Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики

Факультет інформаційних технологій

Кафедра Інформаційних технологій та комп'ютерної інженерії

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційна робота ступеня бакалавра

Студента Коржова Марка Сергійовича

академічної групи ФІТ

спеціальності 126 Інформаційні системи та технології

за освітньо-професійною програмою Інформаційні системи та технології

на тему Програмний модуль автоматизованого налаштування конфігурації мережевого обладнання MikroTik

Керівник	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доцент Каштан В.Ю.			
розділів:				

Рецензент				

Нормоконтролер	проф. Коротенко Г.М.			

Дніпро
2024

ЗАТВЕРДЖЕНО:
завідувач кафедри

Інформаційні технології
та комп'ютерної інженерії

_____ _____
(підпис) (прізвище)

«___» _____ 20 ___ року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня ___бакалавра___

студенту Коржова Марка Сергійовича **академічної групи** ФІТ

спеціальності 126 Інформаційні системи та технології

за освітньо-професійною програмою Інформаційні системи та технології

на тему Програмний модуль автоматизованого налаштування конфігурації мережевого обладнання MikroTik

затверджену наказом ректора НТУ «Дніпровська політехніка» від 23.05.2024 № 469-с

Розділ	Зміст	Термін виконання
Розділ 1	Аналіз сучасного стану вирішення проблеми	11.03.2024
Розділ 2	Розробка програмного модулю автоматизованого налаштування конфігурації мережевого обладнання mikrotik	24.05.20024

Завдання видано _____ Каштан В.Ю.
(підпис керівника) (прізвище, ініціали)

Дата видачі « 05 » 06 2024 року

Дата поданні до екзаменаційної комісії 10.06.2024

Прийнято до виконання _____ Коржов М.С.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 92 стор., 16 рис., 2 додатка, 19 джерел.

Об’єкт розробки: програмний модуль для автоматизованого налаштування конфігурації мережевого обладнання MikroTik.

Мета роботи: розробка програмного модуля, що забезпечує автоматичне налаштування мережевого обладнання MikroTik. Модуль спрямований на спрощення та оптимізацію процесу налаштування, пропонуючи зручний інтерфейс для управління та розгортання конфігурацій на пристроях MikroTik.

У першому розділі проводиться огляд мережевого обладнання MikroTik, включаючи його історію розвитку, основні характеристики, популярність та використання у різних галузях. Розглядаються сучасні технології та методи налаштування мережевого обладнання, а також проблеми та виклики, з якими стикаються системні адміністратори.

У другому розділі наведена проектна складова програмного модуля для автоматизованого налаштування конфігурації мережевого обладнання MikroTik. Визначено основні компоненти модуля, їх функціональні характеристики та взаємодію між ними. Описані алгоритми підключення до пристрою, експорту, імпорту та редагування конфігурацій.

Розроблений програмний модуль можна використовувати в різних організаціях, які експлуатують мережеве обладнання MikroTik, для покращення процесу налаштування та управління конфігураціями мережевого обладнання.

АВТОМАТИЗАЦІЯ, КОНФІГУРАЦІЯ, МЕРЕЖЕВЕ ОБЛАДНАННЯ, MIKROTIK, ПРОГРАМНИЙ МОДУЛЬ, СИСТЕМНИЙ АДМІНІСТРАТОР, SSH, PYTHON, WINBOX

ABSTRACT

Explanatory Note: 92 pages, 16 figures, 2 appendices, 19 sources.

Object of Development: A software module for the automated configuration of MikroTik network equipment.

Purpose of the Work: To develop a software module that ensures automatic configuration of MikroTik network equipment. The module aims to simplify and optimize the configuration process, offering a convenient interface for managing and deploying configurations on MikroTik devices.

The first chapter provides an overview of MikroTik network equipment, including its development history, main characteristics, popularity, and use in various fields. It examines modern technologies and methods for configuring network equipment, as well as the problems and challenges faced by system administrators.

The second chapter presents the design component of the software module for the automated configuration of MikroTik network equipment. The main components of the module, their functional characteristics, and the interaction between them are defined. Algorithms for connecting to the device, exporting, importing, and editing configurations are described.

The developed software module can be used in various organizations that operate MikroTik network equipment to improve the process of configuring and managing network equipment configurations.

AUTOMATION, CONFIGURATION, NETWORK EQUIPMENT, MIKROTIK, SOFTWARE MODULE, SYSTEM ADMINISTRATOR, SSH, PYTHON, WINBOX

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

CLI (Command Line Interface) — це інтерфейс користувача, який дозволяє взаємодіяти з операційною системою або програмою через текстові команди. Замість використання графічного інтерфейсу, де користувачі натискають на кнопки та іконки, у CLI команди вводяться вручну з клавіатури.

Ansible — це інструмент автоматизації, який використовується для конфігураційного управління, розгортання програмного забезпечення та управління IT-інфраструктурою.

Puppet — це інструмент для автоматизації управління конфігурацією, який допомагає системним адміністраторам керувати інфраструктурою як кодом. Puppet дозволяє автоматизувати розгортання, управління і налаштування серверів і програмного забезпечення, забезпечуючи узгодженість та надійність інфраструктури.

Chef — це інструмент для автоматизації управління конфігурацією і розгортання програмного забезпечення, що допомагає системним адміністраторам і розробникам автоматизувати управління інфраструктурою. Chef дозволяє описувати інфраструктуру як код, використовуючи мову програмування Ruby.

MikroTik Cloud Hosted Router (CHR) — це версія операційної системи RouterOS від MikroTik, яка призначена для роботи на віртуальних машинах. CHR підтримує всі функції RouterOS, включаючи маршрутизацію, брандмауери, VPN, QoS, і т.д., і може бути запущена в різних віртуалізаційних середовищах, таких як VMware, Hyper-V, KVM, та інші.

The Dude Network Monitor — це інструмент для моніторингу мережі, розроблений компанією MikroTik. Він забезпечує адміністраторів потужними засобами для відстеження стану мережевих пристроїв та сервісів, допомагаючи швидко виявляти і вирішувати проблеми.

VPN (Virtual Private Network) — це технологія, яка створює захищене з'єднання через публічні мережі, такі як Інтернет. VPN дозволяє користувачам віддалено підключатися до приватних мереж, забезпечуючи безпеку і конфіденційність переданих даних.

QoS (Quality of Service) — це технологія, яка використовується для управління та пріоритизації мережевого трафіку. QoS допомагає забезпечити високу якість обслуговування для критично важливих додатків і служб, зменшуючи затримки, втрати пакетів і коливання (джиттер).

IoT (Internet of Things) — це концепція мережі фізичних пристроїв, транспортних засобів, побутових приладів та інших об'єктів, які оснащені електронікою, програмним забезпеченням, датчиками та з'єднанням, що дозволяє їм збирати і обмінюватися даними через Інтернет.

SMEs (Small and Medium Enterprises) — це малі та середні підприємства, які відіграють важливу роль в економіці багатьох країн. Вони характеризуються певними критеріями, такими як кількість працівників, річний оборот і баланс.

SSH (Secure Shell) — це протокол мережевої безпеки, який забезпечує шифровану передачу даних між клієнтом і сервером. Він широко використовується для віддаленого адміністрування серверів, безпечної передачі файлів та інших мережевих операцій.

API (Application Programming Interface) — це набір протоколів, інструментів і визначень для побудови програмного забезпечення та взаємодії між різними програмними компонентами. API дозволяють розробникам використовувати функціональність інших додатків, сервісів або платформ без необхідності вивчення їх внутрішнього коду.

Zabbix — це потужна система моніторингу з відкритим кодом, яка використовується для відстеження стану різних мережевих сервісів, серверів, додатків та інших IT-ресурсів. Вона надає засоби для збору даних, їх аналізу, сповіщення про проблеми та створення звітів.

Nagios — це система моніторингу з відкритим кодом, яка використовується для відстеження стану мережевих пристроїв, серверів, програм та послуг. Вона дозволяє системним адміністраторам виявляти та вирішувати проблеми в IT-інфраструктурі до того, як вони вплинуть на роботу користувачів.

POS-системи (Point of Sale) – це програмно-апаратні комплекси, призначені для автоматизації процесів торгівлі та обслуговування клієнтів.

ВСТУП

Актуальність роботи. Використання мережевого обладнання MikroTik для забезпечення безпечних, відмово стійких та масштабованих мереж набирає дедалі більшої популярності. Це явище особливо помітне у великих та середніх компаніях, які відкривають регіональні офіси, магазини, склади та інші приміщення, що потребують доступу до інтернету. Завдяки конкурентоспроможній ціновій політиці MikroTik, організації можуть створювати ефективні та надійні мережеві інфраструктури без значних фінансових витрат.

Однак основні труднощі, з якими стикаються системні адміністратори в своїй роботі, пов'язані з використанням головного інструменту налаштування мережевого обладнання MikroTik – WinBox.

WinBox має досить складний інтерфейс, насичений численними елементами та налаштуваннями. Хоча ці функції безумовно допомагають у тонкому налаштуванні обладнання, вони становлять значні труднощі при спробах внести незначні зміни або використовувати одну конфігурацію на багатьох пристроях. Наприклад, системні адміністратори часто знаходять проблематичним реплікування конфігурацій у різних мережевих середовищах, що може призвести до потенційних невідповідностей та збільшення адміністративного навантаження. Крім того, WinBox може використовуватися лише з операційною системою Windows, що значно ускладнює роботу для організацій, які використовують різноманітні операційні системи, такі як Linux або macOS.

Об'єктом дослідження є мережеве обладнання MikroTik, включаючи його конфігурацію та можливості управління.

Метою роботи є розробка програмного модуля, що забезпечує автоматичне налаштування мережевого обладнання MikroTik. Модуль спрямований на спрощення та оптимізацію процесу налаштування, пропонуючи зручний інтерфейс для управління та розгортання конфігурацій

на пристроях MikroTik. Автоматизуючи повторювані завдання та оптимізуючи робочі процеси конфігурації, модуль підвищить операційну ефективність та знизить ймовірність помилок. Крім того, він забезпечить підтримку крос-платформної сумісності, дозволяючи системним адміністраторам використовувати інструмент налаштування на різних операційних системах. Це рішення буде особливо корисним для масштабних розгортань, де узгоджена конфігурація на багатьох пристроях є критично важливою для підтримання цілісності та продуктивності мережі.

РОЗДІЛ 1. АНАЛІЗ СУЧАСНОГО СТАНУ ВИРІШЕННЯ ПРОБЛЕМИ

У сучасному світі інформаційних технологій надійність та ефективність мережевої інфраструктури є критично важливими аспектами для будь-якої організації. Складні мережеві системи потребують високоякісного обладнання та програмного забезпечення для забезпечення безперебійної роботи, швидкого доступу до інформації та захисту від загроз. Компанія MikroTik, заснована в 1996 році у місті Рига, Латвія, є однією з провідних компаній у галузі розробки мережевих рішень, які задовольняють різноманітні потреби бізнесу [10].

Цей розділ присвячений детальному аналізу сучасного стану вирішення проблем, пов'язаних із налаштуванням та управлінням мережевим обладнанням MikroTik. У ньому розглянуто історію розвитку компанії, її основні продукти та технології, а також проблеми та виклики, з якими стикаються системні адміністратори при налаштуванні цього обладнання. Окрім того, досліджуються можливості використання готових рішень та індивідуальної розробки для оптимізації процесу налаштування та управління.

У перших підрозділах розглядається історія розвитку компанії MikroTik, її основні продукти та характеристики, а також популярність та використання обладнання у різних галузях. Далі досліджуються технології та методи налаштування мережевого обладнання, такі як WinBox, WebFig та командний рядок MikroTik RouterOS (CLI), а також їхні переваги та недоліки.

Особлива увага приділяється проблемам та викликам, з якими стикаються системні адміністратори при налаштуванні обладнання MikroTik, зокрема складності інтерфейсу користувача, обмеженої підтримки операційних систем та реплікації конфігурацій на декілька пристроїв. У підрозділах про готові рішення розглядаються можливості використання систем управління конфігураціями, таких як Ansible, Puppet або Chef, а також MikroTik Cloud Hosted Router (CHR) та The Dude Network Monitor.

Завершальний підрозділ аналізує перспективи використання індивідуальних розробок, таких як кросплатформенні інструменти налаштування та системи управління конфігураціями, що можуть бути адаптовані до специфічних потреб організації. Після всебічного аналізу надано висновки щодо найоптимальніших рішень для ефективного управління мережевим обладнанням MikroTik.

1.1. Огляд мережевого обладнання MikroTik

1.1.1. Історія розвитку та основні характеристики

Заснована у 1996 році у місті Рига, Латвія, компанія MikroTik поставила перед собою завдання розробити передове програмне забезпечення для маршрутизації та мережевих рішень, адаптованих до різноманітних потреб бізнесу, як малого, так і великого. У 1997 році компанія представила свій проривний продукт RouterOS, який швидко отримав популярність завдяки своїм вражаючим можливостям та адаптивності. У міру розвитку MikroTik компанія почала займатися апаратними рішеннями і в 2002 році випустила свій перший маршрутизатор. Продовжуючи розширювати межі, компанія постійно вдосконалювала своє програмне та апаратне забезпечення, розширюючи асортимент продукції та збагачуючи функціональність пристроїв. Важливим моментом став запуск у 2005 році високо оціненої серії маршрутизаторів RB500, яка відзначалася винятковою продуктивністю та доступною ціною, що швидко завоювала увагу ринку.

MikroTik зарекомендувала себе як провідний постачальник передового мережевого обладнання, що включає широкий асортимент маршрутизаторів, комутаторів, точок бездротового доступу та різних інших пристроїв. З глобальною присутністю у понад 140 країнах MikroTik обслуговує різноманітну клієнтську базу, що включає як окремих домашніх користувачів, так і великі корпорації та постачальників інтернет-послуг.

Програмне забезпечення RouterOS володіє вражаючим набором функцій, таких як маршрутизація, брандмауер, VPN, QoS та динамічна маршрутизація. Компанія постійно оновлює RouterOS, додаючи нові функції та покращуючи безпеку. Крім того, використання скриптів для автоматизації завдань значно спрощує управління мережею.

Широкий асортимент маршрутизаторів, комутаторів, точок бездротового доступу та інших пристроїв MikroTik відповідає різним рівням складності та вимогам до продуктивності. Пристрої MikroTik відомі своєю високою продуктивністю, надійністю та стійкістю до збоїв, що робить їх переважним вибором для критично важливих мереж.

Важливою відмінністю компанії MikroTik є прагнення надавати високоякісне мережеве обладнання за доступними цінами, що дозволяє організаціям будь-якого розміру отримувати доступ до передових мережевих рішень без значних фінансових витрат.

1.1.2. Популярність та використання у різних галузях

Мережеве обладнання MikroTik здобуло популярність завдяки своїй багатофункціональності, надійності та доступній ціні. Воно широко використовується в різних галузях, від малих та середніх підприємств до великих корпорацій та постачальників інтернет-послуг.

Пристрої MikroTik постачаються з програмним забезпеченням RouterOS, яке надає широкий спектр функцій для управління мережею. Це робить їх привабливими для професіоналів. Обладнання відоме своєю надійністю та довговічністю, забезпечуючи безперебійну роботу мережі. Пристрої MikroTik є гнучкими та масштабованими, легко конфігуруються відповідно до потреб мережі, що робить їх ідеальним вибором для компаній, які ростуть.

Обладнання MikroTik зазвичай використовується в таких галузях:

- малі та середні підприємства (SMEs) [22]. MikroTik широко використовується в офісних мережах для забезпечення стабільного та швидкого доступу до Інтернету, внутрішніх ресурсів та послуг. Його здатність створювати великі мережі з багатьма віддаленими локаціями робить його придатним для налаштування мереж у віддалених офісах та філіях, забезпечуючи їх з'єднання в єдину мережу;
- великі компанії. MikroTik використовується великими корпораціями для створення відмовостійких та масштабованих корпоративних мереж, що дозволяє реалізовувати функції маршрутизації та VPN-з'єднань;
- готелі та ресторани. MikroTik використовується готелями та ресторанами для створення гостьових мереж, забезпечуючи клієнтам стабільний та безпечний доступ до Інтернету. MikroTik безперебійно інтегрується з системами управління готелями та ресторанами, забезпечуючи безперервну роботу POS-систем, відеоспостереження та інших послуг;
- промисловість та виробництво. Виробничі підприємства використовують MikroTik для управління промисловими мережами, що сприяє стабільному зв'язку між виробничими лініями та центральним офісом. MikroTik інтегрується з пристроями IoT для моніторингу та управління виробничими процесами в реальному часі.

1.2. Технології та методи налаштування мережевого обладнання

WinBox є високоефективним та зручним графічним інструментом управління, що полегшує налаштування та адміністрування пристроїв MikroTik у режимі реального часу [26]. Завдяки багатофункціональному інтерфейсу WinBox дозволяє користувачам легко отримувати доступ до всіх налаштувань і змінювати їх в режимі реального часу, що забезпечує миттєву видимість змін, внесених до пристроїв;

Варто пам'ятати, що WinBox спеціально розроблений для операційної системи Windows і може бути несумісним з іншими операційними системами.

WebFig - це веб-інтерфейс, розроблений спеціально для управління пристроями MikroTik [23]. Він надає користувачам доступ до функцій RouterOS через стандартний веб-браузер, що підвищує гнучкість та доступність. Перевагою використання WebFig є можливість доступу з будь-якого пристрою з веб-браузером та зручність відсутності необхідності в установці додаткового програмного забезпечення.

Однак, варто зазначити, що WebFig є менш зручним та повільнішим у порівнянні з WinBox. Крім того, він має обмежені можливості для управління складними конфігураціями.

Командний рядок MikroTik RouterOS (CLI) надає користувачам повний доступ до всіх наявних команд для налаштування та управління пристроями MikroTik [12]. Це включає можливість повного використання всіх функцій та налаштувань RouterOS, а також можливість автоматизації завдань за допомогою скриптів. Однією з ключових переваг використання CLI RouterOS є його незалежність від операційної системи клієнта, що дозволяє легко інтегруватися на різних платформах.

Однак, існують певні недоліки використання CLI RouterOS. Наприклад, для його використання потрібні глибокі знання RouterOS та мережевих технологій, що може становити труднощі для деяких користувачів. Крім того, існує можливість виникнення помилок при ручному введенні команд, що може призвести до неправильної конфігурації або інших проблем. До того ж, для створення безпечного SSH-з'єднання необхідно використовувати складні паролі та/або SSH-ключі, що додає додатковий рівень складності до процесу.

RouterOS пропонує зручну функцію, яка дозволяє користувачам експортувати поточну конфігурацію пристрою до файлу. Цей файл можна імпортувати на інший пристрій, що дозволяє безперешкодно копіювати налаштування. Переваги використання цієї функції включають можливість

швидкого налаштування нового пристрою шляхом імпорту попередньо існуючої конфігурації. Вона також забезпечує легкий спосіб резервного копіювання та відновлення конфігурацій, що економить час та зусилля у випадку виходу з ладу пристрою або необхідності в повторному налаштуванні.

Слід пам'ятати про деякі недоліки. Під час імпорту конфігурацій з різних середовищ можуть виникати конфлікти. Крім того, необхідне ручне змінення конфігурації для відповідності конкретним вимогам нового пристрою, що додає додатковий рівень складності до процесу.

1.3. Проблеми та виклики у налаштуванні обладнання

Налаштування обладнання MikroTik ставить перед системними адміністраторами різні складні завдання. Однією з головних проблем є складність інтерфейсу користувача. Хоча WinBox пропонує широкий спектр функцій, його інтерфейс часто може стати надмірно перевантаженим через велику кількість параметрів і налаштувань, що призводить до можливої плутанини та збільшення часу налаштування.

Крім того, значною проблемою є обмежена підтримка операційних систем. Основний інструмент налаштування, WinBox, розроблений виключно для операційної системи Windows. Це обмеження створює труднощі, змушуючи адміністраторів шукати обхідні шляхи або використовувати додаткове програмне забезпечення для доступу до WinBox.

Реплікація конфігурацій на декілька пристроїв може бути ускладнена та призводити до помилок, особливо коли конфігурації потребують ручного введення. Це не лише збільшує адміністративне навантаження, але й підвищує ймовірність виникнення помилок.Експорт та імпорт конфігурацій між різними мережевими середовищами та обладнанням може створювати проблеми, що призводить до конфліктів, які потребують додаткового часу на усунення та адаптацію.

1.4. Використання існуючих рішень з налаштування мережевого обладнання

При роботі з мережевим обладнанням MikroTik можна використовувати існуючі рішення для спрощення завдань системних адміністраторів. Одним з ефективних підходів є використання систем управління конфігураціями, таких як Ansible, Puppet або Chef [1, 19, 3]. Ці системи автоматизують процес налаштування мережевого обладнання, надаючи централізоване управління конфігураціями, і підтримують безшовну сумісність з пристроями MikroTik через SSH та API інтеграцію [20, 2].

