dell PC	active	User Group:Admin Rack	Admin	Main Office	Change
Admin PC2	Dell PC	active	User Group:Admin Rack	Admin	Main Office	Change
ASA1	Cisco ASA5506	active	Core Rack:Router Rack	Firewall	Main Office	Change
ASA2	Cisco ASA5506	active	Branch Core Racks:Core Router Rack	Firewall	Branch Office	Change
Branch DHCP Server	Cisco Server	active	Branch Server Racks:Server Rack	Server	Branch Office	Change
Branch DNC Server	Cisco Server	active	Branch Server Racks:Server Rack	Server	Branch Office	Change
DB Server	Cisco Server	active	Server Group:Server Rack1	Server	Main Office	Change
DHCP Server	Cisco Server	active	Server Group:Server Rack2	Server	Main Office	Change
DNS Server	Cisco Server	active	Server Group:Server Rack2	Server	Main Office	Change
Guest PC1	Dell PC	offline	User Group:Users Rack	Guest	Main Office	Change
IPAM Server	Cisco Server	active	Server Group:Server Rack1	Server	Main Office	Change
Multilayer Switch1	Cisco C3560	active	Core Rack:Core Switch Rack	Core Switch	Main Office	Change
Multilayer Switch2	Cisco C3560	active	Core Rack:Core Switch Rack	Core Switch	Main Office	Change
Multilayer Switch3	Cisco C2960	active	Core Rack:Core Switch Rack	Core Switch	Main Office	Change
Multilayer Switch4	Cisco C3560	active	Branch Core Racks:Core Switch Rack	Core Switch	Branch Office	Change
Router1	Cisco ISR4321	active	Core Rack:Router Rack	Router	Main Office	Change
Router2	Cisco ISR4321	active	Branch Core Racks:Core Router Rack	Router	Branch Office	Change
Switch1	Cisco C2960	active	User Group:Admin Rack	Access Switch	Main Office	Change
Switch2	Cisco C2960	active	User Group:Users Rack	Access Switch	Main Office	Change
Switch3	Cisco C2960	active	Server Group:Server Rack1	Access Switch	Main Office	Change
Switch4	Cisco C2960	active	Server Group:Server Rack2	Access Switch	Main Office	Change
Switch5	Cisco C2960	active	Branch Users Racks:Users Rack	Access Switch	Branch Office	Change
Switch6	Cisco C2960	active	Branch Server Racks:Server Rack	Access Switch	Branch Office	Change
User PC1	Dell PC	active	User Group:Users Rack	User	Main Office	Change
User PC2	Dell PC	active	User Group:Users Rack	User	Main Office	Change
User PC3	Dell PC	active	Branch Users Racks:Users Rack	User	Branch Office	Change
User PC4	Dell PC	active	Branch Users Racks:Users Rack	User	Branch Office	Change

