Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“___” _____ 20__ р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
«Програмне забезпечення системи проектування та побудови топології мереж у центрах обробки даних»

Виконав здобувач вищої освіти
IV курсу, групи КМ-20
ОПП «Кібербезпека»
спеціальності 123 «Комп’ютерна інженерія»
_____ Поченікін В.О.
«____» _____ 20 р.

Керівник проекту
доктор технічних наук, професор
_____ Коваленко А. С.
«____» _____ 20 р.
Рецензент _____
_____

м. Кропивницький

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *123 Комп'ютерна інженерія*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« __ » _____ 20__ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Поченікін Владислав Олександрович
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи проектування та побудови топології мереж у центрах обробки даних*

керівник роботи *Коваленко Анна Степанівна*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу *№ 133-02 від 01 . 04 .2024 року*

2. Строк подання студентом роботи до захисту *07 . 06 .2024 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи проектування та побудови топології мереж у центрах обробки даних*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

6. Дата видачі завдання «17» січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	19.05.2024 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Поченікін В.О. Програмне забезпечення системи проектування та побудови топології мереж у центрах обробки даних. 123 Комп'ютерні мережі. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

У даній кваліфікаційній бакалаврській роботі розроблено програмне забезпечення, спрямоване на проектування та побудову топології мереж у центрах обробки даних.

Метою роботи є створення програмного інструменту для оптимізації процесу планування та структуризації мережевих зав'язків в центрах обробки даних.

Результатом роботи є програмна реалізація засобу, що дозволяє ефективно вирішувати завдання проектування мережі в таких середовищах.

У процесі розробки програми проведено аналіз існуючих методів та алгоритмів, що використовуються при проектуванні мереж, а також досліджено різноманітні програмні засоби, придатні для цієї мети. На основі цього аналізу було розроблено власне програмне забезпечення, що включає в себе усі необхідні компоненти для планування та побудови мережевих топологій.

Результатом роботи є інтуїтивно зрозумілий інтерфейс користувача, що спрощує процес проектування мережі, а також надані інструкції з використання програмного забезпечення. Програма придатна для використання на персональних комп'ютерах з архітектурою IBM PC та операційною системою Windows 10/11.

Програму розроблено на мові програмування Java.

Ключові слова: проектування мереж, топологія мереж, центри обробки даних, програмне забезпечення.

ABSTRACT

Pochenikin V.O. Software for designing and building network topology in data centers. Central Ukrainian national technical university. Kropyvnytskyi. 2024.

This bachelor's qualification work presents software aimed at designing and constructing network topologies in data processing centers.

The goal of the work is to create a software tool to optimize the process of planning and structuring network connections in data processing centers.

The outcome of the work is the software implementation of a tool that effectively addresses the task of network design in such environments.

During the program's development, an analysis of existing methods and algorithms used in network design was conducted, along with an exploration of various software tools suitable for this purpose. Based on this analysis, proprietary software was developed, encompassing all necessary components for planning and constructing network topologies.

The result is an intuitively understandable user interface that simplifies the network design process, along with provided instructions for software utilization.

The program is suitable for use on personal computers with IBM PC architecture and Windows 10/11 operating systems.

The software is developed using the Java programming language.

Keywords: network design, network topology, data processing centers, software.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

LAN - Локальна мережа (Local Area Network).

WAN - Широкомасштабна мережа (Wide Area Network).

VPN - Віртуальна приватна мережа (Virtual Private Network).

IP - Інтернет-протокол (Internet Protocol).

MAC-адреса - Адреса керуючого доступу до носія (Media Access Control Address).

SSID - Ідентифікатор служби наближеності (Service Set Identifier).

AES - Розширений стандарт шифрування (Advanced Encryption Standard).

TLS - Протокол захищеної передачі даних (Transport Layer Security).

Firewall - Брандмауер, мережевий елемент для контролю та фільтрації трафіку.

DNS - Система доменних імен (Domain Name System).

DHCP - Протокол динамічної конфігурації хоста (Dynamic Host Configuration Protocol).

Router - Маршрутизатор, пристрій для маршрутизації мережевого трафіку.

Switch - Комутатор, пристрій для з'єднання різних сегментів мережі.

Gateway - Шлюз, точка входу або вихід з мережі.

Packet - Пакет даних, основна одиниця передачі інформації в мережі.