Системи управління конфігураціями забезпечують автоматизацію повторюваних завдань, підвищуючи ефективність та продуктивність, а також забезпечуючи послідовність та зменшуючи кількість помилок. Вони ефективно керують конфігураціями на великій кількості пристроїв, що полегшує підтримку контролю під час росту організації. Крім того, завдяки наданню центрального сховища для налаштувань конфігурації, ці системи спрощують управління конфігураціями, знижують складність, покращують безпеку та полегшують вирішення проблем і оновлення.

Водночас, налаштування системи управління конфігураціями може вимагати значних початкових зусиль, включаючи навчання персоналу, адаптацію процесів та сумісність інфраструктури. Хоча існують значні довгострокові переваги, початкові витрати пов'язані з ліцензуванням, інфраструктурою та підтримкою для впровадження та роботи системи. Крім того, можуть виникати витрати на навчання.

MikroTik Cloud Hosted Router (CHR) - це віртуалізована версія RouterOS, спеціально розроблена для роботи на хмарних платформах, таких як AWS, Azure та Google Cloud [11]. Це рішення пропонує можливість централізованого управління мережами, використовуючи хмарну інфраструктуру.

CHR дозволяє віддалений доступ до налаштувань та управління мережею з будь-якого місця з доступом до Інтернету, що надає велику гнучкість та зручність. Його можна легко масштабувати вгору або вниз залежно від потреб мережі, що дозволяє безперешкодно адаптуватися до змін у попиті або росту. Крім того, CHR забезпечує високу доступність та стійкість до збоїв, гарантуючи доступність та надійність мережевих послуг навіть у випадку виходу з ладу обладнання або програмного забезпечення.

Однак, використання хмарних послуг може призвести до додаткових витрат, включаючи абонентську плату, витрати на передачу даних та будь-які додаткові послуги, необхідні для підтримки розгортання CHR. Доступ та управління CHR через хмару вимагають стабільного та надійного інтернет-з'єднання, і перерви в інтернет-з'єднанні можуть вплинути на можливість ефективного доступу та управління мережею.

The Dude Network Monitor - це цінний інструмент, розроблений MikroTik, що дозволяє користувачам легко моніторити та керувати своїми мережами [23]. Він пропонує централізований моніторинг мережевих пристроїв та їх конфігурацій, надаючи користувачам оновлення в реальному часі щодо стану їхньої мережі.

The Dude Network Monitor пропонує можливості моніторингу в реальному часі, що дозволяє користувачам завжди бути в курсі продуктивності та стану їхньої мережі. Він дозволяє централізовано керувати великою кількістю мережевих пристроїв з одного інтерфейсу, спрощуючи процес управління та підвищуючи ефективність. Крім того, інструмент доступний безкоштовно, що означає, що користувачі можуть отримати доступ до його потужних можливостей моніторингу та управління без додаткових витрат на ліцензування.

Налаштування The Dude Network Monitor може вимагати певного часу та зусиль, особливо для користувачів, які новачки в цьому інструменті. Попри це, переваги моніторингових можливостей можуть переважити початкові

складнощі налаштування. Хоча The Dude Network Monitor пропонує потужні функції моніторингу та управління, він може мати деякі функціональні обмеження у порівнянні з комерційними рішеннями. Користувачі повинні оцінити, чи відповідають можливості інструмента їхнім конкретним потребам у управлінні мережею.

Використання професійних послуг та підтримки надає компаніям можливість скористатися знаннями професійних консультантів та партнерів MikroTik, які володіють спеціалізованими навичками в налаштуванні та обслуговуванні мережевої інфраструктури.

Компанії отримують доступ до спеціалізованих знань та багатого досвіду професійних консультантів, що дозволяє ефективно вирішувати проблеми та налаштовувати конфігурації мережі. Завдяки цьому, проблеми можуть бути швидко усунені, що зменшує час простою та підвищує надійність мережевих послуг.

Однак, залучення професійних послуг може призвести до значних витрат, оскільки консультації та підтримка професіоналів зазвичай мають високу ціну. Крім того, залежність від зовнішніх консультантів для підтримки мережі може призвести до залежності від їхніх послуг, що може бути ризикованим для компанії у випадку недоступності або зміни консультанта.

1.5. Використання власної розробки для налаштувань обладнання MikriTik

Використання індивідуальної розробки може надати можливості для вирішення проблем, згідно з вимог та можливостей організацій, надаючи велику кількість шляхів реалізації.

Розробка індивідуального програмного модуля для налаштування мережевого обладнання MikroTik надає перевагу вищої кастомізації відповідно до унікальних вимог організації. Існують різні можливості для

вирішення проблем через персоналізовану розробку, включаючи створення кросплатформенного інструменту налаштування.

Однією з ключових переваг кросплатформенного інструменту налаштування є його здатність працювати на різних операційних системах. Використовуючи мови програмування, які підтримують різні платформи, такі як Python, інструмент можна використовувати на Windows, Linux та macOS. Ця гнучкість дозволяє користувачам вибирати своє улюблене середовище без обмеження на конкретну платформу.

Добре розроблений кросплатформенний інструмент налаштування може запропонувати інтуїтивно зрозумілий та зручний інтерфейс, що спрощує налаштування та управління мережевим обладнанням для мережевих адміністраторів і користувачів. Інтуїтивний характер інтерфейсу покращує процес налаштування, підвищуючи користувацький досвід та продуктивність.

Ще однією перевагою є здатність інструменту адаптуватися до конкретних вимог організації. Організації можуть кастомізувати інструмент налаштування для відповідності їхнім унікальним потребам, додаючи специфічні функції та інтеграції. Ця можливість кастомізації дозволяє створювати індивідуальні рішення, що відповідають специфічним викликам та особливостям різних організаційних установок.

Однак, незважаючи на значні переваги кастомізованого кросплатформенного інструменту налаштування, важливо враховувати пов'язані з ним витрати на розробку. Кастомна розробка часто вимагає значних інвестицій у вигляді часу, людських ресурсів та фінансових витрат. Організації повинні ретельно зважувати переваги та витрати, щоб визначити доцільність розробки такого інструменту. Після початкової розробки необхідне постійне обслуговування та регулярні оновлення для забезпечення ефективності та безпеки інструменту налаштування. Це вимога постійного обслуговування та оновлень потребує виділених ресурсів та постійної

підтримки, що додає до загальної вартості володіння та управління інструментом.

Хоча кросплатформенний інструмент налаштування пропонує значні переваги з точки зору гнучкості, користувацького досвіду та кастомізації, організації повинні оцінити пов'язані з ним витрати на розробку та довгострокове обслуговування перед тим, як приступити до такого проекту.

Розробка та впровадження індивідуальної системи управління конфігураціями надає кілька важливих переваг. Однією з ключових переваг є централізоване управління, що дозволяє створити єдиний і автоматизований процес налаштування. Розробка індивідуальної системи для централізованого управління конфігураціями дозволяє організаціям оптимізувати свої операції та забезпечити послідовність у всій мережі.

Крім того, індивідуальна система управління конфігураціями може бути налаштована для безшовної інтеграції з існуючими системами управління та моніторингу мережі. Ця можливість інтеграції є ключовою для організацій, які вже мають встановлені системи і хочуть підвищити їх функціональність без порушення поточних операцій. Ще однією значною перевагою є масштабованість. Індивідуальна система управління конфігураціями може бути розроблена для підтримки великої кількості пристроїв та мережевих середовищ, забезпечуючи гнучкість для розширення та адаптації до змінюваних бізнесових потреб та технологічних вимог.

Однак, важливо враховувати можливі виклики та недоліки, пов'язані з розробкою індивідуальної системи управління конфігураціями. Одним із головних недоліків є складність, пов'язана з її розробкою. Створення такої системи вимагає значних ресурсів та високого рівня експертизи та кваліфікації розробників. Тому організації повинні ретельно оцінити свої можливості та виділити достатні ресурси для забезпечення успішного впровадження. Крім того, необхідне постійне обслуговування та оновлення, щоб забезпечити ефективність роботи індивідуальної системи управління конфігураціями. Ця

постійна потреба в обслуговуванні та оновленнях додає до загальної вартості володіння та потребує виділеної уваги та ресурсів для забезпечення надійності та відповідності сучасним мережевим технологіям та стандартам безпеки.

При розробці веб-інтерфейсу для налаштування слід враховувати кілька переваг. По-перше, веб-інтерфейс надає доступність, оскільки дозволяє користувачам отримувати доступ до налаштувань з будь-якого пристрою з веб-браузером. Це означає, що налаштування можуть бути зручно доступні зі смартфонів, планшетів та комп'ютерів, що забезпечує гнучкість для користувачів. Крім того, веб-інтерфейс дозволяє створювати інтуїтивно зрозумілий дизайн, який є важливим для користувачів для легкого навігації та налаштування пристроїв. Можливість створення зручного та зрозумілого інтерфейсу покращує користувацький досвід та підвищує загальне задоволення. Також веб-інтерфейс пропонує підтримку різних платформ, що означає, що його можна використовувати на різних операційних системах. Ця універсальність робить інструмент широко доступним та усуває обмеження, пов'язані з операційною системою користувача.

Однак, слід також враховувати потенційні недоліки. Однією з головних проблем є безпека. Необхідно впровадити високі заходи безпеки, щоб запобігти несанкціонованому доступу до веб-інтерфейсу та забезпечити захист конфіденційної інформації. Іншою важливою складністю є вартість розробки, пов'язана зі створенням та підтримкою веб-інтерфейсу. Цей процес вимагає значних ресурсів та постійного обслуговування, щоб забезпечити ефективність роботи інтерфейсу та його відповідність сучасним вимогам. Інтеграція з системами моніторингу та управління має численні переваги для мережевих адміністраторів.

Інтеграція з існуючими системами моніторингу, такими як Zabbix або Nagios, дозволяє централізовано моніторити стан мережевих пристроїв, що означає, що всі пристрої можна контролювати з одного місця, спрощуючи виявлення та вирішення проблем. Можливість автоматизувати управління та

налаштування пристроїв на основі даних моніторингу також є великою перевагою. Це спрощує рутинні завдання та зменшує необхідність ручного втручання, що призводить до покращення ефективності та зменшення людських помилок. Інтеграція з системами моніторингу покращує діагностику та вирішення проблем, надаючи всебічні дані для виявлення та усунення проблем більш ефективно.

Однак інтеграція з існуючими системами може бути складною і вимагати значних зусиль у розумінні існуючої інфраструктури, забезпеченні сумісності та впровадженні необхідних з'єднань. Крім того, постійне обслуговування та оновлення інтеграційних рішень необхідні для забезпечення належного функціонування системи та її сумісності з будь-якими змінами або оновленнями в існуючих системах моніторингу та управління. Регулярна увага потрібна для забезпечення безперебійної роботи інтеграції.

1.6. Висновок

Після детального аналізу різних варіантів вирішення проблем, пов'язаних із налаштуванням мережевого обладнання MikroTik, найоптимальнішим рішенням виявляється розробка кросплатформенного інструменту налаштування. Цей варіант було обрано завдяки наступним ключовим перевагам:

- кросплатформенна сумісність. Інструмент, розроблений із використанням мов програмування, які підтримують різні операційні системи (наприклад, Python або JavaScript із використанням Electron), дозволить його використовувати на Windows, Linux та macOS. Це особливо важливо для організацій, що використовують різноманітні операційні системи. Наявність єдиного інструменту для всіх платформ спростить роботу системних адміністраторів, зменшить потребу в додатковому програмному забезпеченні

(наприклад, віртуальних машинах або емуляторах) та покращить загальну ефективність роботи;

- інтуїтивно зрозумілий інтерфейс. Можливість створення зручного та інтуїтивно зрозумілого інтерфейсу користувача значно спростить налаштування мережевого обладнання. Складний інтерфейс WinBox часто викликає труднощі при внесенні невеликих змін або використанні однієї конфігурації на кількох пристроях. Новий інструмент може запропонувати більш зрозумілі меню, спрощені форми введення даних та інтерактивні підказки, що зменшить ймовірність помилок та скоротить час навчання нових користувачів;

- адаптація до специфічних потреб. Індивідуальна розробка дозволяє адаптувати інструмент до специфічних потреб організації, додаючи конкретні функції та інтеграції. Це включає підтримку скриптів для автоматизації завдань, можливість швидкого імпорту та експорту конфігурацій, а також функції моніторингу та діагностики мережі. Крім того, він може враховувати особливості різних мережевих середовищ та уникати проблем з реплікацією конфігурацій, забезпечуючи їх послідовність на кількох пристроях.

Хоча розробка кросплатформенного інструменту вимагає значних ресурсів, часу та фінансових інвестицій, його переваги значно переважують ці витрати. Індивідуальний інструмент забезпечить гнучкість, адаптивність та зручність в управлінні мережевим обладнанням MikroTik, сприяючи підвищенню ефективності та надійності мережевої інфраструктури організації. Це рішення буде особливо корисним для масштабних розгортань, де послідовність конфігурацій на кількох пристроях є критично важливою для підтримки цілісності та продуктивності мережі.

РОЗДІЛ 2. РОЗРОБКА ПРОГРАМНОГО МОДУЛЮ АВТОМАТИЗОВАНОГО НАЛАШТУВАННЯ КОНФІГУРАЦІЇ МЕРЕЖЕВОГО ОБЛАДНАННЯ MIKROTIK

2.1. Технічне завдання

2.1.1. Назва розробки

Програмний модуль автоматизованого налаштування конфігурації мережевого обладнання MikroTik.

2.1.2. Призначення розробки

Модуль призначений для автоматизації процесу налаштування мережевого обладнання MikroTik, забезпечення зручного інтерфейсу для управління конфігураціями та підтримки крос-платформної сумісності. Він дозволяє адміністраторам мереж здійснювати налаштування мережевого обладнання без необхідності ручного введення команд або складних скриптів, що значно спрощує процес налаштування та знижує ризик помилок.

2.1.3. Завдання розробки

Програмний модуль автоматизованого налаштування конфігурації мережевого обладнання MikroTik (далі – Модуль) призначений для виконання наступних функцій:

- автоматизація налаштування мережевого обладнання MikroTik. Модуль наддасть адміністраторам можливість автоматизувати налаштування різних параметрів мережевого обладнання MikroTik, що ефективно зменшить залежність від ручних процесів налаштування. Ця автоматизація спростить управління та налаштування пристроїв MikroTik, що призведе до підвищення ефективності та зменшення ймовірності помилок;

- зменшення часу на налаштування. Автоматизація процесів дозволяє адміністраторам мереж швидко та ефективно налаштовувати мережеве обладнання, що призводить до економії часу та ресурсів;
- забезпечення крос-платформної сумісності. Модуль буде підтримувати роботу на різних операційних системах (Windows, Linux, macOS), що забезпечить зручність використання для адміністраторів, які працюють з різними платформами;
- підтримка адміністраторів мереж різних масштабів, від малих підприємств до великих корпоративних мереж, забезпечуючи ефективне та надійне управління мережевою інфраструктурою на базі обладнання MikroTik.

2.1.4. Цілі розробки

Цілями розробки є:

- спрощення процесу налаштування мережевого обладнання. Надання простого та інтуїтивно зрозумілого інтерфейсу для адміністраторів мереж та мінімізація кількості ручних операцій, необхідних для налаштування обладнання;
- зменшення кількості помилок під час конфігурації. Автоматизація завдань та процесів налаштування, які виконуються часто, а також впровадження систем для перевірки точності конфігурацій перед їх застосуванням, дозволяють спростити операції та зменшити ймовірність помилок;
- підвищення продуктивності адміністраторів. Завдяки можливості швидкого перенесення та отримання конфігурацій, користувачі можуть легко імпортувати та експортувати налаштування для використання на різних пристроях;
- забезпечення крос-платформної сумісності. Розробка модуля, який підтримує роботу на різних операційних системах, таких як Windows, Linux та macOS;

- розширюваність та гнучкість архітектури модуля. Платформа сприяє безперебійному інтегруванню нових функціональностей та забезпечує сумісність з новітніми моделями пристроїв MikroTik.

2.1.5. Вимоги до функціональних характеристик

Основні вимоги до функціональних характеристик:

- імпорт та експорт конфігурацій. Експорт поточних налаштувань обладнання у файл, що дозволяє зберегти всі конфігурації та параметри в одному файлі. Також можна імпортувати конфігурації з файлу для створення нового обладнання з імпортованими налаштуваннями або оновлення налаштувань існуючого обладнання;
- підтримка всіх основних моделей обладнання mikrotik;
- графічний інтерфейс. Для легкого налаштування та управління обладнанням;
- логічна структура меню та налаштувань. Для швидкого доступу до необхідних параметрів;
- крос-платформна сумісність. Підтримка операційних систем Windows, Linux та macOS;
- безпека. Використання захищених протоколів для передачі даних (SSL/TLS, SSH).

Додаткові вимоги:

- можливість подальшого розширення. Архітектура модуля повинна бути модульною, що дозволяє безперебійно додавати нові функції та компоненти без необхідності значних змін у наявному коді. Це забезпечить легку масштабованість та обслуговування модуля, одночасно гарантуючи його адаптованість до майбутніх вимог та вдосконалень.

2.1.6. Стадії розробки

Розробка програмного модуля проходитиме через кілька ключових стадій, кожна з яких спрямована на забезпечення високої якості та функціональності кінцевого продукту:

1. Проектування архітектури:

- вибір технологій та інструментів для розробки;
- визначення основних компонентів модуля та їх взаємодії;
- розробка архітектури програмного та функціонально-логічних блоків.

2. Проектування інтерфейсів користувача.
3. Розробка коду:

- реалізація інтерфейсу користувача;
- написання коду згідно з розробленими алгоритмами;
- реалізація основних функціональних можливостей модуля.

4. Функціональне тестування:

- проведення тестування основних функцій модуля для перевірки їх відповідності вимогам;
- виявлення та виправлення помилок та недоліків.

2.2. Вибір технологій та інструментів для розробки

2.2.1. Мова програмування

При виборі мови програмування для розробки програмного модуля автоматизованого налаштування конфігурації мережевого обладнання MikroTik розглядались різні варіанти, такі як C++, Java, C# та інші мови. Кожна з цих мов має свої переваги та недоліки, які слід враховувати при прийнятті рішення.

C++ є потужною мовою програмування, що забезпечує високу продуктивність та ефективність роботи. Вона пропонує широкі можливості

для низькорівневого доступу до системних ресурсів. Проте C++ має складнішу синтаксичну структуру порівняно з Python, що може збільшити час розробки та вимоги до кваліфікації розробників. Відсутність вбудованих бібліотек для роботи з мережевими протоколами та SSH-з'єднаннями ускладнює розробку, оскільки потрібно використовувати сторонні бібліотеки або самостійно реалізовувати відповідний функціонал. Також процес налаштування та компіляції є складнішим, ніж у Python.

Java забезпечує портативність завдяки Java Virtual Machine (JVM) та широко використовується в корпоративному середовищі. Однак, важчий процес налаштування та використання для мережевих завдань порівняно з Python, складніший синтаксис і відсутність вбудованих бібліотек для роботи з графічними інтерфейсами робить її менш привабливою для даного проекту. Хоча існують бібліотеки для роботи з SSH (наприклад, JSch), їх використання може бути складнішим.

C# пропонує сильну інтеграцію з Windows-платформою та потужні інструменти розробки, такі як Visual Studio. Проте C# має обмежену крос-платформову підтримку у порівнянні з Python. Хоча .NET Core покращує ситуацію, Python все ще пропонує більш гнучку крос-платформову підтримку. Складніша у використанні для роботи з мережевими протоколами та SSH-з'єднаннями також робить C# менш привабливою для даного проекту.

Враховуючи всі ці фактори, Python виявився найкращим вибором з точки зору функціональності, зручності використання, масштабованості та підтримки. Python має багатий набір бібліотек, що значно полегшують розробку мережевих додатків. Зокрема, бібліотеки paramiko і netmiko забезпечують легкий і безпечний спосіб взаємодії з мережевими пристроями через SSH. Це робить Python ідеальним вибором для задач, пов'язаних з автоматизацією мережевого адміністрування.

Крім того, Python забезпечує крос-платформову сумісність, підтримуючи роботу на різних операційних системах, таких як Windows, Linux

і macOS. Це дозволяє розробити універсальне рішення, яке буде працювати на будь-якій платформі без необхідності значних змін у коді, що важливо для адміністраторів мереж, які можуть використовувати різні операційні системи.