27 entries

Рисунок 24 Табличка з пристроями

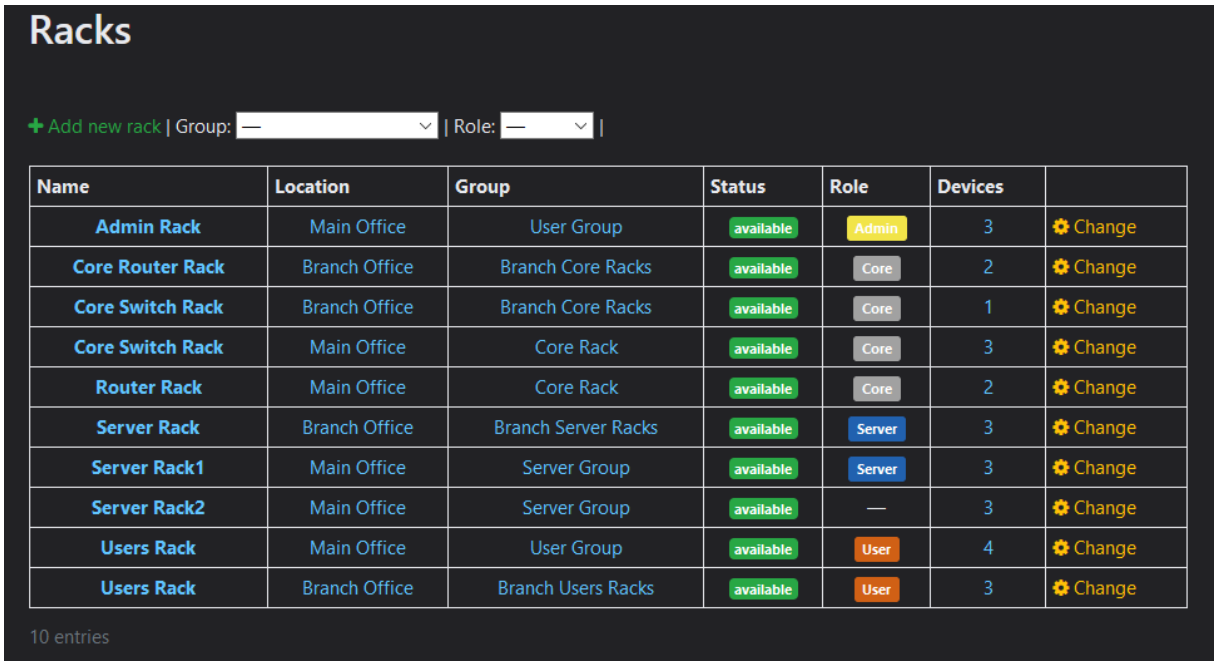
Device Roles

+ Add new Device role

Name	Devices	Color	Description	
Access Switch	6	Access Switch		Change
Admin	2	Admin		Change
Core Switch	4	Core Switch		Change
Firewall	2	Firewall		Change
Guest	1	Guest		Change
Router	2	Router		Change
Server	6	Server		Change
User	4	User		Change