MAC-фільтр - Фільтрація доступу на основі MAC-адрес.

ВСТУП

Актуальність теми. У сучасному ІТ-світі, де обсяги даних зростають в геометричній прогресії, проектування та побудова мережевих топологій в центрах обробки даних є ключовим фактором забезпечення ефективності та надійності інформаційних систем. Розвиток цифрових технологій не тільки збільшує обсяги даних, які необхідно обробляти, але й висуває складні вимоги до мережевої архітектури та програмного забезпечення, яке керує цими мережами.

Однією з ключових вимог до програмного забезпечення, яке проектує і будує мережеві топології центрів обробки даних, є здатність швидко реагувати і адаптуватися до мінливих мережевих вимог. Це включає в себе здатність автоматично розподіляти мережеві ресурси відповідно до навантаження, виконувати резервне копіювання та відновлення даних, а також контролювати та аналізувати пропускну здатність мережі.

Другою ключовою вимогою є безпека даних і мережі. Враховуючи зростання кіберзагроз та зловмисних атак, програмне забезпечення повинно забезпечувати надійний захист від несанкціонованого доступу, шифрування даних та моніторинг мережевої активності для своєчасного виявлення потенційних загроз.

Крім того, програмне забезпечення, яке проектує та будує топологію мережі дата-центру, повинно мати зручний інтерфейс, що дозволяє мережевим інженерам ефективно керувати та налаштовувати мережеві ресурси, не вимагаючи при цьому глибоких технічних знань.

Тому метою даного дослідження є розробка програмного забезпечення, що дозволяє ефективно управляти та створювати топології мережі ЦОД з урахуванням сучасних вимог до швидкості, надійності та безпеки.

Мета й завдання дослідження. Метою даного дослідження є розробка програмного забезпечення для ефективного проектування та побудови мережевих топологій у центрах обробки даних з максимальною швидкістю, надійністю та безпекою.

Для досягнення мети дослідження були визначені наступні завдання.

− Аналіз існуючих технологій та методології проектування мережевих топологій: огляд існуючих стандартів, протоколів та технологій проектування мереж в центрах обробки даних.

− Розробка алгоритмів автоматизованого проектування мережевих топологій: розробка програмних алгоритмів для автоматизації процесу проектування мережевих топологій, що відповідають вимогам швидкості, надійності та безпеки.

− Тестування та валідація розроблених алгоритмів: проведення різноманітних тестів для перевірки ефективності та коректності роботи розроблених алгоритмів при різних вхідних даних та умовах.

− Програмна реалізація: розробка програмного забезпечення для створення мережевих топологій дата-центрів з використанням розроблених алгоритмів.

Практична цінність отриманих результатів. Розроблені алгоритми успішно вирішують завдання створення та проектування топології мережі. Автоматизація процесу проектування економить час і ресурси, забезпечує максимальну швидкість передачі даних і надійність з'єднання, а також підвищує безпеку топології мережі. Програмне забезпечення допомагає компаніям оптимізувати витрати, забезпечуючи ефективну експлуатацію та обслуговування інфраструктури дата-центру.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Призначенням програмного забезпечення для проектування та побудови мережевих топологій в центрах обробки даних є забезпечення комплексного підходу до управління та оптимізації інфраструктури передачі даних. Система спрямована на ефективне розгортання, моніторинг та підтримку мережевих рішень в центрах обробки даних. Ключові можливості системи

– Проектування топології мережі: Інженери можуть створити оптимальну топологію мережі для задоволення конкретних потреб центру обробки даних.

– Візуалізація та аналіз: Надає прості у використанні інструменти для візуалізації мережевої інфраструктури та аналізу її продуктивності, що дозволяє швидко виявляти та вирішувати проблеми.

– Автоматична конфігурація: Автоматично налаштовує та керує мережевим обладнанням для забезпечення швидкості, безпеки та стабільності мережі.

– Моніторинг та оповіщення: Безперервний моніторинг стану мережі та оповіщення про потенційні проблеми та перебої в роботі.

– Інтеграція з іншими системами: Можливість інтеграції з іншими системами управління даними для створення єдиного середовища центру обробки даних.