Python відомий своєю простою і зрозумілою синтаксичною структурою, що дозволяє розробникам швидко навчитися працювати з цією мовою і зосередитися на вирішенні задач, а не на деталях мови програмування. Простота мови також знижує ймовірність виникнення помилок у коді, що підвищує загальну ефективність розробки.

Ще однією вагомою причиною вибору Python є швидкість розробки. Завдяки своїй простоті і великій кількості готових бібліотек, Python дозволяє швидко розробляти і прототипувати додатки. Це особливо важливо на етапах початкової розробки та тестування, коли необхідно швидко отримати робочий прототип для перевірки гіпотез і тестування функціональності.

Активна спільнота розробників Python забезпечує швидке вирішення проблем, доступ до великої кількості навчальних матеріалів і прикладів, а також регулярні оновлення та покращення самої мови і її бібліотек. Велика кількість доступних ресурсів допомагає швидко знаходити рішення для виникаючих проблем і забезпечує підтримку на кожному етапі розробки.

Python також пропонує інтеграцію з інструментами для розробки графічного інтерфейсу користувача. Використовуючи вбудовану бібліотеку Tkinter, можна створити інтуїтивно зрозумілий інтерфейс для введення даних про підключення, вибору файлів конфігурації, запуску команд та перегляду результатів. Це значно спрощує взаємодію користувача з програмним модулем.

Отже, Python був обраний завдяки своїй простоті, гнучкості, багатому набору бібліотек, крос-платформовій підтримці, швидкості розробки та активній спільноті. Ці фактори роблять Python ідеальним вибором для реалізації ефективного, надійного та зручного у використанні рішення для автоматизації налаштування мережевого обладнання MikroTik.

2.2.2. Вибір бібліотек та інструментів

Для розробки програмного модуля автоматизованого налаштування конфігурації мережевого обладнання MikroTik було ретельно обрано кілька основних бібліотек та інструментів. Вони забезпечують необхідну функціональність, зручність використання та масштабованість, що є критично важливим для створення ефективного та надійного рішення.

Однією з ключових бібліотек, що використовується у розробці, є Netmiko [11]. Netmiko спрощує автоматизацію мережевих завдань через SSH, що є важливим аспектом для налаштування мережевого обладнання. Ця бібліотека підтримує широкий спектр мережевих пристроїв, включаючи MikroTik. Завдяки Netmiko можна легко підключатися до пристроїв, виконувати необхідні команди та отримувати результати їх виконання. Це значно полегшує процес автоматизації, оскільки забезпечує зручний інтерфейс для взаємодії з мережевими пристроями, дозволяючи адміністраторам швидко та ефективно виконувати завдання конфігурації.

Ще однією важливою бібліотекою є Paramiko, яка забезпечує роботу з SSH та SFTP [12]. Використання Paramiko дозволяє встановлювати надійні та безпечні з'єднання з мережевими пристроями, що є критично важливим для забезпечення безпеки всіх операцій. Paramiko дозволяє виконувати команди на віддалених серверах та передавати файли, що робить її незамінною для управління мережевими пристроями. Висока надійність та безпека, яку забезпечує ця бібліотека, є важливими аспектами для розробки програмного модуля, оскільки будь-які проблеми з безпекою можуть призвести до серйозних наслідків у мережевій інфраструктурі.

Для створення графічного інтерфейсу користувача була обрана бібліотека Tkinter [13]. Tkinter є вбудованою в стандартну бібліотеку Python, що робить її легко доступною та простою у використанні. Вона не потребує додаткових інсталяцій і підтримує крос-платформову роботу, що дозволяє

створювати зручні та інтуїтивно зрозумілі інтерфейси для взаємодії з користувачами. Використання Tkinter дозволяє розробити інтерфейс, який забезпечить зручність введення даних про підключення, вибору файлів конфігурації, запуску команд та перегляду результатів. Це значно покращує користувацький досвід, роблячи програмний модуль більш доступним для широкого кола користувачів.

Для зберігання локальних даних, таких як інформація про підключення, була обрана система управління базами даних SQLite [14]. SQLite є легкою системою, яка вбудована в Python і не потребує додаткових серверів чи налаштувань. Це робить її ідеальною для зберігання локальних даних, оскільки вона забезпечує швидкий доступ до необхідної інформації та зручність управління даними. Використання SQLite дозволяє легко зберігати та отримувати дані про підключення, що є важливим для забезпечення надійного та швидкого доступу до конфігураційних даних.

Інструмент PyInstaller був обраний для перетворення Python-кодів у виконувані файли. Використання PyInstaller дозволяє створювати крос-платформові виконувані файли, що спрощує розповсюдження додатків. Це забезпечує можливість легко розповсюджувати розроблений програмний модуль серед користувачів різних операційних систем без необхідності додаткових налаштувань. Таким чином, PyInstaller дозволяє забезпечити зручність використання та доступність програмного модуля для широкого кола користувачів, незалежно від операційної системи, яку вони використовують.

Загалом, вибір цих бібліотек та інструментів для розробки програмного модуля автоматизованого налаштування конфігурації мережевого обладнання MikroTik був обумовлений їх функціональністю, зручністю використання та здатністю забезпечити масштабованість рішення. Використання Netmiko, Paramiko, Tkinter, SQLite та PyInstaller дозволяє створити ефективне, надійне та зручне у використанні рішення, яке відповідає всім вимогам сучасного

мережевого адміністрування. Цей вибір інструментів сприяє швидкому та ефективному виконанню завдань, забезпечуючи високу продуктивність та надійність програмного модуля.

2.3. Визначення основних компонентів модуля та їх взаємодії

Для створення програмного модуля автоматизованого налаштування конфігурації мережевого обладнання MikroTik необхідно визначити основні компоненти системи та їх взаємодію. Це допоможе структурувати процес розробки, забезпечити зрозумілість архітектури модуля та ефективність його роботи.

Основними компонентами є графічний інтерфейс користувача (GUI), менеджер підключень, компонент обробки конфігурацій, система управління базами даних та компонент обробки помилок і логування. GUI складається з головного вікна, що містить основні елементи керування (поля вводу, кнопки, меню), форми вводу даних для введення IP-адреси, MAC-адреси, порту, імені користувача та пароля, а також області повідомлень для відображення поточного стану операцій та повідомлень про помилки або успіх. Головне вікно також забезпечує доступ до різних функцій програми, таких як завантаження конфігурацій, збереження поточного стану та інші налаштування.

Менеджер підключень включає модуль підключення через SSH, який використовує бібліотеки Netmiko та Paramiko для встановлення та управління SSH-з'єднаннями з пристроями MikroTik. Він відповідає за обробку введених користувачем даних, таких як IP-адреса, ім'я користувача та пароль, для встановлення безпечного з'єднання з пристроєм MikroTik. Менеджер підключень також зберігає ці дані в базі даних SQLite для швидкого доступу в майбутньому, що дозволяє користувачам легко відновлювати підключення без необхідності повторного введення всіх даних.

Компонент обробки конфігурацій виконує ключові функції модуля - експорт, імпорт та редагування конфігурацій. Експорт конфігураційного файлу дозволяє зберегти поточні налаштування пристрою MikroTik у вигляді файлу, який можна згодом імпортувати на інший пристрій або використовувати як резервну копію. Імпорт конфігураційного файлу дозволяє завантажити збережені налаштування на пристрій, що значно спрощує процес налаштування нового обладнання. Редактор конфігурацій дозволяє користувачам вносити зміни в конфігураційні файли перед їх імпортом, забезпечуючи можливість адаптації налаштувань до специфічних вимог мережі.

Система управління базами даних складається з SQLite бази даних, яка зберігає дані про підключення, конфігурації та історію операцій, і модуля доступу до даних, який забезпечує взаємодію з базою даних для збереження та отримання необхідної інформації. Це дозволяє ефективно керувати даними про підключення та конфігурації, забезпечуючи швидкий доступ до них і можливість збереження історії операцій для подальшого аналізу.

Компонент обробки помилок та логування включає систему логування, що записує всі операції та помилки у файл журналу для подальшого аналізу, та обробку помилок, яка визначає та обробляє помилки, що виникають під час виконання операцій. Це забезпечує надійність роботи модуля, фіксуючи всі операції та помилки, що виникають під час роботи, і дозволяє швидко знаходити і виправляти проблеми.

Взаємодія між компонентами відбувається згідно з наступною послідовністю дій, відображеною на діаграмі (рисунок 2.1):

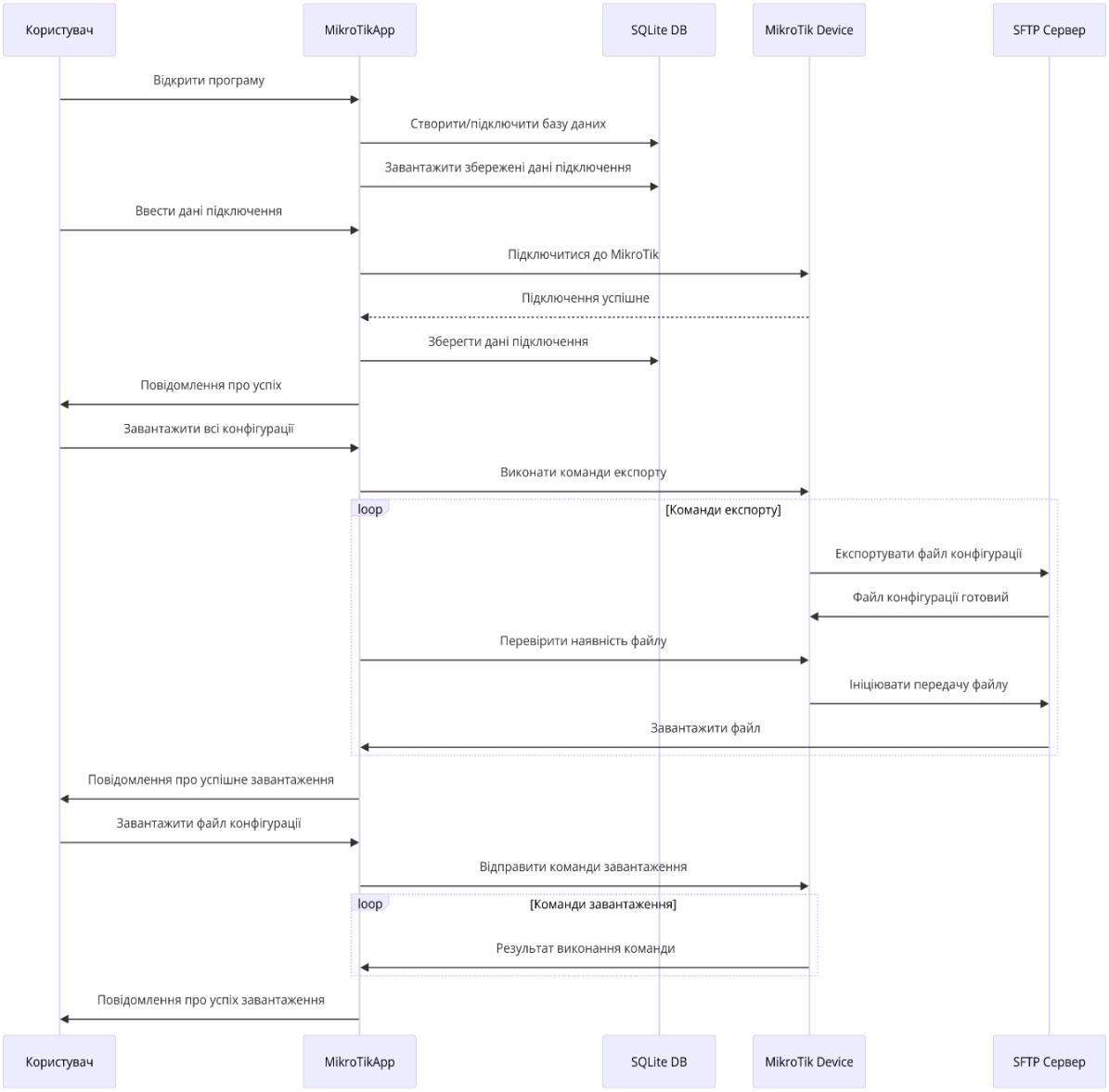

Рисунок 2.1 - Діаграма взаємодії основних компонентів програмного модуля

Користувач відкриває програму, яка підключається до бази даних SQLite та завантажує збережені дані підключення. Після введення даних підключення у GUI програма встановлює SSH-з'єднання з пристроєм MikroTik. У разі успішного підключення, дані підключення зберігаються в базі даних, і користувач отримує повідомлення про успіх. Користувач може завантажити всі конфігурації, відправивши команду експорту на пристрій. Пристрій експортує конфігураційний файл на SFTP сервер, після чого файл завантажується назад у програму. Користувач може завантажити файл

конфігурації на пристрій, відправляючи команди завантаження через GUI. Результати виконання команд відображаються користувачу через GUI, який повідомляє про успіх завантаження.

Ця структура забезпечує надійне, зручне та ефективне управління мережевими конфігураціями пристроїв MikroTik. Використання бібліотек Netmiko та Paramiko забезпечує безпечне і стабільне SSH-з'єднання, а інтеграція з базою даних SQLite дозволяє зберігати та швидко відновлювати підключення та конфігурації. Компонент обробки конфігурацій дозволяє легко експортувати та імпортувати конфігураційні файли, а також редагувати їх перед завантаженням, забезпечуючи гнучкість та адаптивність налаштувань. Компонент обробки помилок та логування забезпечує надійність і дозволяє швидко виявляти та виправляти помилки, що виникають під час роботи.

2.4 Розробка архітектури програмного забезпечення

2.4.1. Функціонально-логічні блоки

Розроблений програмний продукт складається з таких функціонально-логічних блоків: графічний інтерфейс користувача (GUI), менеджер підключень, компонент обробки конфігурацій, система управління базами даних, компонент обробки помилок та логування. Взаємодію між якими представлено на діаграмі (рисунок 2.2):

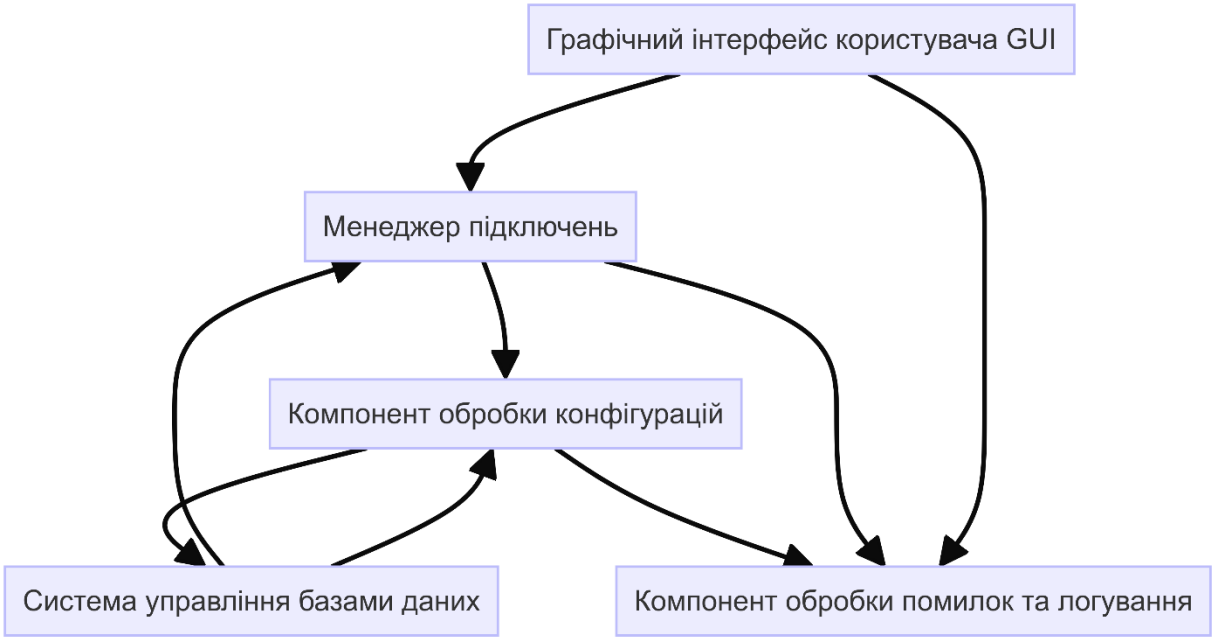

Рисунок 2.2 - Діаграма взаємодію функціонально-логічних блоків

Графічний інтерфейс користувача (GUI) забезпечує взаємодію користувача з програмним модулем. Він містить елементи введення даних, кнопки керування та область для відображення стану операцій та повідомлень. GUI відповідальний за зчитування введених даних і передачу їх відповідним компонентам. Основні елементи GUI включають поля для введення IP-адреси, MAC-адреси, порту, імені користувача та пароля, кнопки для підключення, завантаження, вивантаження та редагування конфігурацій, а також повідомлення про успіх чи помилки операцій.

Менеджер підключень відповідає за встановлення та управління підключеннями до мережевих пристроїв MikroTik. Він використовує бібліотеки Netmiko та Paramiko для встановлення SSH-з'єднань і зберігає дані про підключення в базі даних SQLite. Основні функції менеджера підключень включають установку SSH-з'єднання на основі введених користувачем даних, а також збереження та завантаження даних про підключення.

Компонент обробки конфігурацій відповідає за експорт, імпорт та редагування конфігураційних файлів пристроїв MikroTik. Він включає модулі

для експорту конфігураційних файлів, імпорту та редагування. Основні функції компонента обробки конфігурацій включають виконання команд експорту конфігурацій та завантаження файлів, виконання команд імпорту конфігураційних файлів, а також надання можливості редагування конфігураційних файлів перед імпортом.

Система управління базами даних відповідає за збереження та отримання даних про підключення і конфігурації. Вона використовує базу даних SQLite. Основні функції системи управління базами даних включають збереження інформації про підключення та конфігурації, а також отримання збережених даних для використання в інших компонентах.

Компонент обробки помилок та логування відповідає за обробку помилок і логування операцій. Він включає систему логування для запису всіх операцій та помилок. Основні функції компонента обробки помилок та логування включають запис операцій та помилок для подальшого аналізу, а також визначення та обробку помилок для забезпечення надійності роботи модуля.

2.4.2. Алгоритм підключення до пристрою MikroTik

Алгоритм підключення до пристрою MikroTik призначений для встановлення безпечного SSH-з'єднання з мережевим обладнанням, використовуючи введені користувачем дані. Процес починається з введення користувачем необхідних для підключення даних, таких як IP-адреса, MAC-адреса, порт, ім'я користувача та пароль. Ці дані є критично важливими для встановлення з'єднання та ідентифікації пристрою в мережі.

Після введення даних алгоритм перевіряє їх на повноту та коректність. Якщо будь-яке з обов'язкових полів залишилося незаповненим або містить некоректні значення, виводиться повідомлення про помилку. Користувачеві пропонується перевірити та повторно ввести дані. Це забезпечує захист від

некоректних або неповних даних, що можуть завадити встановленню з'єднання.

Коли дані успішно перевірені, алгоритм ініціює спробу встановлення SSH-з'єднання з пристроєм MikroTik. Використовується стандартний SSH-протокол для створення захищеного каналу зв'язку між клієнтом та сервером. Це забезпечує безпечну передачу даних і запобігає можливим загрозам безпеки під час налаштування пристрою.

У випадку успішного підключення дані про підключення, такі як IP-адреса, MAC-адреса, порт, ім'я користувача та час підключення, зберігаються в базі даних. Це дозволяє зберігати історію підключень для подальшого використання та моніторингу. Користувач отримує повідомлення про успішне підключення, що підтверджує коректність введених даних та налаштувань.

Якщо з'єднання не вдалося встановити, виводиться повідомлення про помилку з описом можливих причин, таких як невірні дані, недоступний пристрій або помилки мережі. Користувачеві пропонується повторити спробу підключення після виправлення помилок. Це дозволяє користувачеві швидко виявляти та усувати проблеми, що можуть виникнути під час підключення.

На блок-схемі (рисунок 2.3) показано процес введення даних, їх перевірки, спроби підключення та обробки результатів. Алгоритм включає механізми повторних спроб підключення у випадку тимчасових збоїв. Це підвищує надійність з'єднання та мінімізує вплив тимчасових проблем з мережею.