8 entries

Рисунок 25 Табличка з ролями пристроїв



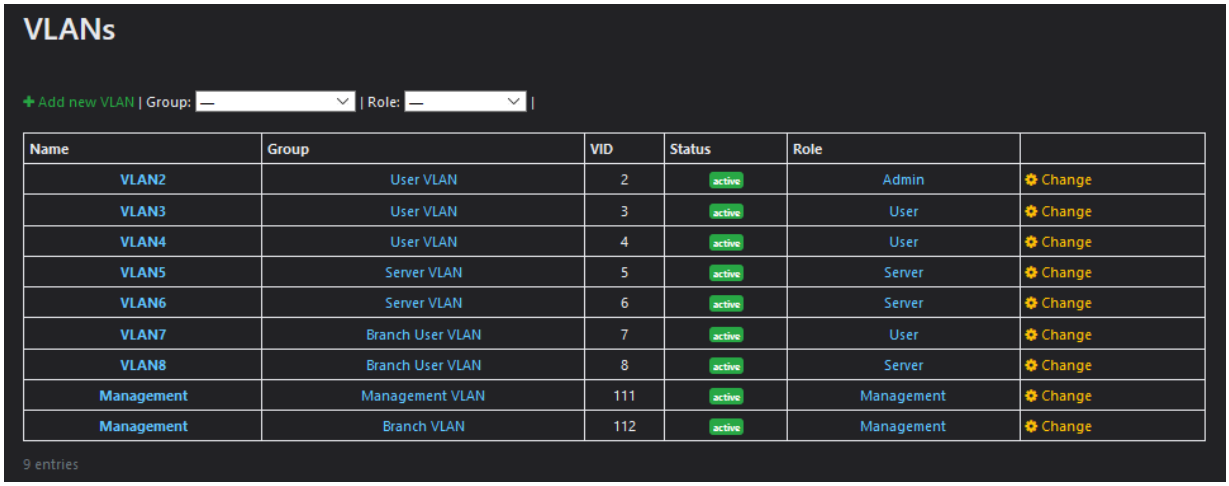
Racks

+ Add new rack | Group: — | Role: — |

Name	Location	Group	Status	Role	Devices	
Admin Rack	Main Office	User Group	available	Admin	3	Change
Core Router Rack	Branch Office	Branch Core Racks	available	Core	2	Change
Core Switch Rack	Branch Office	Branch Core Racks	available	Core	1	Change
Core Switch Rack	Main Office	Core Rack	available	Core	3	Change
Router Rack	Main Office	Core Rack	available	Core	2	Change
Server Rack	Branch Office	Branch Server Racks	available	Server	3	Change
Server Rack1	Main Office	Server Group	available	Server	3	Change
Server Rack2	Main Office	Server Group	available	—	3	Change
Users Rack	Main Office	User Group	available	User	4	Change
Users Rack	Branch Office	Branch Users Racks	available	User	3	Change

10 entries

Рисунок 26 Табличка зі стійками



VLANs

+ Add new VLAN | Group: — | Role: — |

Name	Group	VID	Status	Role	
VLAN2	User VLAN	2	active	Admin	Change
VLAN3	User VLAN	3	active	User	Change
VLAN4	User VLAN	4	active	User	Change
VLAN5	Server VLAN	5	active	Server	Change
VLAN6	Server VLAN	6	active	Server	Change
VLAN7	Branch User VLAN	7	active	User	Change
VLAN8	Branch User VLAN	8	active	Server	Change
Management	Management VLAN	111	active	Management	Change
Management	Branch VLAN	112	active	Management	Change

9 entries

Рисунок 27 Табличка з VLAN

Is Enabled	Name	Type	Device	MacAddress	
✓	FastEthernet0	ethernet	Admin PC2	00:E0:B0:C8:2E:A2	Change
✓	FastEthernet0	ethernet	User PC3	00:01:C9:C2:43:D5	Change
✓	FastEthernet0	ethernet	Admin PC	00:05:5E:50:89:14	Change
✓	FastEthernet0	ethernet	DNS Server	00:60:47:62:20:13	Change
✓	FastEthernet0	ethernet	DHCP Server	00:09:7C:7A:3C:4D	Change
✓	FastEthernet0	ethernet	User PC4	00:E0:F9:86:0E:06	Change
✓	FastEthernet0	ethernet	User PC1	00:D0:58:EB:C9:4C	Change
✓	FastEthernet0	ethernet	User PC2	00:0B:BE:A3:DC:79	Change
✓	FastEthernet0	ethernet	Guest PC1	00:01:63:A2:B7:67	Change
✓	FastEthernet0	ethernet	Branch DHCP Server	00:0C:85:1C:76:B5	Change
✓	FastEthernet0	ethernet	Branch DNC Server	00:02:4A:0E:18:77	Change
✓	FastEthernet0	ethernet	IPAM Server	00:02:17:C8:36:A6	Change
✓	FastEthernet0	ethernet	DB Server	00:00:0C:4B:D6:70	Change
✓	FastEthernet0/1	ethernet	Multilayer Switch1	00:E0:B0:89:88:01	Change
✓	FastEthernet0/1	ethernet	Multilayer Switch4	00:01:43:06:50:01	Change
✓	FastEthernet0/1	ethernet	Multilayer Switch3	00:01:64:66:AA:01	Change
✓	FastEthernet0/1	ethernet	Multilayer Switch2	00:0A:F3:CC:41:01	Change
✓	FastEthernet0/2	ethernet	Multilayer Switch3	00:01:64:66:AA:02	Change
✓	FastEthernet0/2	ethernet	Multilayer Switch2	00:0A:F3:CC:41:02	Change
✓	FastEthernet0/2	ethernet	Multilayer Switch4	00:01:43:06:50:02	Change
✓	FastEthernet0/2.1	virtual	Multilayer Switch1	00:E0:B0:89:88:02	Change
✓	FastEthernet0/2.2	virtual	Multilayer Switch1	00:E0:B0:89:88:03	Change
✓	FastEthernet0/3	ethernet	Multilayer Switch2	00:0A:F3:CC:41:03	Change
✓	FastEthernet0/3	ethernet	Multilayer Switch1	00:E0:B0:89:88:04	Change
✓	FastEthernet0/3	ethernet	Multilayer Switch4	00:01:43:06:50:03	Change
✓	FastEthernet0/3	ethernet	Multilayer Switch3	00:01:64:66:AA:03	Change
✓	GigabitEthernet0/0	ethernet	Router2	00:D0:97:1A:05:01	Change
✓	GigabitEthernet0/0/0	ethernet	Router1	00:0A:41:EB:9C:01	Change
✓	GigabitEthernet0/0/1	ethernet	Router1	00:0A:41:EB:9C:02	Change
✓	GigabitEthernet0/1	ethernet	Router2	00:D0:97:1A:05:02	Change
✓	GigabitEthernet1/1	ethernet	ASA1	00:E0:B0:23:89:01	Change
✓	GigabitEthernet1/1	ethernet	ASA2	00:09:7C:6D:8D:01	Change
✓	GigabitEthernet1/2	ethernet	ASA2	00:09:7C:6D:8D:02	Change
✓	GigabitEthernet1/2	ethernet	ASA1	00:E0:B0:23:89:02	Change