Крім того, ключовим завданням при проектуванні мережі в центрі обробки даних є дотримання ідеалів найкоротшого шляху і максимальної пропускної здатності. Для досягнення цих параметрів програмний пакет надає можливість аналізувати різні варіанти архітектурної топології та їх вплив на показники зв'язку і продуктивності. Програмне забезпечення надає інструменти для легкого перегляду та визначення фізичної пропускної здатності різних точок мережевої інфраструктури для забезпечення оптимального балансування

навантаження.

Програмний пакет надає інструменти для виявлення та графічного відображення проблемних місць у мережі, а також пропонує рішення. Це скорочує час і витрати на обслуговування мережі.

Крім того, програмні комплекси забезпечують високий ступінь автоматизації та стандартизації процесу проектування, значно зменшуючи ймовірність помилок і забезпечуючи одноманітність конфігурації мережі. Використання таких систем сприяє підвищенню якості проекту та скороченню часу розробки.

Програмне забезпечення для проектування та побудови топологій мереж відіграють важливу роль у підвищенні ефективності та якості мережевих архітектур. Надаючи інструменти для аналізу, оптимізації та інсайту, вони забезпечують оптимальне балансування навантаження та ефективність мережі в центрах обробки даних.

1.2 Область застосування

Однією з основних функцій програмного забезпечення для побудови топології мережі є надання інженерам і адміністраторам інструментів, необхідних для створення оптимізованої топології мережі. Від великих центрів обробки даних до невеликих офісних мереж, правильне проектування топології мережі може значно підвищити продуктивність і ефективність інфраструктури передачі даних.

У світі віртуалізації та хмарних обчислень, де ресурси можуть бути розподілені між декількома серверами та центрами обробки даних, програмне забезпечення для топології мережі відіграє важливу роль у створенні та управлінні віртуальними мережами. Воно допомагає забезпечити безпеку і надійність мережі та ефективний обмін даними і ресурсами між різними обчислювальними вузлами.

В Інтернеті речей, де мільйони підключених пристроїв обмінюються даними, програмне забезпечення мережевої топології використовується для створення надійних і масштабованих мереж зв'язку між цими пристроями. Це дозволяє ефективно управляти великими обсягами даних, що генеруються пристроями Інтернету речей, а також швидко і надійно передавати їх в центри обробки даних.

У сфері зв'язку та управління мережею програмне забезпечення мережевої топології використовується для конфігурації та управління мережевими пристроями, такими як комутатори та маршрутизатори. Воно допомагає адміністраторам контролювати мережеві вузли, відстежувати мережевий трафік і виявляти потенційні проблеми в мережі.

Все більше підприємств та організацій усвідомлюють важливість правильної побудови та управління мережевою інфраструктурою. Програмне забезпечення для топології мережі є важливим інструментом у цьому процесі, що дозволяє їм будувати та оптимізувати складні мережеві інфраструктури, забезпечуючи при цьому безпеку, надійність та ефективність мережі.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Топологія мережі описує фізичну або логічну структуру зв'язку між комп'ютерами та іншими пристроями в мережі. Вона визначає, як вузли та пристрої взаємодіють один з одним і як вони обмінюються даними.

Правильна топологія має вирішальне значення для ефективного функціонування мережі та надійного обміну інформацією між пристроями. Основні причини використання мережевої топології наступні.

– Забезпечення належного підключення: Топологія визначає, як пристрої підключаються до мережі. Це гарантує, що пристрої підключаються правильно і ефективно, і що дані можуть швидко обмінюватися.

– Масштабованість: Для масштабованості: Налаштувавши топологію мережі, мережу можна легко розширювати в міру зростання ваших потреб. Наприклад, ви можете збільшити розмір вашої мережі, підключивши додаткові пристрої до існуючої мережі.

– Ізолювати проблеми: У деяких топологіях такі проблеми, як апаратні збої або пошкодження кабелю, можна локально ізолювати, щоб решта мережі продовжувала працювати.

– Управління трафіком: Управління трафіком: Використання правильної топології може забезпечити ефективну передачу даних через мережу, регулюючи потік трафіку і мінімізуючи перевантаження.

Загалом, топологія мережі визначає, як пристрої підключаються один до одного і як відбувається обмін даними. Вибір правильної топології може підвищити ефективність і надійність мережі і є ключовим елементом проектування

та управління мережею. Існує низка програм, які допомагають аналізувати та візуалізувати топологію мережі.