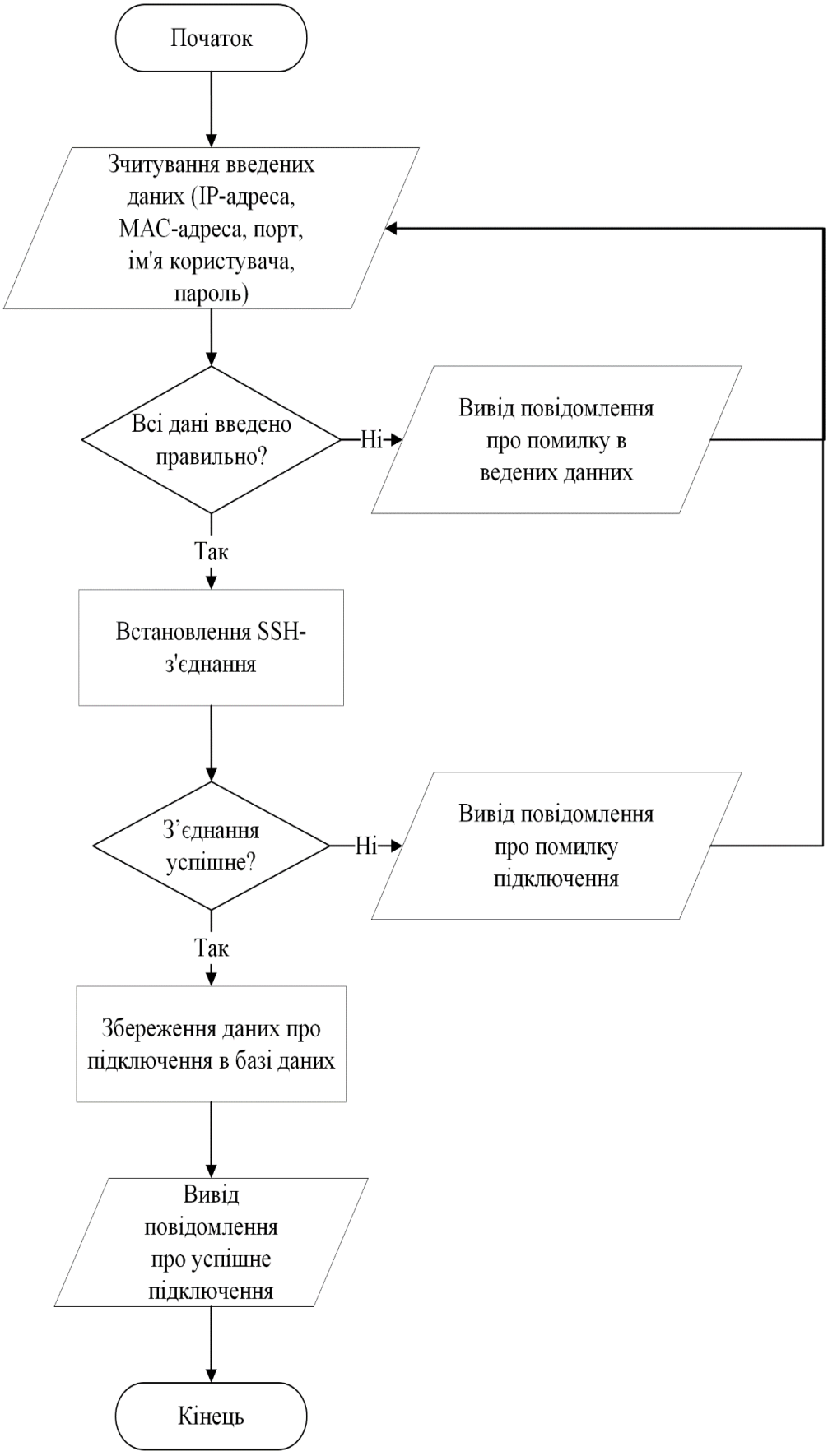

Рисунок 2.3 - Блок-схема алгоритму підключення до пристрою MikroTik

2.4.3. Алгоритм експорту конфігурацій

Алгоритм забезпечує експорт конфігураційних файлів з пристрою MikroTik і завантаження їх на локальний комп'ютер. Після встановлення SSH-з'єднання запускаються команди експорту конфігурацій для різних компонентів мережевого обладнання, таких як IP-адреси, мітки файрволу, NAT, фільтри, черги, DNS, скрипти, розкладники, налаштування електронної пошти, списки адрес, маршрути, мережі DHCP-сервера, типи черг, дерево черг, налаштування інтерфейсів, пули IP-адрес, профілі PPP, логування та загальний експорт конфігурації.

Після успішного встановлення SSH-з'єднання з пристроєм MikroTik алгоритм ініціює процес експорту конфігураційних даних. Для кожного компонента мережевого обладнання запускається відповідна команда експорту. Наприклад, для експорту налаштувань IP-адрес використовується команда export ip address, для експорту міток файрволу — команда export firewall mangle, і так далі для кожного компонента. Ці команди дозволяють отримати повний набір конфігураційних даних для кожного аспекту мережевих налаштувань.

Після виконання кожної команди експорту конфігураційний файл, створений на пристрої MikroTik, завантажується на локальний комп'ютер. Цей процес включає передачу файлу через захищене SSH-з'єднання, що гарантує безпечне переміщення даних. Після кожного успішного завантаження файлу користувач отримує повідомлення про успіх, що підтверджує коректність виконання операції.

Якщо завантаження не вдалося, алгоритм виводить повідомлення про помилку. Це повідомлення може містити інформацію про можливі причини збою, такі як проблеми з підключенням, недостатні права доступу або помилки у виконанні команд експорту. Після цього алгоритм автоматично повторює

спробу завантаження файлу. Цей процес повторюється до успішного завершення операції або до досягнення встановленої кількості спроб.

На блок-схемі (рисунок 2.4) показано процес встановлення SSH-з'єднання, виконання команд експорту, завантаження файлів на локальний комп'ютер та обробки результатів. Важливою частиною алгоритму є механізм повторних спроб завантаження, що підвищує надійність процесу і забезпечує успішне завершення навіть при виникненні тимчасових збоїв.

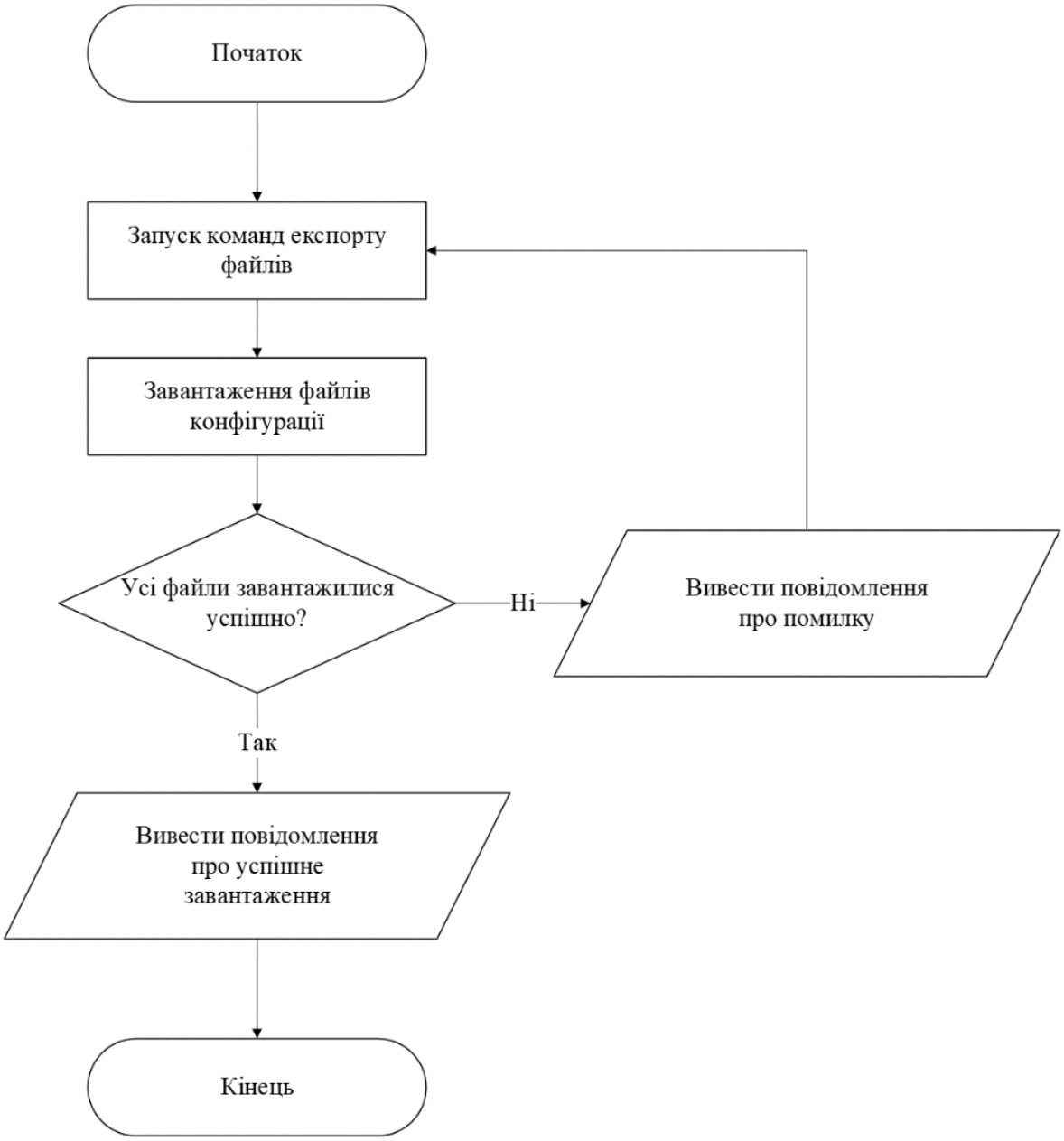

Рисунок 2.4 - Блок-схема алгоритму експорту конфігурацій

2.4.4. Алгоритм імпорту конфігурацій

Алгоритм забезпечує імпорт конфігураційних файлів на пристрій MikroTik, дозволяючи користувачеві легко відновлювати налаштування або оновлювати конфігурацію мережевого обладнання. Процес починається з вибору користувачем конфігураційного файлу для імпорту. Цей файл може містити налаштування різних компонентів мережевого обладнання, які були попередньо експортовані та збережені.

Після вибору файлу алгоритм встановлює SSH-з'єднання з пристроєм MikroTik. Використання SSH забезпечує безпечне з'єднання між клієнтом та сервером, що дозволяє безпечно передавати конфігураційні дані. Після встановлення з'єднання алгоритм ініціює процес імпорту конфігураційних даних з вибраного файлу.

Кожна команда імпорту виконується послідовно, відповідно до змісту конфігураційного файлу. Для кожного компонента мережевого обладнання (наприклад, IP-адреси, мітки файрволу, NAT, фільтри тощо) виконуються відповідні команди імпорту. Після виконання кожної команди алгоритм перевіряє успішність імпорту. Це забезпечує контроль за процесом імпорту і дозволяє вчасно виявити та виправити можливі проблеми.

У разі успішного імпорту користувач отримує повідомлення про успіх. Це підтверджує, що конфігураційні дані були коректно застосовані до пристрою, і мережеве обладнання налаштоване відповідно до вибраного файлу. Якщо імпорт не вдалося, виводиться повідомлення про помилку, що містить інформацію про можливі причини збою. Наприклад, це може бути через неправильний формат файлу, помилки в командах або проблеми з підключенням.

Алгоритм автоматично повторює спробу імпорту у разі невдачі. Цей процес повторюється до успішного завершення операції або до досягнення встановленої кількості спроб. Механізм повторних спроб підвищує надійність

процесу імпорту і забезпечує успішне завершення навіть при виникненні тимчасових збоїв.

На блок-схемі (рисунок 2.5) показано процес вибору конфігураційного файлу, встановлення SSH-з'єднання, виконання команд імпорту та обробки результатів. Важливою частиною алгоритму є перевірка успішності кожного кроку і забезпечення повторних спроб у разі невдачі.

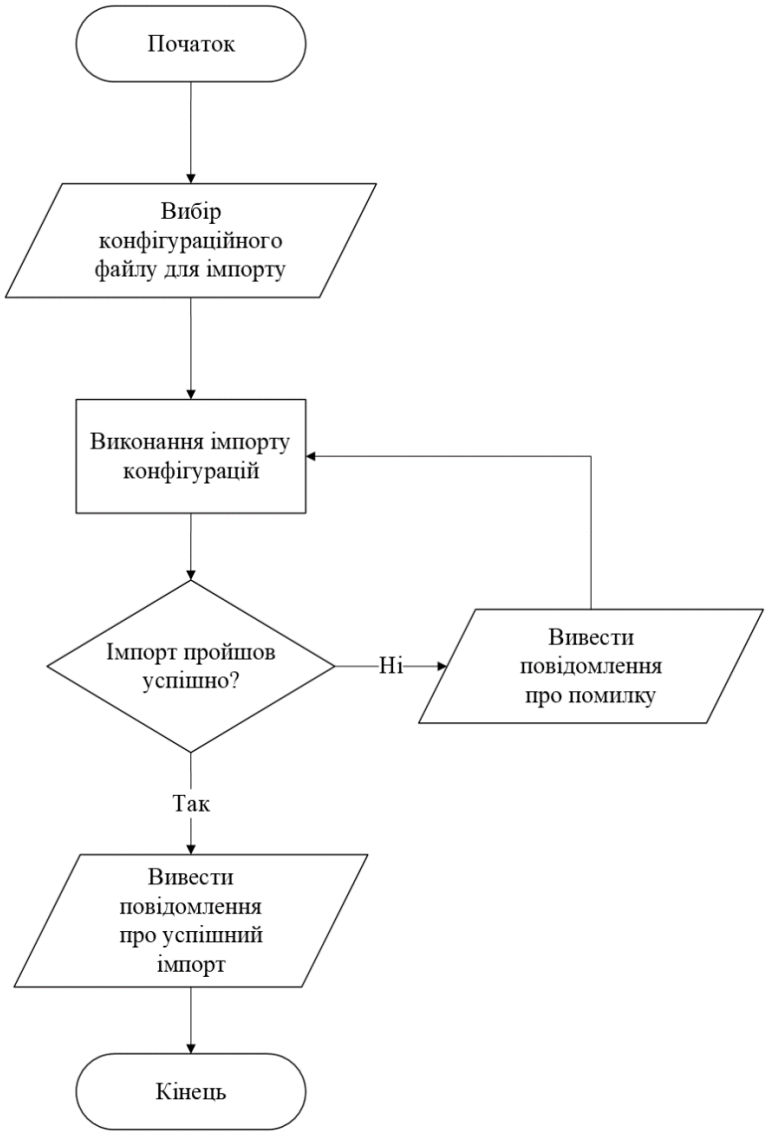

Рисунок 2.5 - Блок-схема алгоритму імпорту конфігурацій

Цей алгоритм забезпечує зручний та ефективний спосіб імпорту конфігураційних даних на пристрої MikroTik, що значно спрощує процес відновлення та оновлення мережевих налаштувань. Завдяки автоматизації

імпорту користувачі можуть бути впевнені в надійності та безпеці своїх мережевих конфігурацій.

2.4.5. Алгоритм редагування конфігурацій

Алгоритм забезпечує редагування конфігураційних файлів перед їх імпортом на пристрій MikroTik, що дозволяє користувачам коректувати та оновлювати налаштування мережевого обладнання відповідно до актуальних потреб. Процес починається з вибору користувачем конфігураційного файлу, який потрібно редагувати.

Після вибору файлу алгоритм зчитує його вміст і відображає у зручному графічному інтерфейсі для редагування. Користувач бачить налаштування та параметри, які містяться у файлі, у зрозумілому форматі.

Користувач може змінювати вміст файлу за допомогою графічного інтерфейсу, додаючи, видаляючи або змінюючи параметри конфігурації. Наприклад, можна змінити IP-адреси, налаштування файрволу, NAT, фільтри, черги, DNS, скрипти, розкладники та інші параметри, необхідні для правильного функціонування мережевого обладнання.

Після внесення змін користувач зберігає зміни у конфігураційному файлі. Це відбувається шляхом натискання на відповідну кнопку або вибору опції "Зберегти" у графічному інтерфейсі. Алгоритм перевіряє коректність збереження даних і виводить повідомлення про успіх у разі успішного збереження змін. Це повідомлення підтверджує, що всі зміни були успішно застосовані та збережені у файлі.

Якщо збереження змін не вдалося, виводиться повідомлення про помилку. Це може бути через проблеми з доступом до файлу, помилки у форматі конфігураційних даних або інші технічні труднощі. Користувачеві пропонується виправити помилки і повторити спробу збереження.

На блок-схемі (рисунок 2.6) показано процес вибору конфігураційного файлу, зчитування та відображення його вмісту, редагування параметрів через графічний інтерфейс, збереження змін та обробки результатів.

Рисунок 2.6 - Блок-схема алгоритму редагування конфігурацій

Цей алгоритм забезпечує зручний та ефективний спосіб редагування конфігураційних даних перед їх імпортом на пристрої MikroTik, що значно спрощує процес налаштування та управління мережевим обладнанням. Завдяки можливості графічного редагування користувачі можуть легко

коректувати налаштування без необхідності ручного введення команд або використання складних скриптів, що підвищує продуктивність та знижує ймовірність помилок.

2.4.6. Розроблена архітектура

Розроблена архітектура програмного забезпечення включає комплексну взаємодію між різними алгоритмами та функціонально-логічними блоками, що забезпечує виконання ключових завдань автоматизації конфігурації мережевого обладнання MikroTik. Схему роботи якої можна представити у вигляді діаграми (рисунок 2.7.):

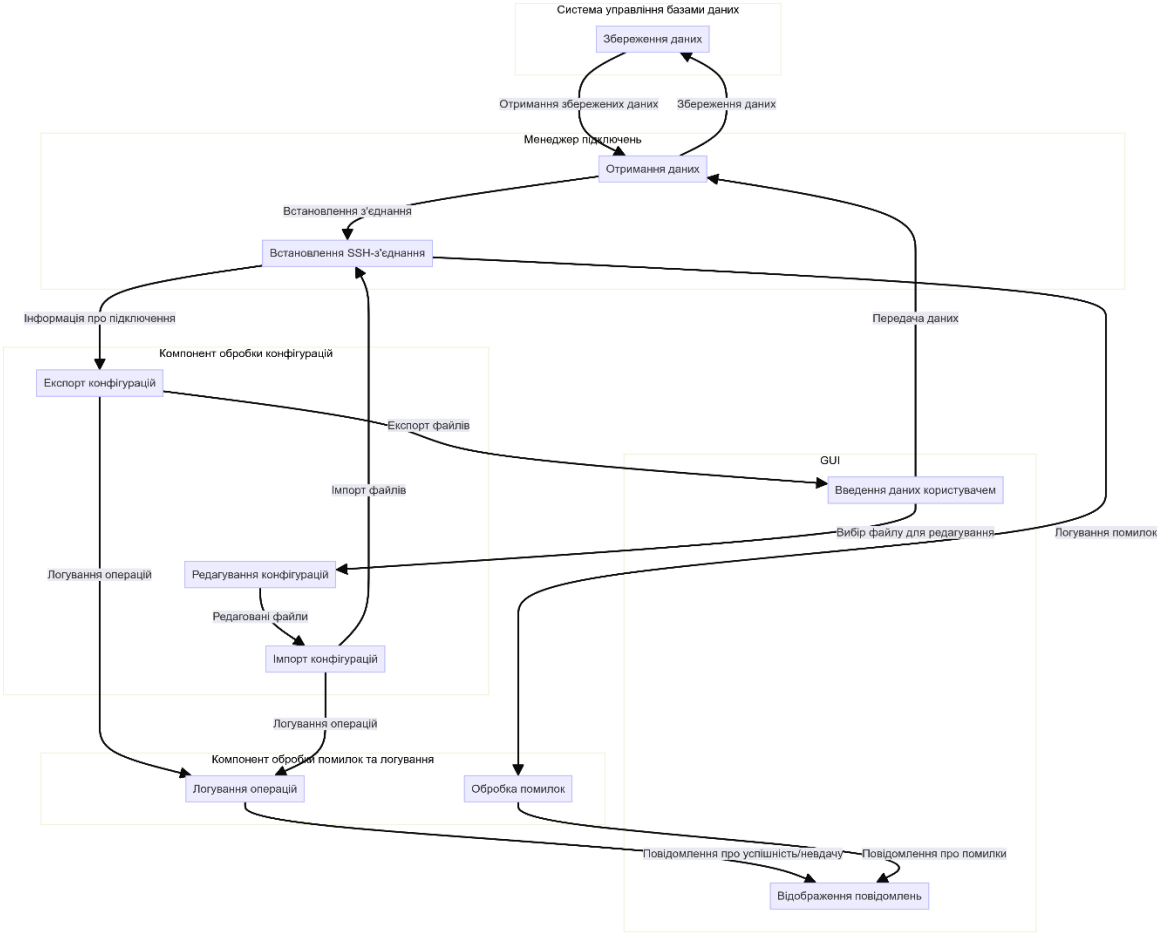

Рисунок 2.7 - Діаграма архітектури програмного модуля

В основі цього процесу лежить графічний інтерфейс користувача, який дозволяє користувачам вводити необхідні дані, такі як IP-адреса, MAC-адреса, порт, ім'я користувача та пароль. Після збору даних GUI передає їх менеджеру підключень, який відповідає за створення та управління SSH-з'єднаннями з пристроями.