34 entries

Рисунок 28 Табличка з інтерфейсами пристроїв

Prefix list

Add new prefix | Role: — |

Type	Prefix	Status	Utilization	Location	VLAN	Role	
🗁	10.0.0.0/16	active	0.02%	—	—	—	Change
	10.0.1.0/24	active	1.18%	Branch Office	Branch User VLAN:VLAN7	Users	Change
	10.0.2.0/24	active	1.18%	Branch Office	Branch User VLAN:VLAN8	Servers	Change
	10.0.3.0/24	active	1.57%	Branch Office	—	Core	Change
🗁	192.168.0.0/16	active	0.04%	—	—	—	Change
	192.168.1.0/24	active	1.17%	Main Office	User VLAN:VLAN2	Admin	Change
	192.168.2.0/25	active	2.34%	Main Office	User VLAN:VLAN3	Users	Change
	192.168.2.128/25	active	1.56%	Main Office	User VLAN:VLAN4	Guests	Change
	192.168.3.0/24	active	1.17%	Main Office	Server VLAN:VLAN5	Servers	Change
	192.168.4.0/24	active	1.17%	Main Office	Server VLAN:VLAN6	Servers	Change
	192.168.5.0/24	active	0.78%	Main Office	—	Core	Change
	192.168.6.0/24	active	0.78%	Main Office	—	Core	Change
	192.168.7.0/24	active	1.56%	Main Office	—	Core	Change
	192.168.8.0/24	active	0.78%	Main Office	—	VPN	Change