Cisco Packet Tracer - це інтерактивне програмне забезпечення, розроблене компанією Cisco Systems, яке призначене для моделювання, налагодження та аналізу мережевих систем. Воно широко використовується для навчання студентів та спеціалістів у сфері мережевих технологій та для розробки різноманітних мережевих конфігурацій. Основні характеристики Cisco Packet Tracer включають:

– Симуляція мережевих пристроїв: Packet Tracer дозволяє створювати віртуальні пристрої Cisco, такі як маршрутизатори, комутатори, ПК, сервери тощо, та налаштовувати їх для моделювання різних мережевих сценаріїв.

– Графічне інтерфейсне середовище: Програма має зручний графічний інтерфейс, що спрощує процес налаштування та взаємодії з мережевими пристроями.

– Підтримка різних мережевих протоколів: Packet Tracer підтримує широкий спектр мережевих протоколів, включаючи Ethernet, TCP/IP, OSPF, EIGRP, VLAN, IPv6 та багато інших.

– Навчальні можливості: Використання Packet Tracer дозволяє студентам вивчати різні аспекти мережевих технологій, проводити експерименти та розв'язувати завдання у віртуальному середовищі.

– Модульність та розширюваність: Програма має можливість розширення за допомогою додаткових модулів, що дозволяє розширити її функціональні можливості.

Переваги:

– Навчальні можливості: Packet Tracer розроблений спеціально для навчання студентів та працівників IT у сфері мережевих технологій. Він дозволяє створювати віртуальні мережі, налаштовувати пристрої та виконувати різноманітні мережеві сценарії без реального обладнання.

– Легкий доступ: Packet Tracer доступний для завантаження на офіційному веб-сайті Cisco та може бути використаний безкоштовно або за невелику плату.

– Широкий функціонал: Програма підтримує багато різних мережевих пристроїв Cisco, таких як маршрутизатори, комутатори, маршрутизатори серії ASA та інші. Вона також включає в себе різноманітні мережеві протоколи та сервіси.

– Симуляція реального середовища: Packet Tracer дозволяє студентам емулювати реальне мережеве середовище, що допомагає їм краще зрозуміти та набути практичних навичок.

Недоліки:

– Обмежений функціонал: В порівнянні з реальним мережевим обладнанням, Packet Tracer має обмежений функціонал та не підтримує всі аспекти реальних мережевих конфігурацій.

– Не всі пристрої підтримуються: Packet Tracer має обмежений список підтримуваних мережевих пристроїв, і деякі нові або складні моделі можуть бути відсутніми.

– Відсутність реального обладнання: Хоча Packet Tracer допомагає студентам набути практичних навичок, він не замінює реального досвіду роботи з реальним мережевим обладнанням.

– Ліцензійні обмеження: Деякі функції Packet Tracer можуть бути недоступними без оплаченої ліцензії, що може бути обмеженням для студентів та навчальних закладів з обмеженим бюджетом.

SolarWinds Network Topology Mapper є інструментом, розробленим компанією SolarWinds, який призначений для автоматизованого створення карт топології мережі. Це програмне забезпечення дозволяє автоматично відстежувати та візуалізувати всі з'єднання та пристрої у мережі, що допомагає адміністраторам легко зрозуміти структуру мережі та виявити можливі проблеми. Основні функції SolarWinds Network Topology Mapper включають:

– Автоматичне відстеження мережевих пристроїв: Програмне забезпечення автоматично сканує мережу для виявлення всіх підключених пристроїв, включаючи комутатори, маршрутизатори, сервери, принтери тощо.

– Створення діаграм топології мережі: SolarWinds Network Topology Mapper автоматично створює діаграму топології мережі на основі зібраної інформації про підключення та конфігурації пристроїв.

– Візуалізація зв'язків між пристроями: Програма відображає всі з'єднання між мережевими пристроями на діаграмі, що дозволяє адміністраторам швидко зрозуміти структуру мережі та виявити можливі проблеми з підключенням.

– Збереження та експорт діаграм: SolarWinds Network Topology Mapper дозволяє зберігати та експортувати діаграми топології мережі у різних форматах, таких як PNG, SVG, PDF, що полегшує їх подальше використання та обмін з іншими користувачами.

– Моніторинг змін у мережі: Програмне забезпечення може автоматично оновлювати діаграми топології мережі при виявленні змін у мережі, таких як додавання нових пристроїв або зміни конфігурації існуючих.