Менеджер підключень перевіряє отримані відомості та ініціює встановлення з'єднання. У разі успішного підключення, інформація про сесію зберігається в системі управління базами даних, яка використовує SQLite для збереження всіх необхідних даних. Ця система також забезпечує відновлення збережених даних для подальшого використання, якщо це потрібно іншим компонентам програми.

Компонент обробки конфігурацій активується після успішного встановлення з'єднання. Він керує процесами експорту, імпорту та редагування конфігураційних файлів. Цей компонент забезпечує збереження внесених користувачем змін до конфігурацій перед їх імпортом назад на пристрій. У випадку виникнення помилок під час операцій, компонент обробки помилок та логування відіграє ключову роль, забезпечуючи запис подій та обробку помилок. Цей компонент інформує користувача через GUI про результати виконання операцій, чи це успіх, чи помилка.

Весь процес від вводу даних до завершення конфігураційних змін супроводжується систематичним логуванням і контролем за допомогою зазначених компонентів, що забезпечує цілісність та надійність роботи програмного продукту.

2.4.6. Варіанти використання програмного модулю

На діаграмі варіантів використання (рисунок 2.8) представлена взаємодія користувача з програмним модулем MikroTikConfigTool. Користувач має можливість виконувати такі основні дії: підключитися до

пристрою MikroTik, завантажити файли конфігурації та редагувати файли конфігурації. Ці дії є ключовими етапами в процесі управління конфігураціями мережевого обладнання MikroTik.

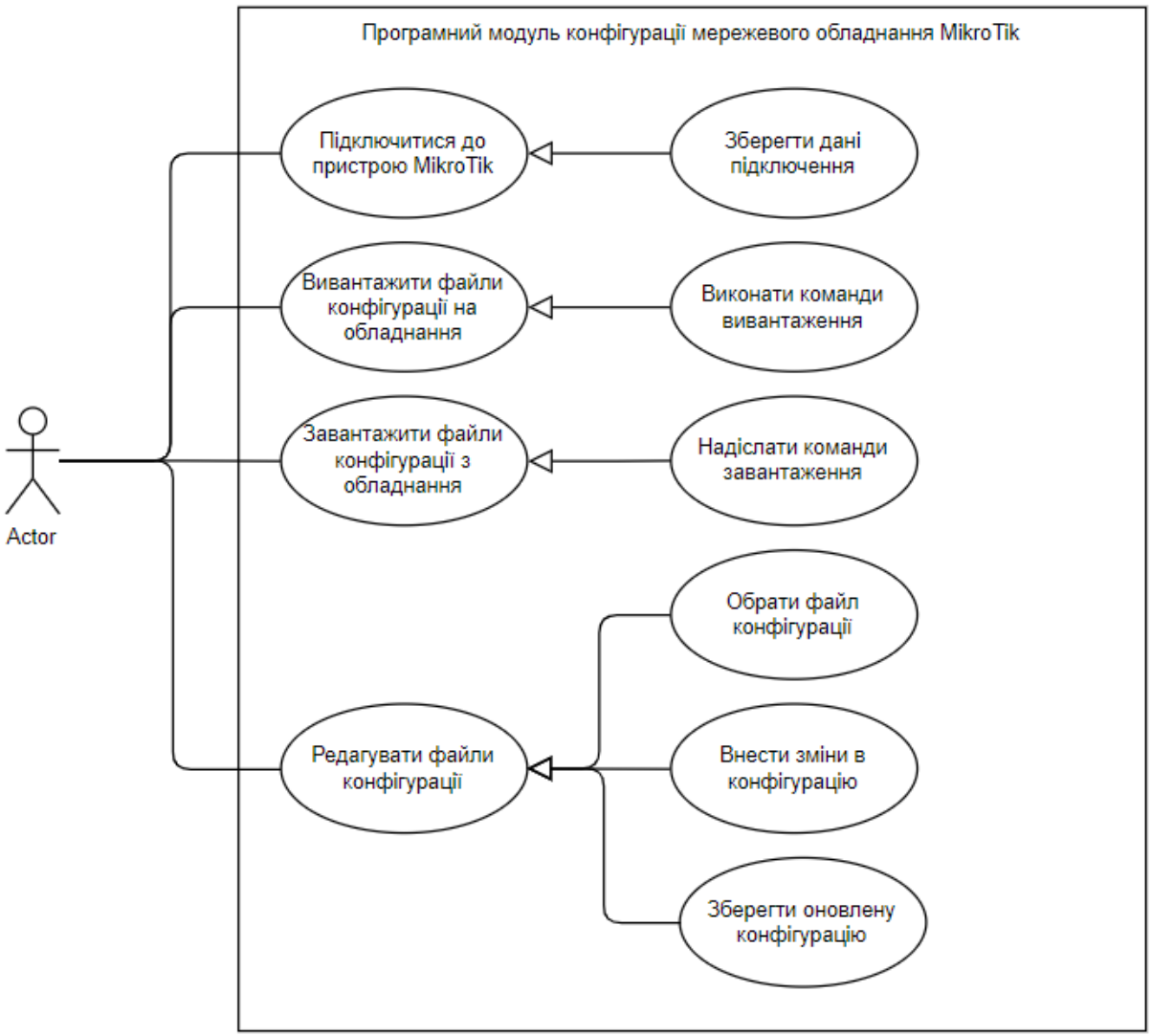

Рисунок 2.8 - Діаграма використання програмного модуля для конфігурації мережевого обладнання MikroTik

При підключенні до пристрою MikroTik користувач ініціює з'єднання з пристроєм, що дозволяє отримати доступ до його налаштувань. Після встановлення з'єднання зберігаються дані підключення, що дозволяє

користувачеві повторно підключатися до пристрою без необхідності повторного введення налаштувань. Це значно спрощує процес управління мережею, оскільки зменшує час, необхідний для підключення до різних пристроїв.

Завантаження файлів конфігурації дозволяє користувачеві отримати поточні налаштування пристрою MikroTik для подальшого аналізу або редагування. Виконання команд завантаження забезпечує автоматизований процес отримання файлів конфігурації, що значно спрощує завдання для системного адміністратора. Надсилання команд завантаження дозволяє впровадити нові або змінені конфігурації на пристрій, забезпечуючи їх відповідність поточним вимогам мережі.

Редагування файлів конфігурації включає кілька етапів: завантаження файлу конфігурації, внесення змін до його записів та збереження оновленої конфігурації. Це дозволяє користувачеві адаптувати налаштування пристрою відповідно до змінних потреб мережі. Завантаження файлу конфігурації забезпечує можливість роботи з поточними налаштуваннями, а внесення змін дозволяє оптимізувати роботу мережі, враховуючи нові вимоги або усуваючи виявлені проблеми. Збереження оновленої конфігурації гарантує, що всі внесені зміни будуть збережені та застосовані до пристрою, забезпечуючи стабільність та ефективність його роботи.

Дана діаграма демонструє основні функціональні можливості модуля та взаємодію користувача з ним, що дозволяє чітко зрозуміти процеси налаштування та управління конфігураціями мережевого обладнання MikroTik. Вона відображає, як користувач може ефективно виконувати ключові завдання, пов'язані з управлінням мережевими пристроями, використовуючи програмний модуль MikroTikConfigTool. Це включає автоматизацію багатьох процесів, що знижує ризик помилок та підвищує загальну ефективність управління мережею.

2.5 Проектування інтерфейсів користувача

Проектування інтерфейсів користувача (GUI) є важливим етапом у розробці програмного модуля автоматизованого налаштування конфігурації мережевого обладнання MikroTik. Цей етап включає створення зручного, інтуїтивно зрозумілого та функціонального інтерфейсу, що дозволяє користувачам легко взаємодіяти з програмним модулем. Користувацький інтерфейс має відповідати високим стандартам юзабіліті, забезпечуючи легкість у використанні та мінімізуючи можливість помилок під час налаштування обладнання.

Головне вікно програми (рисунок 2.9) розташовує основні елементи керування таким чином, щоб забезпечити зручність та ефективність роботи користувачів. У верхній частині вікна розташоване випадаюче меню збережених підключень, яке дозволяє швидко вибрати одне з раніше збережених підключень. Це дозволяє користувачам швидко та легко перемикатися між різними конфігураціями без необхідності вводити всі дані щоразу заново. Поруч із цим меню знаходиться кнопка для видалення вибраного підключення зі списку, що дозволяє підтримувати актуальність і чистоту списку підключень, забезпечуючи, що непотрібні або застарілі підключення не будуть заважати роботі.

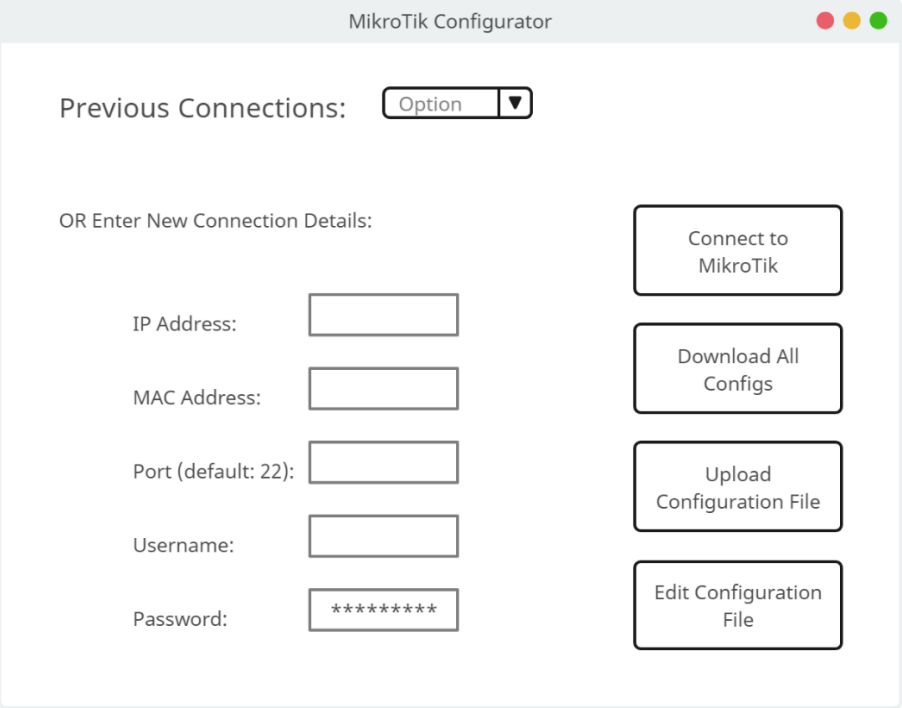

Рисунок 2.9 - Головне вікно програми

У середній частині вікна розміщені поля для введення даних підключення, які включають поля для введення IP-адреси, MAC-адреси, порту (за замовчуванням 22), імені користувача та пароля. Всі ці поля розташовані по центру вікна в стовпчик, що забезпечує зручність введення даних. Така організація інтерфейсу дозволяє користувачам швидко орієнтуватися та вводити необхідні дані без зайвих зусиль. Під полями введення даних розташовані кнопки керування, які активуються в залежності від стану підключення. Кнопка для підключення до пристрою MikroTik активується після заповнення всіх необхідних полів для підключення. Це дозволяє забезпечити, що всі необхідні дані введені правильно перед тим, як користувач зможе спробувати встановити з'єднання.

Після успішного підключення до пристрою активуються кнопки для завантаження всіх конфігурацій та для завантаження конфігураційного файлу на пристрій. Це забезпечує швидкий доступ до всіх необхідних функцій для управління та налаштування обладнання. Крім того, тут розміщена кнопка для

редагування конфігураційного файлу, яка відкриває відповідне вікно редагування. Це дозволяє користувачам безпосередньо змінювати налаштування конфігураційного файлу, забезпечуючи гнучкість і контроль над налаштуваннями.

Вікно редагування конфігурацій (рисунок 2.10) надає користувачам можливість вносити зміни у конфігураційні файли перед їх імпортом на пристрій MikroTik. Це є важливою функцією, оскільки дозволяє користувачам адаптувати конфігурацію під свої конкретні потреби і вимоги до мережевого обладнання.

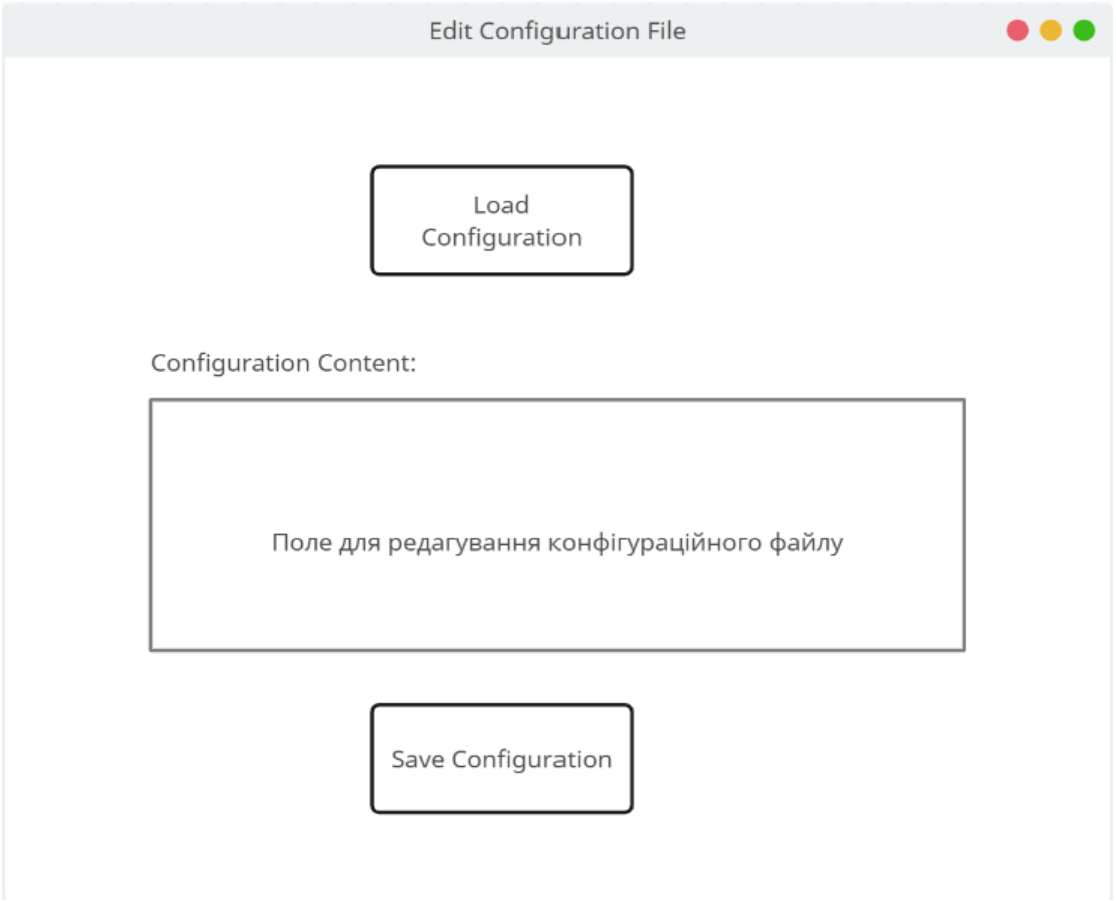

Рисунок 2.10 - Вікно редагування конфігурацій

У верхній частині цього вікна розташований заголовок "Edit Configuration File", який вирівняний по центру, що забезпечує зручність та зрозумілість для користувача. Заголовок одразу привертає увагу і дає зрозуміти, що саме в цьому вікні можна редагувати конфігураційні файли.

Поруч із заголовком розташована кнопка для завантаження конфігураційного файлу з локального комп'ютера, що дозволяє користувачам вибрати потрібний файл для редагування. Ця кнопка спрощує процес імпорту файлів і мінімізує можливість помилок при введенні даних.

Основну частину вікна займає текстове поле для редагування конфігураційного файлу. Це поле відображає вміст вибраного конфігураційного файлу та дозволяє користувачам редагувати його безпосередньо у вікні програми. Текстове поле достатньо велике для зручного перегляду і редагування конфігураційних даних, що дозволяє користувачам з легкістю вносити зміни.

Під текстовим полем розташовані кнопки для збереження змін у конфігураційному файлі та для збереження відредагованого файлу на локальний комп'ютер під новим іменем або в іншому місці. Кнопка "Save" дозволяє швидко зберегти зміни у вже існуючий файл, тоді як кнопка "Save As" надає можливість зберегти файл у новому місці або під новим іменем, що може бути корисно для створення резервних копій або збереження різних версій конфігурації.

Дизайн головного вікна та вікна редагування конфігурацій забезпечує простий та інтуїтивно зрозумілий інтерфейс для взаємодії з користувачем. Усі основні функції програмного модуля легко доступні, а введення даних і виконання основних операцій не вимагають зайвих зусиль. Використання зрозумілих елементів керування, таких як поля введення, випадаючі меню та кнопки, дозволяє швидко навчитися роботі з програмою та ефективно виконувати необхідні завдання з налаштування та управління конфігураціями мережевого обладнання MikroTik.

2.6. Реалізація Інтерфейсу Користувача

Реалізація інтерфейсу користувача в програмі починається зі створення основного вікна, яке буде місцем основної взаємодії користувача з програмою. Спочатку потрібно імпортувати необхідні модулі, такі як tkinter та ttk з пакету tkinter, а також messagebox і filedialog для додаткових функцій.

Наступним кроком є створення класу MikroTikApp, який буде відповідати за ініціалізацію та налаштування головного вікна. В методі __init__ необхідно встановити заголовок вікна за допомогою методу self.root.title("MikroTik Configurator") та розміри вікна за допомогою методу self.root.geometry("450x400"). Далі потрібно викликати метод create_widgets, який створюватиме всі необхідні віджети. Також потрібно ініціалізувати змінні для з'єднання з базою даних та створити таблицю для збереження підключень.

Для створення віджетів використовується метод create_widgets, де спочатку створюється основний фрейм main_frame за допомогою ttk.Frame та розміщується на головному вікні за допомогою методу grid. Далі додається випадаючий список для вибору збережених підключень, який створюється за допомогою ttk.Combobox і прив’язується до події <<ComboboxSelected>> для завантаження вибраного підключення. Комбобокс налаштовується на режим "readonly" для запобігання ручного введення.

Після цього додаються текстові поля для введення нових даних підключення: IP-адреси, MAC-адреси, порту, імені користувача та пароля. Кожне текстове поле створюється за допомогою ttk.Entry і розміщується на фреймі за допомогою методу grid. Для кожного текстового поля також встановлюється початкове значення за допомогою методу insert. Наприклад, для поля IP-адреси використовується self.ip_input.insert(0, "Enter IP Address").

Наступним кроком є створення кнопок для підключення до пристрою MikroTik, завантаження всіх конфігурацій та завантаження конфігураційного

файлу. Кнопки створюються за допомогою ttk.Button і також розміщуються на фреймі за допомогою методу grid. Кнопка для підключення має прив’язану команду self.connect_to_mikrotik, яка буде викликатися при натисканні. Кнопки для завантаження та завантаження конфігураційного файлу спочатку деактивуються (state=tk.DISABLED) і будуть активовані після встановлення підключення.

Додатково створюється кнопка для редагування конфігураційного файлу, яка відкриватиме додаткове вікно. Це вікно дозволяє користувачам завантажувати, редагувати та зберігати конфігураційні файли. Ініціалізація додаткового вікна виконується в методі open_additional_window. У цьому методі створюється нове вікно за допомогою tk.Toplevel, встановлюється його заголовок і розміри за допомогою методів title та geometry.