14 entries

Рисунок 29 Табличка з префіксами

IP Address	Status	Role	Device	
10.0.1.1/24	active	Users	Multilayer Switch4	Change
10.0.1.2/24	active	Users	User PC3	Change
10.0.1.3/24	active	Users	User PC4	Change
10.0.2.1/24	active	Servers	Multilayer Switch4	Change
10.0.2.2/24	active	Servers	Branch DNC Server	Change
10.0.2.3/24	active	Servers	Branch DHCP Server	Change
10.0.3.1/24	active	Core	Router2	Change
10.0.3.2/24	active	Core	ASA2	Change
10.0.3.3/24	active	Core	ASA2	Change
10.0.3.4/24	active	Core	Multilayer Switch4	Change
192.168.1.1/24	active	Admin	Multilayer Switch1	Change
192.168.1.2/24	active	Admin	Admin PC	Change
192.168.1.3/24	active	Admin	Admin PC2	Change
192.168.2.1/25	active	Users	Multilayer Switch1	Change
192.168.2.2/25	active	Users	User PC1	Change
192.168.2.3/25	active	Users	User PC2	Change
192.168.2.129/25	active	Guests	Multilayer Switch1	Change
192.168.2.130/25	active	Guests	Guest PC1	Change
192.168.3.1/24	active	—	Multilayer Switch2	Change
192.168.3.2/24	active	Servers	IPAM Server	Change
192.168.3.3/24	active	Servers	DB Server	Change
192.168.4.1/24	active	Servers	Multilayer Switch2	Change
192.168.4.2/24	active	Servers	DNS Server	Change
192.168.4.3/24	active	—	DHCP Server	Change
192.168.5.1/24	active	Core	Multilayer Switch3	Change
192.168.5.2/24	active	Core	Multilayer Switch1	Change
192.168.6.1/24	active	Core	Multilayer Switch3	Change
192.168.6.2/24	active	Core	Multilayer Switch2	Change
192.168.7.1/24	active	—	Router1	Change
192.168.7.2/24	active	Core	ASA1	Change
192.168.7.3/24	active	Core	ASA1	Change
192.168.7.4/24	active	Core	Multilayer Switch3	Change
192.168.8.1/24	active	VPN	Router1	Change
192.168.8.2/24	active	VPN	Router2	Change

34 entries

Рисунок 30 Табличка з IP-адресами

Services

+ Add new service

Name	Device	Protocol	Ports	
branch dhcp	Branch DHCP Server	tcp, udp	67	Change
branch dns	Branch DNC Server	tcp, udp	53	Change
dhcp	DHCP Server	tcp, udp	67	Change
dns	DNS Server	tcp, udp	53	Change
IPAM	IPAM Server	tcp	80	Change
PostgreSQL	DB Server	tcp	4321	Change

6 entries

Рисунок 31 Табличка із запущеними сервісами

Висновок

В ході роботи було проведено дослідження систем керування адресним простором мережі. Було розглянуто проблеми та актуальність таких систем в сучасних підприємствах та здійснено порівняльний аналіз готових рішень на ринку. На основі цих досліджень було спроєктовано і розроблено власний прототип IPAM-системи. Складовими частинами повноцінної системи стали IPAM-застосунок, сервер бази даних, DNS-сервер та DHCP-сервер. Було проведено детальну розробку одного з компонентів такої системи, а саме — IPAM-застосунку.

В результаті було розроблено рішення мовою програмування Python з використанням фреймворку Django для інвентаризації та зручного розподілу адресного простору IP-мережі. Серед основних компонентів розробки: “Користувачі”, “DCIM”, “IPAM”. Серед підтримуваних протоколів і розроблених функціональних особливостей в застосуванні:

- Підтримка IPv4 і IPv6 адрес;
- Інформація про підмережі та задіяні/вільні адреси;
- Інформація про віртуальні локальні мережі – VLANи;
- Інформація про пристрої та їх мережеві інтерфейси, з призначенням MAC-адрес;
- Ім`я IP-адрес хостів в DNS.

Можливі варіанти розширення функціоналу застосування:

- Повноцінна інтеграція зі службами DHCP і DNS;
- Система відправлення інформаційних повідомлень;
- Функція сканування мережі;
- Виведення інформації про DHCP/Static конфігурацію IP-адреси на пристроях.

Було зроблено висновок, що підхід до організації IPAM-систем з єдиним джерелом істини гарно підходить для підприємств.

Список використаної літератури

[1] – Https://habr.com/ru/ // Habr: [Веб-сайт]. 2019. URL: https://habr.com/ru/post/453516/ (дата звернення: 17.05.2021).

[2] – Https://www.efficientip.com/ // Efficient IP: [Веб-сайт]. URL: https://www.efficientip.com/what-is-ddi-dns-dhcp-ipam/ (дата звернення: 17.05.2021).

[3] – Https://en.wikipedia.org/wiki/ // Wikipedia: [Веб-сайт]. 2021. URL: https://en.wikipedia.org/wiki/IP_address_management (дата звернення: 17.05.2021).