Переваги:

– Автоматизована візуалізація: SolarWinds Network Topology Mapper автоматично сканує вашу мережу та створює діаграму топології, що дозволяє швидко отримати зображення структури мережі.

– Широкі можливості інтеграції: Програмне забезпечення може інтегруватися з іншими продуктами SolarWinds та зовнішніми системами моніторингу мережі, що полегшує управління мережею та аналіз проблем.

– Спрощена адміністрація мережі: Завдяки візуалізації топології мережі адміністратори можуть легше виявляти та вирішувати проблеми в мережевій інфраструктурі.

– Автоматизація оновлень: SolarWinds Network Topology Mapper може автоматично оновлювати топологію мережі при зміні конфігурації або додаванні нових пристроїв.

– Зручний інтерфейс користувача: Програмне забезпечення має інтуїтивно зрозумілий інтерфейс, що полегшує роботу з ним навіть для новачків.

Недоліки:

– Вартість: SolarWinds Network Topology Mapper є комерційним програмним забезпеченням, що може бути дорогим в порівнянні з іншими альтернативами, особливо для малих компаній або приватних користувачів.

– Обмежені можливості для великих мереж: Деякі користувачі відзначають, що програмне забезпечення може бути менш ефективним для великих або складних мереж, порівняно з іншими інструментами.

– Залежність від точності сканування мережі: Якщо сканування мережі неправильно налаштовано або відбувається з помилками, це може призвести до неточностей у візуалізації топології.

– Потреба у високій відмовостійкості: При використанні програмного забезпечення для моніторингу мережі важливо, щоб воно було надійним та стійким до відмов. Якщо програма працює нестабільно або часто виходить з ладу, це може ускладнити адміністрування мережі.

Lucidchart - це веб-сервіс для створення професійних діаграм та схем, включаючи топології мережі. Він надає користувачам зручний інтерфейс та багатий набір інструментів для візуалізації структури мережі та її компонентів. Основні функції Lucidchart для створення топологій мережі включають:

– Бібліотека мережевих форм: Lucidchart має велику бібліотеку мережевих форм, таких як роутери, комутатори, сервери, пристрої зв'язку, що дозволяє користувачам швидко створювати складні мережеві схеми.

– Зручний графічний інтерфейс: Програма має інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам швидко та ефективно створювати, редагувати та маніпулювати елементами діаграм.

– Спільна робота та співпраця: Lucidchart підтримує спільну роботу, що дозволяє кільком користувачам одночасно працювати над однією діаграмою, а також обмінюватися коментарями та змінами.

– Експорт та інтеграція: Програмне забезпечення дозволяє експортувати діаграми у різні формати, такі як PDF, PNG, SVG, що полегшує подальше використання та обмін. Крім того, Lucidchart інтегрується з іншими сервісами,

такими як Google Drive, Microsoft Office, що дозволяє зручно працювати з документами.

– Шаблони та зразки: Сервіс надає широкий вибір шаблонів та зразків, які можна використовувати для створення різних видів мережевих топологій.

Переваги:

– Веб-заснована платформа: Lucidchart доступний через веб-браузер, що дозволяє користувачам працювати з діаграмами з будь-якого пристрою та місця з Інтернетом.

– Простота використання: Інтерфейс користувача Lucidchart інтуїтивно зрозумілий, що дозволяє навіть новачкам швидко створювати складні діаграми.

– Широкі можливості візуалізації: Платформа має великий набір форм та символів для створення мережевих топологій, що дозволяє користувачам точно відображати всі аспекти мережі.

– Спільна робота та співпраця: Lucidchart підтримує спільну роботу над діаграмами, що дозволяє кільком користувачам одночасно працювати над проектом та обмінюватися коментарями та змінами.

– Інтеграція з іншими сервісами: Платформа інтегрується з іншими популярними інструментами та сервісами, такими як Google Drive, Microsoft Office, Slack та інші.

Недоліки:

– Вартість: Lucidchart пропонує різні платні плани, що може бути дорогим для деяких користувачів, особливо для індивідуальних користувачів або невеликих компаній.

– Обмежена можливість безпеки даних: Зважаючи на те, що Lucidchart є хмарним сервісом, деякі користувачі можуть мати сумніви щодо безпеки їх даних та конфіденційності.