Нижче можна побачити, як виглядає вікно редагування конфігураційного файлу (рисунок 2.11):

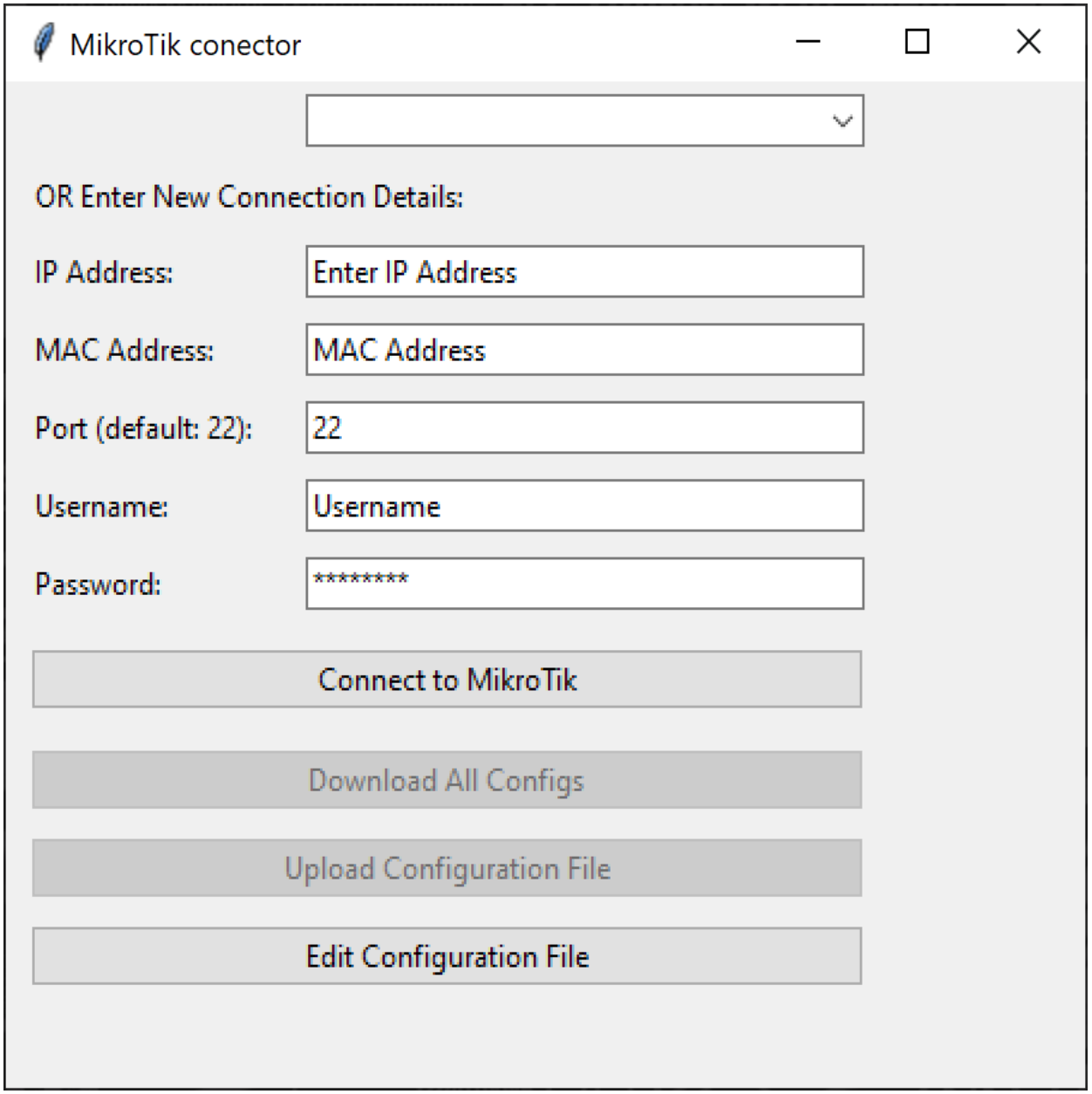

Рисунок 2.11 - Реалізоване головне вікно програми

Далі створюються віджети для завантаження, редагування та збереження конфігураційного файлу. Для завантаження файлу використовується кнопка tk.Button, яка викликає метод self.load_config. Для редагування конфігурацій створюються текстові поля (tk.Entry) з мітками (tk.Label), де користувач може ввести нові значення. Наприклад, для редагування діапазону IP-адрес створюється мітка self.pool_range_label = tk.Label(additional_window, text="/ip pool:") та текстове поле self.new_pool_entry = tk.Entry(additional_window,

width=70). Кнопка збереження конфігурації self.save_button спочатку деактивована і буде активована після завантаження файлу конфігурації.

Також необхідно додати методи для підключення до пристрою MikroTik, завантаження та завантаження конфігурацій, а також обробки даних конфігураційного файлу. Ці методи будуть викликатися відповідними кнопками та забезпечуватимуть основний функціонал програми.

Нижче можна побачити, як виглядає реалізоване головне вікно програми (рисунок 2.12):

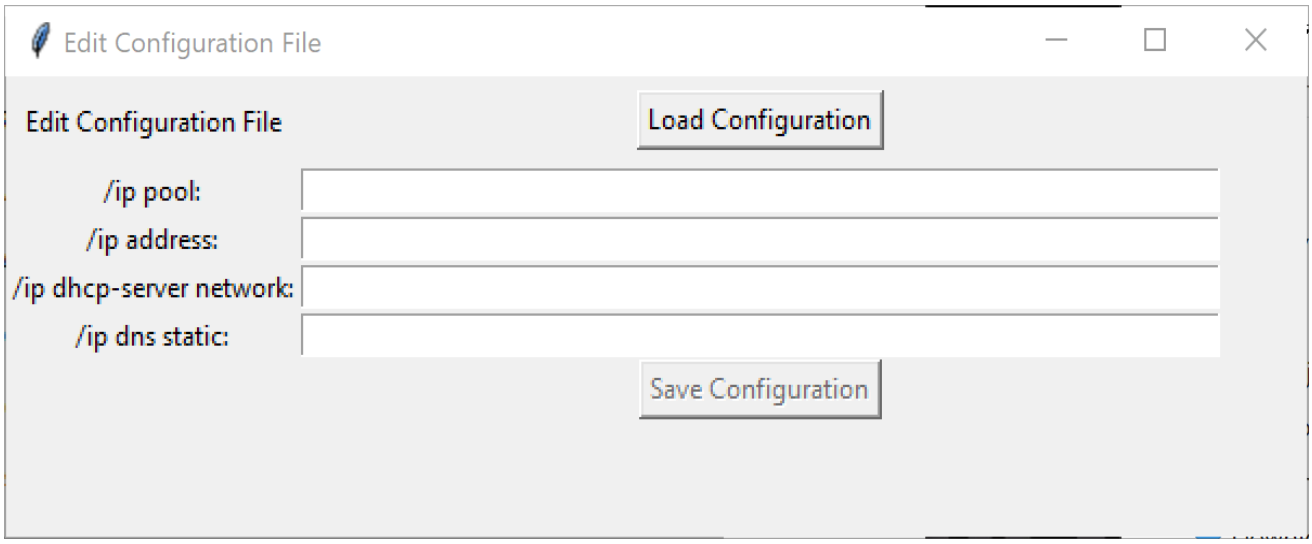

Рисунок 2.12 - Реалізоване вікно редагування конфігурацій

2.7. Написання коду згідно з розробленими алгоритмами

2.7.1. Алгоритм підключення до пристрою MikroTik

Алгоритм підключення до пристрою MikroTik забезпечує встановлення SSH-з'єднання з пристроєм на основі введених користувачем даних. Беручи до уваги всі розроблені елементи алгоритму та враховую сві необхідні вимоги використовуючи розроблену діаграму алгоритму (рисунок 2.13) почнемо його реалізацію.

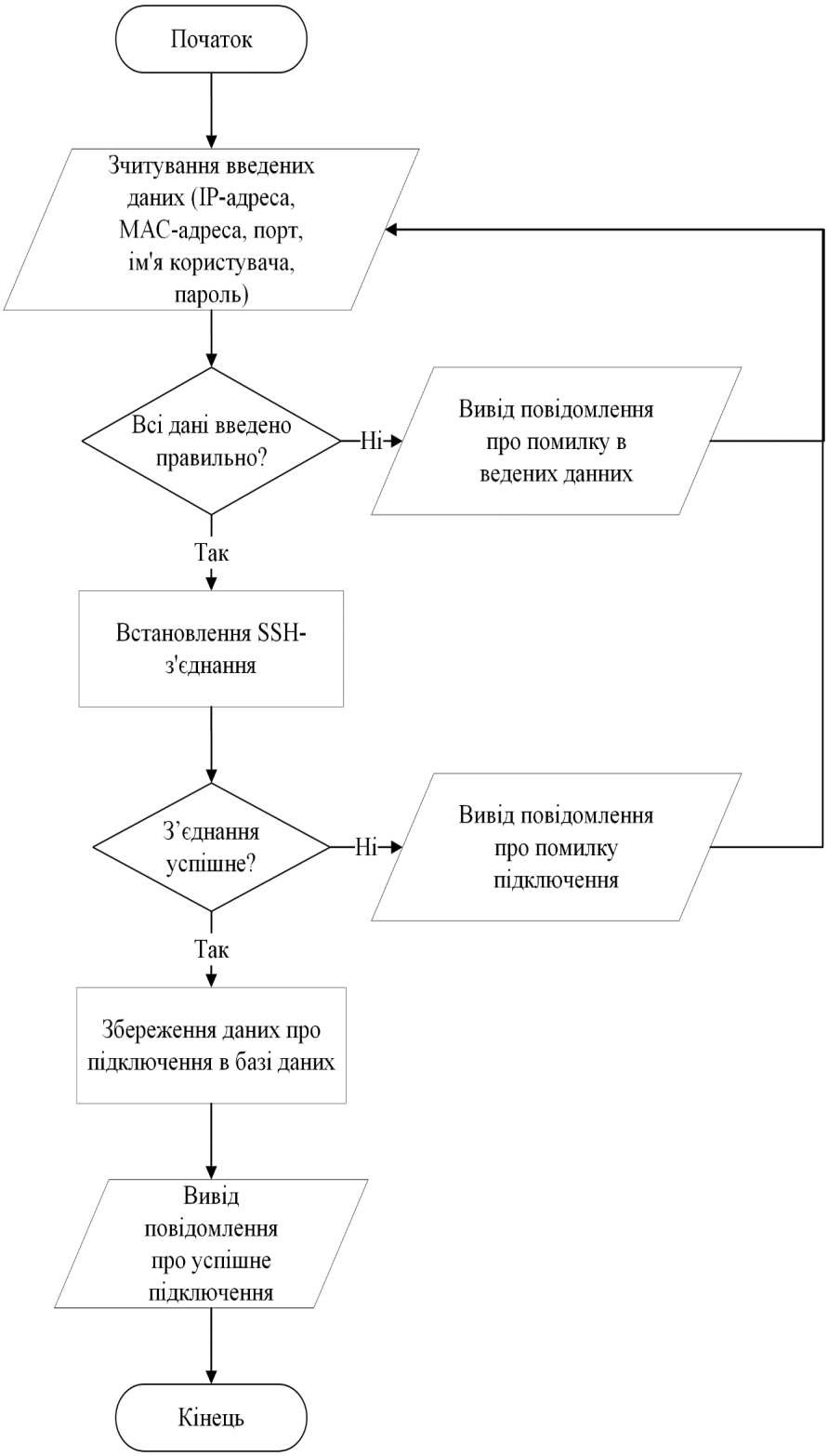

Рисунок 2.13 - Блок-схема алгоритму підключення до пристрою MikroTik

Спочатку у створеному методі create_widgets створюються поля для введення IP-адреси, MAC-адреси, порту, імені користувача та паролю за

допомогою ttk.Entry. Для кожного поля створюється відповідний текстовий лейбл за допомогою ttk.Label.

Поля введення та їхні лейбли розміщуються на основному фреймі main_frame з використанням методу grid, що дозволяє розташовувати елементи на вікні яке буде бачити користувач у вигляді сітки.

Користувач заповнює кожне поле відповідними даними: IP-адреса, MAC-адреса, порт, ім'я користувача та пароль. Значення зберігаються у відповідних змінних класу (self.ip_input, self.mac_input, self.port_input, self.username_input, self.password_input).

Після заповнення полів користувач натискає кнопку "Connect to MikroTik", яка була створена за допомогою ttk.Button та прив'язується до методу connect_to_mikrotik.

При натисканні кнопки викликається метод connect_to_mikrotik, який зчитує значення з усіх полів введення за допомогою методу get(), обрізає зайві пробіли за допомогою методу strip() та зберігає їх у змінні: ip_address, mac_address, port, username, password.

Якщо будь-яке з обов'язкових полів (IP-адреса, MAC-адреса, порт, ім'я користувача) порожнє, виводиться повідомлення про помилку за допомогою messagebox.showwarning з проханням заповнити всі поля, окрім паролю.

Якщо всі поля заповнені, створюється словник device з параметрами підключення. Значення "device_type" по замовченню бузе заповнене даними про тип дивайсу, в нашому випадку mikrotik_routeros. Далі заповнюється значення "host" в чке передається змінна ip_address яку ми отримуємо з даних які вводить користувач. По аналогії заповнюються значення для "username" - username, "password" - password значення паролю може бути не заповнене. Та "port" в яке передається int(port)

Далі використовується ConnectHandler з бібліотеки Netmiko для встановлення SSH-з'єднання з пристроєм "self.netmiko_connection = ConnectHandler(**device) "

Якщо підключення встановлено успішно, виводиться повідомлення про успішне підключення за допомогою messagebox.showinfo. У разі виникнення помилки під час підключення, виводиться повідомлення про помилку за допомогою messagebox.showerror з описом проблеми.

2.7.2. Алгоритм експорту конфігурацій

Використовуючи розроблену блок-схему (рисунок 2.14), почнемо реалізацію алгоритму експорту конфігурації з роутера MikroTik на пристрій, з якого ініціалізоване підключення. Алгоритм надсилає команди експорту конфігурацій на мережеве обладнання для створення файлів конфігурації для подальшого їх завантаження, зокрема для різних компонентів конфігурації.

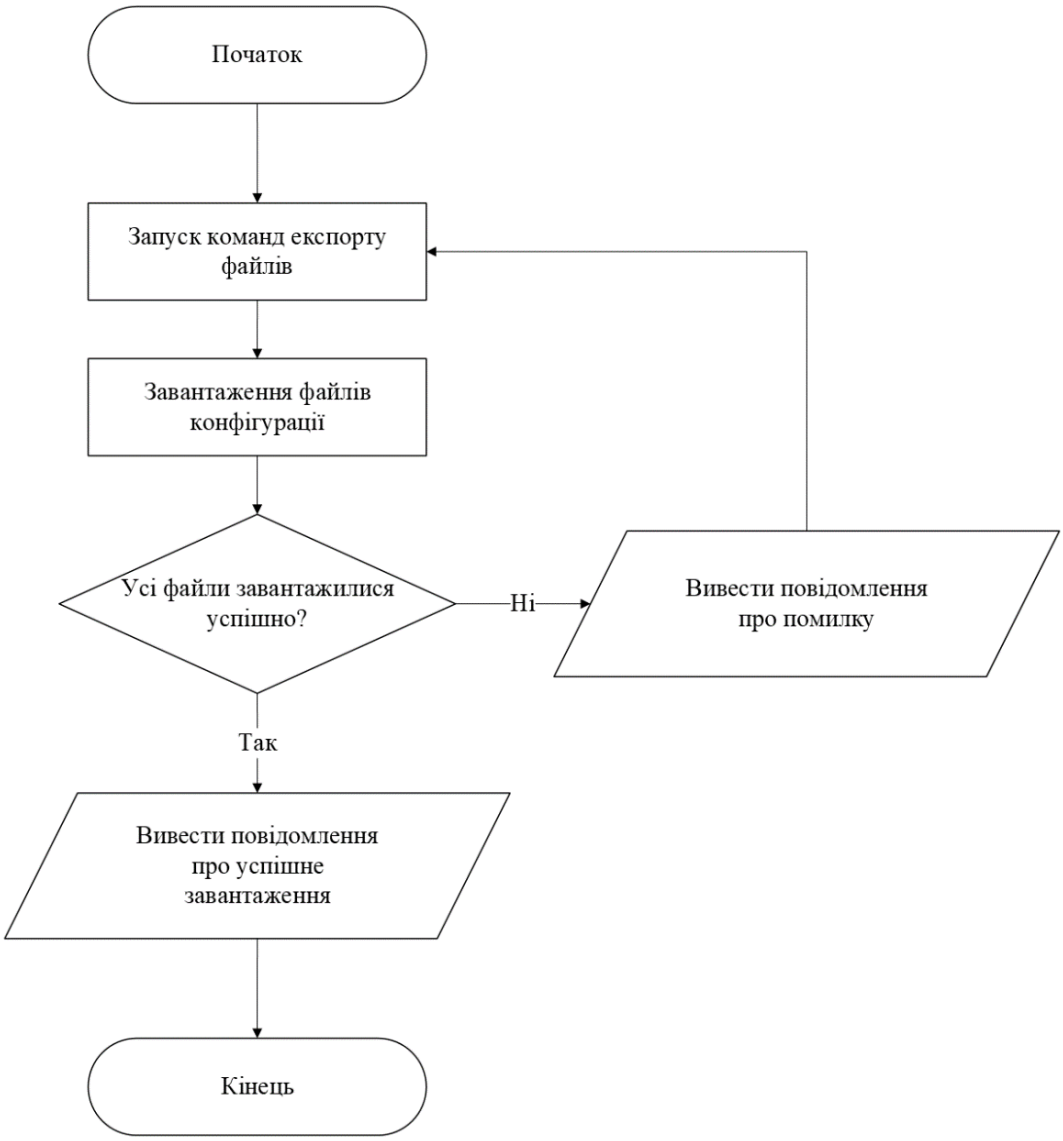

Рисунок 2.14 - Блок-схема алгоритму експорту конфігурацій

Команди експорту для різних компонентів конфігурації, такі як IP-адреси, мітки файрволу, NAT, фільтри, черги, DNS, скрипти, розкладники, налаштування електронної пошти, списки адрес, маршрути, мережі DHCP-сервера, типи черг, дерево черг, налаштування інтерфейсів, пули IP-адрес, профілі PPP, логування, та загальний експорт конфігурації, зберігаються в списку команд. Алгоритм додає дату до імені кожного файлу, що забезпечує унікальність та запобігає перезапису існуючих файлів.

Після виконання команди експорту конфігураційний файл завантажується на локальний комп'ютер. Алгоритм перевіряє наявність файлу

на пристрої та завантажує його за допомогою SFTP. Якщо файл не знайдено, алгоритм чекає деякий час і повторює спробу. Цей процес повторюється до успішного завершення або до досягнення встановленої кількості спроб.

Алгоритм перевіряє, чи файл успішно завантажений. Якщо файл не завантажений після кількох спроб, виводиться повідомлення про помилку. Якщо всі файли успішно завантажені, користувачу виводиться повідомлення про успіх. Після завершення роботи алгоритм закриває всі підключення і ресурси, що використовувалися під час виконання.

Приклад команд експорту конфігурацій:

- ip address export file=ip.rsc - експортує налаштування IP-адрес;
- ip firewall mangle export file=mangle.rsc - експортує налаштування mangle-фільтрів;
- ip firewall nat export file=nat.rsc - експортує налаштування NAT;
- ip firewall filter export file=filter.rsc - експортує налаштування фільтрів брандмауера;
- queue simple export file=simple.rsc - експортує налаштування простих черг;
- ip dns export file=dns.rsc - експортує налаштування DNS;
- system script export file=script.rsc - експортує скрипти системи;
- system scheduler export file=scheduler.rsc - експортує налаштування планувальника завдань;
- tool e-mail export file=email.rsc - експортує налаштування інструментів електронної пошти;
- ip firewall address-list export file=address-list.rsc - експортує списки адрес брандмауера;
- ip route export file=route.rsc - експортує маршрути;
- ip dhcp-server network export file=network.rsc - експортує налаштування DHCP-серверів;
- queue type export file=type.rsc - експортує типи черг;

- queue tree export file=tree.rsc - експортує налаштування черг за деревом;
- interface ethernet export file=ethernet.rsc - експортує налаштування Ethernet-інтерфейсів;
- ip pool export file=pool.rsc - експортує налаштування пулів IP-адрес;
- ppp profile export file=profile.rsc - експортує профілі PPP;
- system logging export file=log.rsc - експортує налаштування журналів системи;
- export file=config_backup.rsc - експортує всі налаштування пристрою в один файл.

2.7.3. Алгоритм імпорту конфігурацій

Алгоритм забезпечує імпорт конфігураційних файлів на пристрій MikroTik, дозволяючи користувачеві легко відновлювати налаштування або оновлювати конфігурацію мережевого обладнання. Його реалізація виконана на основі розробленої блок-схеми (рисунок 2.15).