[4] – Https://netbox.readthedocs.io/en/stable/ // NetBox Documentation: [Веб-сайт]. URL: https://netbox.readthedocs.io/en/stable/#design-philosophy (дата звернення: 17.05.2021).

[5] – Https://habr.com/ru/ // Habr: [Веб-сайт]. 2020. URL: https://habr.com/ru/post/486000/ (дата звернення: 17.05.2021).

[6] – Https://www.ansible.com/blog // Red Hat Ansible: [Веб-сайт]. 2020. URL: https://www.ansible.com/blog/using-netbox-for-ansible-source-of-truth (дата звернення: 17.05.2021).

[7] – Https://developer.cisco.com/netdevops/live/ // Cisco devnet: [Веб-сайт]. 2020. URL: https://pubhub.devnetcloud.com/media/netdevops-live/site/files/s03t06.pdf (дата звернення: 17.05.2021).

[8] – Https://github.com/ // Github: [Веб-сайт]. URL: https://github.com/phpipam/phpipam (дата звернення: 17.05.2021).

[9] – Https://pandorafms.com/blog/ // Pandorafms: [Веб-сайт]. 2018. URL: https://pandorafms.com/blog/php-ipam/ (дата звернення: 17.05.2021).

[10] – Https://phpipam.net/ // Phpipam: [Веб-сайт]. 2013. URL: https://phpipam.net/news/automatic-host-availability-check/ (дата звернення: 17.05.2021).

[11] – Https://www.liquidweb.com/ // Liquidweb: [Веб-сайт]. 2018. URL: https://www.liquidweb.com/kb/what-is-power-dns/ (дата звернення: 17.05.2021).

[12] – Http://www.techspacekh.com/ // Tech space kh: [Веб-сайт]. 2017. URL: http://www.techspacekh.com/integrating-phpipam-authentication-with-ldapactive-directory-ad/ (дата звернення: 17.05.2021).

[13] – Solarwinds // Solarwinds Documantation: Administrator Guide "IP AddressManager". 2021. URL: https://documentation.solarwinds.com/archive/pdf/ipam/orionipamadministratorguide.pdf (дата звернення: 17.05.2021).

[14] – Https://netbox.readthedocs.io/en/stable/ // NetBox Documentation: [Веб-сайт]. URL: https://netbox.readthedocs.io/en/stable/#serve-as-a-source-of-truth (дата звернення: 17.05.2021).

[15] – Https://pypi.org/ // Python Package Index: [Веб-сайт]. 2020. URL: https://pypi.org/project/netbox-ddns/ (дата звернення: 17.05.2021).

[16] – Sys4: [Веб-сайт]. 2021. URL: https://blog.sys4.de/netbox-kea-dhcp-en.html (дата звернення: 17.05.2021).

[17] – Https://packetlife.net/blog/ // PacketLife: [Веб-сайт]. 2015. URL: https://packetlife.net/blog/2015/aug/10/writing-custom-ipam-application/ (дата звернення: 17.05.2021).

[18] – Https://docs.djangoproject.com/en/3.2/topics/auth/default/ // Django Documentation: [Веб-сайт]. URL: https://docs.djangoproject.com/en/3.2/topics/auth/default/#user-objects (дата звернення: 17.05.2021).

[19] – Https://docs.djangoproject.com/en/3.2/topics/auth/default/ // Django Documentation: [Веб-сайт]. URL: https://docs.djangoproject.com/en/3.2/topics/auth/default/#groups (дата звернення: 17.05.2021).