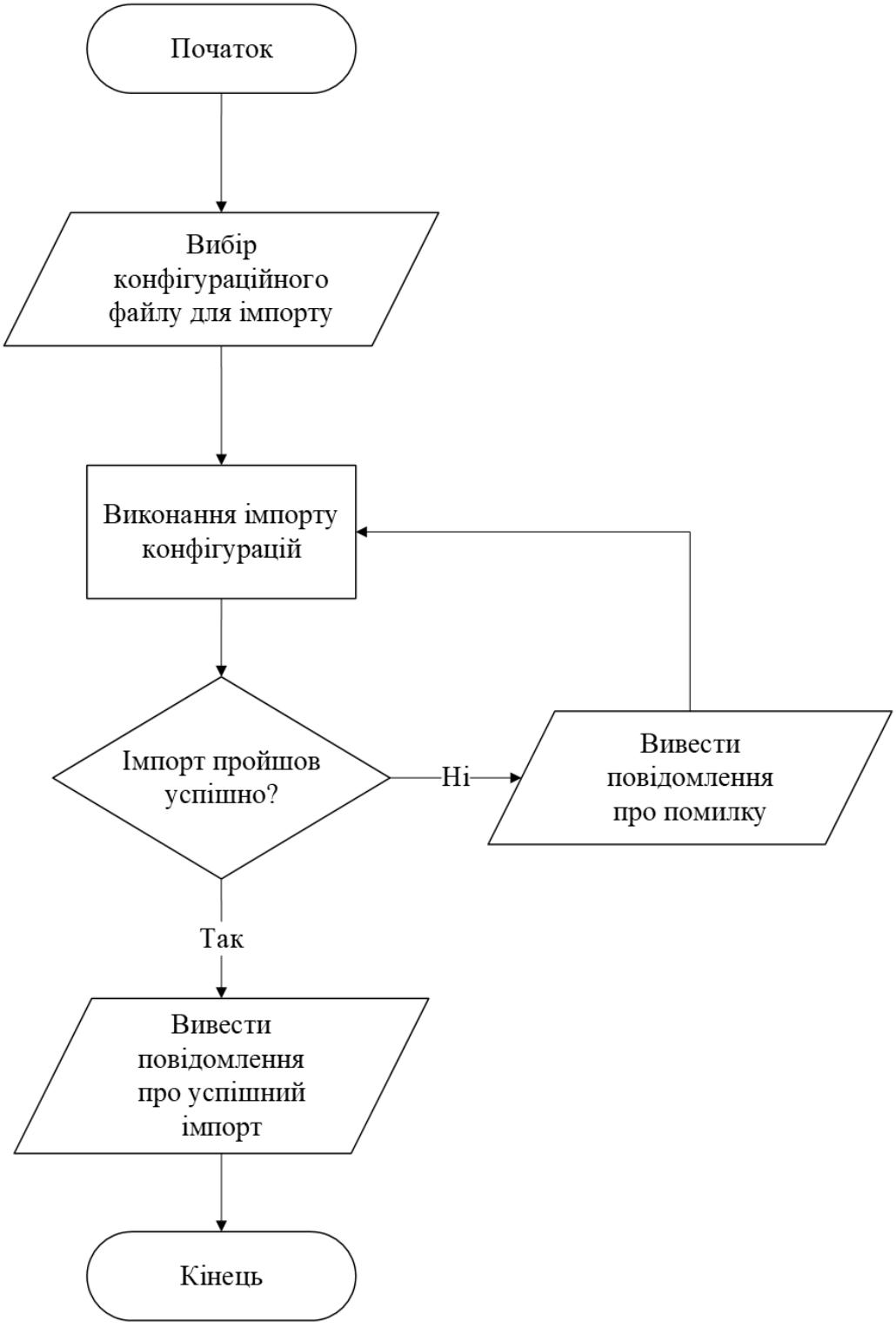

Рисунок 2.15 - Блок-схема алгоритму імпорту конфігурацій

Процес починається з того, що користувач вибирає конфігураційний файл для імпорту. Використовується діалогове вікно для вибору файлу конфігурації, що дозволяє користувачеві вибрати файл з розширенням .rsc.

Після вибору файлу алгоритм зчитує його вміст і виконує команди імпорту конфігурацій.

Команди зчитуються з вибраного файлу та зберігаються у вигляді списку команд. Далі ці команди по черзі виконуються на пристрої. Після кожної виконаної команди алгоритм перевіряє успішність імпорту. У разі успішного імпорту користувач отримує повідомлення про успіх. Якщо імпорт не вдається, виводиться повідомлення про помилку, і алгоритм повторює спробу імпорту.

Алгоритм використовує методи для надсилання команд на пристрій і перевірки їх виконання. У разі помилки команда повторюється кілька разів до успішного виконання або до досягнення встановленої кількості спроб. Після успішного виконання всіх команд користувач отримує повідомлення про успіх.

Приклад команд з файлу конфігурації:

/ip address add address=192.168.88.1/24 interface=ether1;

/ip dns set servers=8.8.8.8,8.8.4.4;

/system identity set name=MikroTikRouter.

Ці команди додають IP-адресу до інтерфейсу ether1, встановлюють DNS-сервери та змінюють ім'я пристрою.

Алгоритм завершується, коли всі команди імпортовано успішно або після вичерпання встановленої кількості спроб. У разі успіху користувач отримує відповідне повідомлення, а у разі невдачі алгоритм надає детальну інформацію про помилки, що виникли, та пропонує варіанти їх виправлення.

2.7.4. Алгоритм редагування конфігурації

Враховуючи реалізовану блок-схему (рисунок 2.16), алгоритм редагування конфігураційних файлів забезпечує можливість користувачам вносити зміни до конфігураційних файлів перед їх імпортом на пристрій

MikroTik. Розглянемо процес реалізації алгоритму з урахуванням всіх умов його роботи та функціоналу на прикладі конфігураційного файлу.

Рисунок 2.16 - Блок-схема алгоритму редагування конфігурацій

Запуск програми MikroTikApp починається з відображення основного інтерфейсу на екрані. Після цього користувач натискає кнопку "Edit

Configuration File", що відкриває додаткове вікно для редагування конфігураційних файлів. Користувач натискає кнопку "Load Configuration", після чого відкривається діалогове вікно для вибору файлу. Користувач вибирає необхідний конфігураційний файл, наприклад, config_backup.rsc.

Програма зчитує вміст файлу config_backup.rsc, зберігаючи його у змінну self.config_lines, яка містить список рядків. Вміст файлу відображається у формі для редагування. Програма шукає рядки, що містять ключові слова, такі як /ip pool, /ip address, /ip dhcp-server network, /ip dns static. Якщо ключове слово знайдено, програма відображає наступний рядок у відповідному полі вводу.

Наприклад, конфігураційний файл містить:

/ip pool

add name=dhcp_pool ranges=192.168.88.10-192.168.88.20

/ip address

add address=192.168.88.1/24 interface=bridge1

/ip dhcp-server network

add address=192.168.88.0/24 gateway=192.168.88.1

/ip dns static

add name=example.com address=192.168.88.2

Програма знайде рядок /ip pool та відобразить наступний рядок add name=dhcp_pool ranges=192.168.88.10-192.168.88.20 у полі вводу для /ip pool.

Користувач вносить зміни у відповідні поля. Наприклад, змінює значення поля /ip pool на add name=dhcp_pool ranges=192.168.88.10-192.168.88.30.

Після внесення змін користувач натискає кнопку "Save Configuration". Програма зчитує нові значення з полів вводу і знаходить відповідні рядки у списку self.config_lines, замінюючи їх новими значеннями. Як приклад, програма знаходить індекс рядка /ip pool у файлі та змінює наступний рядок на add name=dhcp_pool ranges=192.168.88.10-192.168.88.30.

Програма відкриває діалогове вікно для вибору місця збереження нового файлу. Користувач може зберегти змінений файл під новим ім'ям або у тому ж місці. Після успішного збереження змін програма виводить повідомлення про успіх.

2.8 Проведення тестування основних функцій модуля для перевірки їх відповідності вимогам

Проведення тестування основних функцій модуля є важливим етапом розробки програмного забезпечення для забезпечення його коректної роботи та відповідності вимогам. Тестування включає декілька кроків, таких як розробка тестових сценаріїв, виконання тестів та аналіз результатів. Для кожної основної функції модуля були розроблені відповідні тестові сценарії.

Тестування підключення до пристрою MikroTik починається з перевірки правильності підключення з коректними параметрами. Сценарій включає введення правильних даних (IP-адреса, MAC-адреса, порт, ім'я користувача, пароль) і очікує успішного підключення, збереження даних про підключення в базу даних, активування кнопок завантаження та вивантаження конфігурацій. Також тестується підключення з некоректними параметрами, коли введення неправильних даних (наприклад, невірний IP-адрес або пароль) призводить до повідомлення про помилку підключення.

Тестування експорту конфігурацій включає перевірку успішного експорту конфігурацій та завантаження їх на локальний комп'ютер. Сценарій включає встановлення з'єднання з пристроєм MikroTik, виконання команд експорту конфігурацій, перевірку створення файлів конфігурацій на пристрої та їх завантаження на локальний комп'ютер. У разі невдалого експорту конфігурацій сценарій перевіряє повідомлення про помилку і повторні спроби експорту. Команди для експорту включають різні аспекти налаштувань

мережевого обладнання MikroTik, такі як IP-адреси, налаштування брандмауера, черги, налаштування DHCP та інші.

Тестування імпорту конфігурацій перевіряє успішний імпорт конфігурацій на пристрій MikroTik та обробку помилок під час імпорту. Сценарій включає вибір конфігураційного файлу для імпорту, встановлення з'єднання з пристроєм, виконання команд імпорту конфігурацій, перевірку успішності імпорту та повідомлення про успіх. У разі невдалого імпорту сценарій перевіряє повідомлення про помилку і повторні спроби імпорту.

Тестування редагування конфігурацій перевіряє можливість успішного редагування конфігураційного файлу перед імпортом на пристрій MikroTik. Сценарій включає вибір конфігураційного файлу для редагування, зчитування вмісту файлу і відображення його у формі для редагування, редагування файлу через графічний інтерфейс, збереження змін у файлі і перевірку успішного збереження змін. У разі невдалого редагування сценарій перевіряє повідомлення про помилку збереження.

Проведення тестування основних функцій модуля підтвердило відповідність модуля вимогам та його готовність до реального використання для налаштування мережевого обладнання MikroTik. Всі тестові сценарії були виконані з очікуваними результатами. Тестування показало, що основні функції модуля працюють коректно і відповідають вимогам. Під час тестування були виявлені деякі незначні помилки, які були виправлені в процесі тестування. В результаті, модуль був успішно протестований і готовий до використання. Детальний опис коду кожного тесту наведено в Додаткок Б.

Таким чином, проведення тестування підтвердило відповідність модуля вимогам та його готовність до реального використання для налаштування мережевого обладнання MikroTik. Тестування дозволило впевнитися в коректній роботі основних функцій модуля, що є важливим для забезпечення надійності та ефективності програмного продукту.

2.9 Виявлення та виправлення помилок та недоліків

Після проведення початкового тестування основних функцій модуля, було виявлено кілька помилок та недоліків, які потребували виправлення. Виявлення та виправлення цих проблем є важливим етапом у процесі розробки програмного забезпечення, оскільки вони забезпечують підвищення стабільності, надійності та продуктивності системи.

Виявлені помилки та недоліки:

1. Проблеми з підключенням:

- при підключенні до пристроїв MikroTik виникали помилки, що призводило до неможливості виконання подальших операцій;
- помилка обробки виключень призводила до невизначеного стану програми після невдалої спроби підключення.

2. Помилки експорту конфігурацій:

- команди експорту конфігурацій не завжди виконувалися коректно, що призводило до неповних або відсутніх файлів конфігурацій;
- проблеми з повторними спробами завантаження файлів, якщо початкове завантаження не вдалося.

3. Проблеми з імпортом конфігурацій:

- неправильне оброблення команд імпорту конфігурацій, що призводило до невдалого завантаження конфігурацій на пристрій;
- недостатня перевірка наявності та правильності конфігураційних файлів перед імпортом.

4. Редагування конфігураційних файлів:

- некоректне відображення та редагування конфігураційних даних у графічному інтерфейсі;
- проблеми з збереженням змін у конфігураційних файлах.

Процес виправлення помилок та недоліків:

1. Виправлення проблем з підключенням:

- було додано додаткову перевірку введених даних на коректність перед встановленням з'єднання;

- покращено обробку виключень для забезпечення стабільного стану програми після невдалої спроби підключення;

- додано додаткові повідомлення для користувача, що пояснюють причину невдачі підключення.

2. Виправлення помилок експорту конфігурацій:

- переглянуто команди експорту для забезпечення їх коректного виконання;

- додано повторні спроби завантаження файлів з більш надійною обробкою помилок та додатковими затримками.

3. Виправлення проблем з імпортом конфігурацій:

- переглянуто метод імпорту для забезпечення коректного виконання команд та перевірки файлів перед імпортом;

- додано повідомлення про успішний імпорт та обробка помилок.

4. Виправлення редагування конфігураційних файлів:

- переглянуто методи зчитування та відображення конфігураційних даних у графічному інтерфейсі;

- забезпечено коректне збереження змін у конфігураційних файлах з додатковою перевіркою наявності файлів та коректності даних.

Процес виявлення та виправлення помилок був проведений ретельно для забезпечення коректної роботи модуля та відповідності вимогам. Після виправлення всіх виявлених проблем, модуль був повторно протестований для перевірки коректності внесених змін. Результати тестування показали, що всі функції працюють належним чином, і модуль готовий до використання.

2.10 Створення виконуваного файлу

Створення виконуваного файлу для програмного модуля автоматизованого налаштування конфігурації мережевого обладнання MikroTik є ключовим етапом у завершенні розробки. Цей процес включає

компіляцію коду, підготовку інсталяційного пакету та тестування виконуваного файлу на сумісність та функціональність. Для компіляції Python-коду до виконуваного файлу використовуються інструменти, такі як PyInstaller, cx_Freeze або py2exe. Вибір інструменту залежить від вимог до платформи та додаткових функцій, які можуть бути необхідні для пакування залежностей.

Перед початком компіляції необхідно налаштувати середовище розробки. Це включає встановлення всіх необхідних бібліотек та залежностей у середовищі розробки. Використання віртуальних середовищ дозволяє ізолювати проект та уникнути конфліктів між бібліотеками. Важливо переконатися, що всі компоненти проекту працюють коректно в межах ізольованого середовища, що забезпечує стабільність та передбачуваність процесу компіляції.

Наступним кроком є конфігурація інструменту для компіляції, наприклад, PyInstaller. Створюється конфігураційний файл, який містить інформацію про основний файл проекту, додаткові файли та залежності, які повинні бути включені до виконуваного файлу. У конфігураційному файлі також вказуються параметри налаштування виконуваного файлу, такі як ім'я файлу, іконки та інші ресурси. Після налаштування конфігурації проект компілюється за допомогою відповідної команди, що дозволяє створити самостійний виконуваний файл, який можна запускати без необхідності встановлення додаткових компонентів.

Після створення виконуваного файлу необхідно провести його тестування на всіх підтримуваних платформах, таких як Windows, Linux та macOS. Переконайтесь, що всі функції працюють належним чином і що виконуваний файл містить всі необхідні залежності. Це важливо для забезпечення стабільної роботи програмного модуля на різних операційних системах. Тестування включає перевірку коректності виконання основних

функцій програми, стабільності роботи під навантаженням та сумісності з різними версіями операційних систем.

Після успішного тестування можна приступати до створення інсталяційного пакету. Для цього використовуються інструменти, які дозволяють створювати інсталятори для різних операційних систем. Інсталятор повинен включати всі необхідні файли та надавати простий та інтуїтивно зрозумілий процес установки для кінцевого користувача. Під час розробки інсталяційного пакету важливо врахувати різні сценарії використання та забезпечити можливість налаштування параметрів установки відповідно до потреб користувача.

2.11 Інструкція з використання програмного модуля для конфігурації мережевого обладнання MikroTik

Програмний модуль автоматизованого налаштування конфігурації мережевого обладнання MikroTik розроблений для спрощення процесу налаштування та управління мережевими пристроями. Інструкція з використання містить кроки щодо установки, налаштування та основного використання програмного модуля, забезпечуючи користувачів необхідною інформацією для ефективного застосування продукту.

Першим кроком є встановлення програмного модуля на комп'ютер користувача. Для цього необхідно завантажити інсталяційний пакет з офіційного сайту або іншого наданого джерела. Запустіть інсталятор і слідуйте інструкціям на екрані, щоб завершити процес установки. Переконайтесь, що всі необхідні компоненти були встановлені належним чином.

Після успішної установки програмного модуля необхідно виконати початкове налаштування. Запустіть програму і введіть дані для підключення до пристрою MikroTik, включаючи IP-адресу, ім'я користувача та пароль. Переконайтесь, що підключення встановлено успішно, перевіривши стан

з'єднання у відповідному розділі програми. Якщо підключення не вдалося, перевірте правильність введених даних та доступність мережевого пристрою.

Однією з основних функцій програмного модуля є автоматизація налаштування конфігурацій мережевого обладнання. Для цього скористайтесь вбудованими шаблонами конфігурацій або створіть власні конфігурації, які відповідають специфічним потребам вашої мережі. Виберіть потрібний шаблон або створіть новий, вкажіть необхідні параметри та застосуйте конфігурацію до пристрою. Програмний модуль автоматично виконає всі необхідні команди та налаштування, забезпечуючи коректну роботу обладнання.

Програмний модуль також підтримує функції експорту та імпорту конфігурацій. Для експорту існуючої конфігурації оберіть відповідну опцію в меню програми, вкажіть місце збереження файлу та збережіть конфігурацію. Це дозволить створити резервну копію налаштувань, яку можна використовувати для відновлення або застосування на іншому пристрої. Для імпорту конфігурації завантажте файл конфігурації та застосуйте його до потрібного пристрою, що значно спрощує процес налаштування та зменшує ризик помилок.

ВИСНОВКИ

Кваліфікаційна робота була спрямована на створення програмного модуля для автоматизованого налаштування мережевого обладнання MikroTik. Основна мета була досягнута — розроблено інструмент, який спрощує та оптимізує процес налаштування, забезпечуючи зручний інтерфейс для керування конфігураціями та підтримуючи крос-платформну сумісність.

Аналіз існуючих рішень. Було проведено ретельний аналіз сучасних методів налаштування мережевого обладнання MikroTik, розглянуто основні технології та методи, включаючи готові рішення та індивідуальні розробки. Виявлено ключові проблеми та виклики, з якими стикаються системні адміністратори під час налаштування обладнання MikroTik.

Розробка програмного модуля. Визначено технічне завдання, цілі та задачі розробки програмного модуля. Обрано технології та інструменти для розробки, де було віддано перевагу мові програмування Python через її простоту, гнучкість та підтримку широкого спектру бібліотек для мережевої автоматизації. Спроектовано архітектуру програмного забезпечення та визначено основні компоненти модуля, їх функціональні характеристики та взаємодію між ними.

Реалізація та тестування. Створено графічний інтерфейс користувача (GUI), який забезпечує інтуїтивно зрозумілий доступ до всіх функцій модуля. Реалізовано основні функціональні можливості, включаючи автоматизацію процесів імпорту, експорту та редагування конфігурацій. Проведено тестування програмного модуля, що підтвердило його відповідність вимогам технічного завдання.

Розроблений програмний модуль має значну практичну цінність, оскільки дозволяє значно спростити процес налаштування мережевого обладнання MikroTik, зменшуючи час та ймовірність помилок. Інструмент забезпечує крос-платформну сумісність, що робить його зручним для використання в різних операційних системах та мережевих середовищах.

Таким чином, виконана робота не лише демонструє можливості автоматизації процесу налаштування мережевого обладнання, але й вносить вагомий вклад у розвиток інструментів для ефективного управління мережевими конфігураціями, що є актуальним для сучасних інформаційних технологій.

ПЕРЕЛІК ПОСИЛАНЬ

1. MikroTik. [Електронний ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/MikroTik
2. SMEs Definition. [Електронний ресурс] - Режим доступу: https://ec.europa.eu/growth/smes/sme-definition_en
3. WinBox Documentation. [Електронний ресурс] - Режим доступу: https://wiki.mikrotik.com/wiki/Manual:Winbox
4. WebFig Documentation. [Електронний ресурс] - Режим доступу: https://wiki.mikrotik.com/wiki/Manual:Webfig
5. Ansible Documentation. [Електронний ресурс] - Режим доступу: https://docs.ansible.com/
6. Puppet Documentation. [Електронний ресурс] - Режим доступу: https://puppet.com/docs/
7. Chef Documentation. [Електронний ресурс] - Режим доступу: https://docs.chef.io/
8. Secure Shell (SSH) Overview. [Електронний ресурс] - Режим доступу: https://www.ssh.com/ssh/
9. API Documentation. [Електронний ресурс] - Режим доступу: https://www.api-documentation.org/
10. MikroTik Cloud Hosted Router. [Електронний ресурс] - Режим доступу: https://mikrotik.com/products/cloud-hosted-router
11. Netmiko Library. Netmiko Documentation. [Електронний ресурс] - Режим доступу: https://github.com/ktbyers/netmiko
12. Paramiko Library. Paramiko Documentation. [Електронний ресурс] - Режим доступу: http://www.paramiko.org/
13. Tkinter Documentation. [Електронний ресурс] - Режим доступу: https://docs.python.org/3/library/tkinter.html
14. SQLite Documentation. [Електронний ресурс] - Режим доступу: https://www.sqlite.org/docs.html
15. Методичні рекомендації до виконання та оформлення кваліфікаційних робіт бакалаврів / Гаркуша І.М., Гнатушенко В.В., Коротенко Г.М. – Д.: НТУ «ДП», 2020. – 27 с.
16. ДСТУ 3008:2015. Державний стандарт України. Інформація та документація. Звіти в сфері науки і техніки. Структура та правила оформлювання. Видання офіційне. Держстандарт України – К.: Держстандарт, 2015. – 32 с.
17. ДСТУ 3582:2013. Інформація та документація. Бібліографічний опис. Скорочення слів і словосполучень українською мовою. Загальні вимоги та

правила (ISO 4:1984, NEQ; ISO 832:1994, NEQ) / Нац. стандарт України. – Вид. офіц. – [На заміну ДСТУ 3582–97; чинний від 2013–08–22]. – Київ: Мінекономрозвитку України, 2014. – 15 с.

18. ДСТУ 8302:2015. Державний стандарт України. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Видання офіційне. Держстандарт України – К.: Держстандарт, 2015. – 20 с.
19. Положення про систему запобігання та виявлення плагіату в Національному технічному університеті «Дніпровська політехніка», затверджене Вченою радою 13.06.2018 (протокол №8).

ДОДАТОК А

Код програмного модулу автоматизованого налаштування конфігурації мережевого обладнання MikroTik

```
import re
import os
import time
import paramiko
import ipaddress
import sqlite3
from datetime import datetime
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
from netmiko import ConnectHandler

class Worker:
    def __init__(self, net_connection, save_dir=None):
        self.net_connection = net_connection
        self.save_dir = save_dir
        self.commands = [
            'ip address export file=ip.rsc',
            'ip firewall mangle export file=mangle.rsc',
            'ip firewall nat export file=nat.rsc',
            'ip firewall filter export file=filter.rsc',
            'queue simple export file=simple.rsc',
            'ip dns export file=dns.rsc',
            'system script export file=script.rsc',
            'system scheduler export file=scheduler.rsc',
            'tool e-mail export file=email.rsc',
            'ip firewall address-list export file=address-list.rsc',
            'ip route export file=route.rsc',
            'ip dhcp-server network export file=network.rsc',
            'queue type export file=type.rsc',
            'queue tree export file=tree.rsc',
            'interface ethernet export file=ethernet.rsc',
            'ip pool export file=pool.rsc',
            'ppp profile export file=profile.rsc',
            'system logging export file=log.rsc',
            'export file=config_backup.rsc'
        ]

    def run_in(self):
        total_commands = len(self.commands)
        for idx, command in enumerate(self.commands, start=1):
            modified_command = self.modify_command_filename(command)
```

```
            self.net_connection.send_command_timing(modified_command)
            time.sleep(5)  # First delay 5 seconds

            filename = modified_command.split('=')[-1]
            self.download_file(filename)

        messagebox.showinfo("Success", "All configurations downloaded successfully.")

    def run_out(self, commands):
        total_commands = len(commands)
        for idx, command in enumerate(commands, start=1):
            self.net_connection.send_command_timing(command)
            time.sleep(2)  # Delay 2 seconds

        messagebox.showinfo("Success", "Configuration uploaded successfully.")

    def modify_command_filename(self, command):
        # Modify the command to include the current date in filename
        filename = command.split('=')[-1]
        date_string = datetime.now().strftime("%d_%m_%Y")
        modified_command = command.replace(filename, f"{filename.split('.')[0]}_{date_string}.rsc")
        return modified_command

    def download_file(self, filename):
        max_attempts = 3  # Maximum number of attempts
        found_file = False

        while not found_file and max_attempts > 0:
            try:
                # Use send_command instead of send_command_timing to better handle the response
                output = self.net_connection.send_command(f"file print where name={filename}")

                # Check if the filename is in the output
                if f'{filename}' in output and 'script' in output:
                    transport = paramiko.Transport((self.net_connection.host, self.net_connection.port))
                    transport.connect(username=self.net_connection.username, password=self.net_connection.password)
                    sftp = paramiko.SFTPClient.from_transport(transport)

                    remote_file_path = f'/{filename}'
                    date_string = datetime.now().strftime("%d.%m.%Y")
                    filename_with_date = f"{filename.split('.')[0]}_{date_string}.rsc"
                    local_file_path = os.path.join(self.save_dir, filename_with_date)

                    # Download the file
                    sftp.get(remote_file_path, local_file_path)
```

```
                print(f"File downloaded successfully: {local_file_path}")

                found_file = True  # Set flag to exit the loop

                # Close SFTP and transport
                sftp.close()
                transport.close()

            else:
                print(f"File {filename} was not created. Retrying in 30 seconds...")
                time.sleep(30)  # Additional delay in 30 seconds
                max_attempts -= 1  # Decrement attempts

        except Exception as e:
            print(f"Error downloading file: {e}")
            max_attempts -= 1  # Decrement attempts

    if not found_file:
        # If the file was not created after all attempts
        print(f"File {filename} was not created after {max_attempts} attempts.")

class MikroTikApp:
    def __init__(self, root):
        self.root = root
        self.root.title("MikroTik Configurrator")
        self.root.geometry("450x400")
        self.create_widgets()

        self.netmiko_connection = None
        self.worker = None

        # Create or connect to the SQLite database
        self.conn = sqlite3.connect('connection_data.db')
        self.cursor = self.conn.cursor()

        # Create the connections table if it doesn't exist
        self.create_connections_table()

        # Load saved connection data
        self.populate_previous_connections_menu()

    def populate_previous_connections_menu(self):
        # Retrieve saved connection data
        self.cursor.execute("SELECT * FROM connections ORDER BY id DESC")
        data = self.cursor.fetchall()
```

```
        # Populate the dropdown menu with the saved connection records
        connection_names = [f"{data[i][1]} - {data[i][2]}:{data[i][3]}" for i in range(len(data))]
        self.previous_connections["values"] = connection_names
        # Установка размера выпадающего меню для отображения всех данных
        self.previous_connections.config(height=len(connection_names))

    def create_connections_table(self):
        # Create the connections table if it doesn't exist
        self.cursor.execute('''
            CREATE TABLE IF NOT EXISTS connections (
                id INTEGER PRIMARY KEY,
                ip_address TEXT,
                mac_address TEXT,
                port INTEGER,
                username TEXT
            )
        ''')
        self.conn.commit()

    def create_widgets(self):
        main_frame = ttk.Frame(self.root)
        main_frame.grid(row=0, column=0, sticky="nsew")

        # Other widget creation code remains the same...

        self.previous_connections = ttk.Combobox(main_frame, state="readonly", width=35)
        self.previous_connections.grid(row=0, column=1, padx=10, pady=5, sticky="ew")
        self.previous_connections.bind("<<ComboboxSelected>>", self.load_selected_connection)


        ttk.Label(main_frame, text="OR Enter New Connection Details:").grid(row=1, column=0, columnspan=3, padx=10,
pady=5, sticky="w")

        ttk.Label(main_frame, text="IP Address:").grid(row=2, column=0, padx=10, pady=5, sticky="w")
        self.ip_input = ttk.Entry(main_frame)
        self.ip_input.grid(row=2, column=1, padx=10, pady=5, sticky="ew")
        self.ip_input.insert(0, "Enter IP Address")

        ttk.Label(main_frame, text="MAC Address:").grid(row=3, column=0, padx=10, pady=5, sticky="w")
        self.mac_input = ttk.Entry(main_frame)
        self.mac_input.grid(row=3, column=1, padx=10, pady=5, sticky="ew")
        self.mac_input.insert(0, "MAC Address")

        ttk.Label(main_frame, text="Port (default: 22):").grid(row=4, column=0, padx=10, pady=5, sticky="w")
```

```
        self.port_input = ttk.Entry(main_frame)
        self.port_input.grid(row=4, column=1, padx=10, pady=5, sticky="ew")
        self.port_input.insert(0, "22")

        ttk.Label(main_frame, text="Username:").grid(row=5, column=0, padx=10, pady=5, sticky="w")
        self.username_input = ttk.Entry(main_frame)
        self.username_input.grid(row=5, column=1, padx=10, pady=5, sticky="ew")
        self.username_input.insert(0, "Username")

        ttk.Label(main_frame, text="Password:").grid(row=6, column=0, padx=10, pady=5, sticky="w")
        self.password_input = ttk.Entry(main_frame, show="*")
        self.password_input.grid(row=6, column=1, padx=10, pady=5, sticky="ew")
        self.password_input.insert(0, "Password")

        connect_button = ttk.Button(main_frame, text="Connect to MikroTik", command=self.connect_to_mikrotik)
        connect_button.grid(row=7, column=0, columnspan=3, padx=10, pady=10, sticky="ew")

        self.download_button = ttk.Button(main_frame, text="Download All Configs", command=self.download_all_configs,
state=tk.DISABLED)
        self.download_button.grid(row=8, column=0, columnspan=3, padx=10, pady=5, sticky="ew")

        self.upload_button = ttk.Button(main_frame, text="Upload Configuration File", command=self.upload_configuration,
state=tk.DISABLED)
        self.upload_button.grid(row=9, column=0, columnspan=3, padx=10, pady=5, sticky="ew")

        edit_button = ttk.Button(main_frame, text="Edit Configuration File", command=self.open_additional_window)
        edit_button.grid(row=10, column=0, columnspan=3, padx=10, pady=5, sticky="ew")

    def load_saved_connection_data(self):
        # Retrieve saved connection data
        self.cursor.execute("SELECT * FROM connections ORDER BY id DESC")
        data = self.cursor.fetchall()

        # Populate the dropdown menu with the saved connection records
        connection_names = [f"{data[i][1]} - {data[i][2]}:{data[i][3]}" for i in range(len(data))]
        self.previous_connections["values"] = connection_names

    def load_selected_connection(self, event=None):
        # Get the selected connection index
        selected_index = self.previous_connections.current()
        if selected_index >= 0:
            # Retrieve the selected connection record from the database
            self.cursor.execute("SELECT * FROM connections ORDER BY id DESC")
            data = self.cursor.fetchall()
            selected_connection = data[selected_index]
```

```
        # Populate the input fields with the selected connection data
        self.ip_input.delete(0, tk.END)
        self.ip_input.insert(0, selected_connection[1])
        self.mac_input.delete(0, tk.END)
        self.mac_input.insert(0, selected_connection[2])
        self.port_input.delete(0, tk.END)
        self.port_input.insert(0, str(selected_connection[3]))
        self.username_input.delete(0, tk.END)
        self.username_input.insert(0, selected_connection[4])


def connect_to_mikrotik(self):
    ip_address = self.ip_input.get().strip()
    mac_address = self.mac_input.get().strip()
    port = self.port_input.get().strip()
    username = self.username_input.get().strip()
    password = self.password_input.get().strip()

    # Check for empty fields
    if not ip_address or not mac_address or not port or not username:
        messagebox.showwarning("Warning", "Please fill in all fields except Password.")
        return

    # Connect to MikroTik using netmiko
    try:
        device = {
            "device_type": "mikrotik_routeros",
            "host": ip_address,
            "username": username,
            "password": password,  # Can be empty
            "port": int(port)
        }
        self.netmiko_connection = ConnectHandler(**device)

        # Save connection data to the database
        self.cursor.execute("INSERT INTO connections (ip_address, mac_address, port, username) VALUES (?, ?, ?, ?)",
                            (ip_address, mac_address, int(port), username))
        self.conn.commit()

        # If connection successful, enable buttons
        self.download_button.config(state=tk.NORMAL)
        self.upload_button.config(state=tk.NORMAL)

        messagebox.showinfo("Success", "Connected to MikroTik.")
```

```
    except Exception as e:
        messagebox.showerror("Error", f"Failed to connect to MikroTik: {str(e)}")

def open_additional_window(self):
    additional_window = tk.Toplevel(self.root)
    additional_window.title("Edit Configuration File")
    additional_window.geometry("600x200")

    ttk.Label(additional_window, text="Edit Configuration File").grid(row=0, column=0, pady=10)

    self.load_button = tk.Button(additional_window, text="Load Configuration", command=self.load_config)
    self.load_button.grid(row=0, column=1)

    self.pool_range_label = tk.Label(additional_window, text="/ip pool:")
    self.pool_range_label.grid(row=1, column=0)
    self.new_pool_entry = tk.Entry(additional_window, width=70)
    self.new_pool_entry.grid(row=1, column=1)

    self.address_range_label = tk.Label(additional_window, text="/ip address:")
    self.address_range_label.grid(row=2, column=0)
    self.new_address_entry = tk.Entry(additional_window, width=70)
    self.new_address_entry.grid(row=2, column=1)

    self.dhcp_network_label = tk.Label(additional_window, text="/ip dhcp-server network:")
    self.dhcp_network_label.grid(row=3, column=0)
    self.new_dhcp_network_entry = tk.Entry(additional_window, width=70)
    self.new_dhcp_network_entry.grid(row=3, column=1)

    self.dns_static_label = tk.Label(additional_window, text="/ip dns static:")
    self.dns_static_label.grid(row=4, column=0)
    self.new_dns_static_entry = tk.Entry(additional_window, width=70)
    self.new_dns_static_entry.grid(row=4, column=1)

    self.save_button = tk.Button(additional_window, text="Save Configuration", command=self.save_config, state="disabled")
    self.save_button.grid(row=5, column=1)

    self.config_lines = []
    self.loaded_file_path = ""


def load_config(self):
    file_path = filedialog.askopenfilename(title="Select Configuration File")
    if file_path:
        with open(file_path, 'r') as file:
```

```
            self.config_lines = file.readlines()
            self.loaded_file_path = file_path
            self.save_button.config(state="normal")

    # Search for "/ip pool" and "/ip address" in the loaded configuration
    pool_index = self.find_index_of_line_containing("/ip pool")
    address_index = self.find_index_of_line_containing("/ip address")
    dhcp_network_index = self.find_index_of_line_containing("/ip dhcp-server network")
    dns_static_index = self.find_index_of_line_containing("/ip dns static")

    # Display the next lines in the entry fields if found
    if pool_index != -1:
        self.display_next_line(pool_index, self.new_pool_entry)
    if address_index != -1:
        self.display_next_line(address_index, self.new_address_entry)
    if dhcp_network_index != -1:
        self.display_next_line(dhcp_network_index, self.new_dhcp_network_entry)
    if dns_static_index != -1:
        self.display_next_line(dns_static_index, self.new_dns_static_entry)

def find_index_of_line_containing(self, keyword):
    index = -1
    for i, line in enumerate(self.config_lines):
        if keyword in line:
            index = i
            break
    return index

def display_next_line(self, index, entry_widget):
    next_line_index = index + 1
    if next_line_index < len(self.config_lines):
        next_line = self.config_lines[next_line_index].strip()
        entry_widget.delete(0, tk.END)
        entry_widget.insert(0, next_line)

def save_config(self):
    new_pool_value = self.new_pool_entry.get()
    new_address_value = self.new_address_entry.get()
    new_dhcp_network_value = self.new_dhcp_network_entry.get()
    new_dns_static_index_value = self.new_dns_static_entry.get()

    if not self.loaded_file_path:
        messagebox.showerror("Error", "No configuration loaded.")
        return
```

```
    with open(self.loaded_file_path, 'r') as file:
        lines = file.readlines()

    # Find the index of the line containing "/ip pool" and "/ip address"
    pool_index = self.find_index_of_line_containing("/ip pool")
    address_index = self.find_index_of_line_containing("/ip address")
    dhcp_network_index = self.find_index_of_line_containing("/ip dhcp-server network")
    dns_static_index = self.find_index_of_line_containing("/ip dns static")


    # Update the next lines after "/ip pool" and "/ip address" with the new values
    if pool_index + 1 < len(lines):
        lines[pool_index + 1] = new_pool_value + '\n'
    if address_index + 1 < len(lines):
        lines[address_index + 1] = new_address_value + '\n'
    if dhcp_network_index + 1 < len(lines):
        lines[dhcp_network_index + 1] = new_dhcp_network_value + '\n'
    if dns_static_index + 1 < len(lines):
        lines[dns_static_index + 1] = new_dns_static_index_value + '\n'

    file_path = filedialog.asksaveasfilename(title="Save Configuration As", defaultextension=".txt")
    if file_path:
        with open(file_path, 'w') as file:
            file.writelines(lines)
        messagebox.showinfo("Success", "Configuration saved successfully!")



def download_all_configs(self):
    save_dir = filedialog.askdirectory()

    if not save_dir:
        return

    self.worker = Worker(self.netmiko_connection, save_dir)
    self.worker.run_in()

def upload_configuration(self):
    file_path = filedialog.askopenfilename(filetypes=[("MikroTik Config files", "*.rsc")])

    if not file_path:
        return

    try:
        with open(file_path, 'r') as file:
```

```
                commands = [line.strip() for line in file if not line.startswith('#')]

            self.worker = Worker(self.netmiko_connection)
            self.worker.run_out(commands)

        except Exception as e:
            messagebox.showerror("Error", f"Failed to read configuration file: {str(e)}")

    def load_saved_connection_data(self):
        # Retrieve saved connection data
        self.cursor.execute("SELECT * FROM connections ORDER BY id DESC LIMIT 1")
        data = self.cursor.fetchone()

        # Display the latest saved connection data in the input fields
        if data:
            self.ip_input.delete(0, tk.END)
            self.ip_input.insert(0, data[1])
            self.mac_input.delete(0, tk.END)
            self.mac_input.insert(0, data[2])
            self.port_input.delete(0, tk.END)
            self.port_input.insert(0, str(data[3]))
            self.username_input.delete(0, tk.END)
            self.username_input.insert(0, data[4])

    def __del__(self):
        # Close the database connection when the object is destroyed
        self.conn.close()

if __name__ == "__main__":
    root = tk.Tk()
    app = MikroTikApp(root)
    root.mainloop()
```

ДОДАТОК Б

Код тестування основних функцій модуля

Тестування підключення до пристрою MikroTik:

```
def test_connect_to_mikrotik():
    app.ip_input.insert(0, "192.168.88.1")
    app.mac_input.insert(0, "0d:1a:fe:3c:17:b2")
    app.port_input.insert(0, "22")
    app.username_input.insert(0, "admin")
    app.password_input.insert(0, "password")

    app.connect_to_mikrotik()

    assert app.netmiko_connection is not None
    assert app.download_button.cget("state") == "normal"
    assert app.upload_button.cget("state") == "normal"
    print("Test connect_to_mikrotik passed")

test_connect_to_mikrotik()
```

Тестування експорту конфігурацій:

```
def test_export_configurations():
    save_dir = "/path/to/save/directory"
    worker = Worker(app.netmiko_connection, save_dir)

    worker.run_in()

    for command in worker.commands:
        filename = worker.modify_command_filename(command).split('=')[-1]
        local_file_path = os.path.join(save_dir, filename)
        assert os.path.exists(local_file_path)
    print("Test export_configurations passed")

test_export_configurations()
```

Тестування імпорту конфігурацій:

```
def test_import_configurations():
    file_path = "/path/to/config/file.rsc"
    commands = []

    with open(file_path, 'r') as file:
        commands = [line.strip() for line in file if not line.startswith('#')]

    worker = Worker(app.netmiko_connection)
```

```
    worker.run_out(commands)

    # This is a simplified check. In a real scenario, more validation may be required.
    print("Test import_configurations passed")

test_import_configurations()
```

Тестування редагування конфігурацій:

```
def test_edit_configurations():
    file_path = "/path/to/config/file.rsc"
    app.open_additional_window()
    app.load_config()

    app.new_pool_entry.insert(0, "new pool value")
    app.new_address_entry.insert(0, "new address value")
    app.new_dhcp_network_entry.insert(0, "new dhcp network value")
    app.new_dns_static_entry.insert(0, "new dns static value")

    app.save_config()

    with open(file_path, 'r') as file:
        lines = file.readlines()
        assert "new pool value\n" in lines
        assert "new address value\n" in lines
        assert "new dhcp network value\n" in lines
        assert "new dns static value\n" in lines
    print("Test edit_configurations passed")

test_edit_configurations()
```