Додаток А
(обов`язковий)

Текст моделі “IPPrefix”

```
import netaddr
from django.db import models
from ipam.choices import IPPrefixStatusChoices
from ipam.fields import IPAddressField, IPNetworkField
from ipam.models.vlan import VLAN
from dcim.models.locations import Location
from netaddr import IPNetwork, IPSet
from ipam.utils import toset, AttributeGenerator
from ipam.utils import calc_ipaddress_children

class IPPrefix(models.Model, AttributeGenerator):
    prefix = IPNetworkField(
        help_text='IPv4 or IPv6 network with mask'
    )
    is_container = models.BooleanField(
        verbose_name='Is a container',
        default=False,
        help_text='Other IP prefixes can be nested inside this prefix'
    )
    is_pool = models.BooleanField(
        verbose_name='Is a pool',
        default=True,
        help_text='All IP addresses within this prefix are considered usable'
    )
    prefix_container = models.ForeignKey(
        to='self',
        on_delete=models.CASCADE,
        blank=True,
        null=True,
        limit_choices_to={'is_container': True},
        help_text='Choose prefix container if this prefix is nested',
        related_name='subnets'
    )
    location = models.ForeignKey(
        Location,
        related_name='ip_prefixes',
        on_delete=models.PROTECT,
        blank=True,
        null=True,
    )
    vlan = models.ForeignKey(
        VLAN,
        related_name='ip_prefixes',
        on_delete=models.PROTECT,
        blank=True,
        null=True,
    )
    status = models.CharField(
        max_length=50,
        choices=IPPrefixStatusChoices.choices,
        default=IPPrefixStatusChoices.STATUS_ACTIVE
    )
    role = models.ForeignKey(
        IPRole,
        related_name='ip_prefixes',
        on_delete=models.PROTECT,
        blank=True,
```

```
        null=True,
        help_text='Functional role'
    )
    description = models.CharField(
        max_length=200,
        blank=True
    )

    @property
    def family(self):
        if self.prefix:
            return self.prefix.version
        return None

    def get_utilization(self):
        """
        Determine the utilization of the prefix and return it as a percentage.
        For container, calculate utilization based on subnet prefixes.
        For all others, count child IP addresses.
        """
        if self.is_container:
            child_count = calc_ipaddress_children(self.subnets.all())
            return round(float(child_count) / self.prefix.size * 100, 2)
        else:
            child_count = IPSet([ip.address.ip for ip in
self.ip_addresses.all()]).size
            prefix_size = self.prefix.size
            if self.prefix.version == 4 and self.prefix.prefixlen < 31 and not
self.is_pool:
                prefix_size -= 2
            return round(float(child_count) / prefix_size * 100, 2)

    def get_available_ips(self):
        """
        Return all available IP addresses within this prefix as an IPSet.
        """
        prefix = IPSet(self.prefix)

        # If the prefix is container and has children,
        # count available children addresses
        if self.is_container and self.subnets:
            available_ips = IPSet([])
            for subnet in self.subnets.all():
                available_ips.update(subnet.get_available_ips())
            return available_ips

        prefix_ips = IPSet([ip.address.ip for ip in self.ip_addresses.all()])
        available_ips = prefix - prefix_ips

        # If the prefix is IPv6 protocol, is a pool or has prefix length 31
        # return all the addresses from the prefix
        if self.family == 6 or self.is_pool or self.prefix.prefixlen == 31:
            return available_ips

        # If the prefix is not pool, omit first and last addresses
        available_ips -= IPSet([
            netaddr.IPAddress(self.prefix.first),
            netaddr.IPAddress(self.prefix.last),
        ])

        return available_ips
```

Додаток Б

(обов`язковий)

Конфігураційний файл Apache2 сервера

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port th
at
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        ServerName 192.168.0.120

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
```

```
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf

        Alias /static /home/vadym/ipam/ipam_project/static
        <Directory /home/vadym/ipam/ipam_project/static>
          Require all granted
        </Directory>

        <Directory /home/vadym/ipam/ipam_project/ipam_project>
          <Files wsgi.py>
            Require all granted
          </Files>
        </Directory>

        WSGIScriptAlias / /home/vadym/ipam/ipam_project/ipam_project/wsgi.py
        WSGIDaemonProcess ipam_project python-
path=/home/vadym/ipam/ipam_project
python-home=/home/vadym/ipam/ipam_project/venv
        WSGIProcessGroup ipam_project


    </VirtualHost>
```