– Залежність від Інтернету: Lucidchart потребує постійного підключення до Інтернету для роботи, що може бути не зручним для користувачів, які працюють у місцях з обмеженим або нестабільним з'єднанням.

Microsoft Visio - це програмне забезпечення для створення професійних діаграм та схем, включаючи топології мережі. Воно є одним з найпопулярніших інструментів у своєму класі та використовується в різних галузях, включаючи IT, інженерію, бізнес та інші. Основні функції Microsoft Visio для створення топологій мережі включають:

– Широкий вибір форм та шаблонів: Visio має велику бібліотеку форм та шаблонів, спеціально призначених для створення мережевих топологій. Це включає в себе різні типи пристроїв, з'єднання та інші елементи мережі.

– Простий та потужний інтерфейс: Програма має інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам швидко та ефективно створювати, редагувати та маніпулювати елементами діаграм.

– Спільна робота та співпраця: Visio підтримує спільну роботу, що дозволяє кільком користувачам одночасно працювати над однією діаграмою, а також обмінюватися коментарями та змінами.

– Експорт та інтеграція: Програмне забезпечення дозволяє експортувати діаграми у різні формати, такі як PDF, PNG, SVG, що полегшує подальше використання та обмін. Крім того, Visio інтегрується з іншими сервісами, такими як Microsoft Office, що дозволяє зручно працювати з документами.

– Масштабованість: Visio може використовуватися для створення як простих, так і дуже складних мережевих топологій, що робить його популярним інструментом для професійних інженерів мереж.

Переваги:

– Розширені можливості: Microsoft Visio має широкий набір функцій та інструментів для створення складних мережевих топологій з різноманітними елементами.

– Інтеграція з іншими продуктами Microsoft: Візіо інтегрується з іншими продуктами Microsoft Office, що полегшує імпорт та експорт діаграм у форматах Word, Excel, PowerPoint тощо.

– Широкий вибір шаблонів та форм: Програмне забезпечення має велику бібліотеку шаблонів та форм, що дозволяє швидко створювати різноманітні типи мережевих топологій.

– Простота використання: Візіо має інтуїтивно зрозумілий інтерфейс користувача, що дозволяє навіть початківцям легко працювати з програмою.

– Спільна робота: Visio підтримує спільну роботу над проектами, що дозволяє кільком користувачам одночасно працювати над діаграмою та спів модифікувати її.

Недоліки:

– Вартість ліцензії: Microsoft Visio є комерційним програмним забезпеченням, і вартість ліцензії може бути значною для індивідуальних користувачів або невеликих компаній.

– Обмеження в масштабуванні: Деякі користувачі відзначають, що Visio може бути обмеженим у масштабуванні для дуже великих або складних мережевих топологій.

– Залежність від операційної системи Windows: Visio доступний тільки для операційних систем Windows, що може бути не зручним для користувачів, які використовують інші операційні системи.

GNS3 (Graphical Network Simulator-3) - це інноваційний інструмент для моделювання мережі, який надає можливість віртуалізувати та тестувати різні мережеві конфігурації. Це відкрите програмне забезпечення, що дозволяє користувачам створювати віртуальні мережі на основі реальних мережевих пристроїв та технологій. Основні функції GNS3 для створення мережевих топологій включають:

– Віртуалізація мережевих пристроїв: GNS3 дозволяє користувачам створювати віртуальні пристрої Cisco, Juniper, HP, Huawei та інших виробників, включаючи маршрутизатори, комутатори, файрволи та інші.

– Моделювання мережевих топологій: Програма дозволяє користувачам створювати складні мережеві топології, що включають в себе різні типи пристроїв та з'єднань.

– Тестування різних сценаріїв: GNS3 дозволяє користувачам тестувати різні мережеві сценарії, включаючи маршрутизацію, комутацію, безпеку та інші аспекти мережевої інфраструктури.

– Інтеграція з реальною мережею: Програма підтримує можливість інтеграції віртуальних мереж з реальною мережею, що дозволяє користувачам тестувати конфігурації перед їх впровадженням.

– Спільна робота та співпраця: GNS3 має можливість спільної роботи та обміну проектами між користувачами, що спрощує колективну роботу над проектами мереж.

GNS3 є потужним інструментом для моделювання та тестування мережевих топологій, який дозволяє користувачам ефективно вивчати, розробляти та тестувати різноманітні аспекти мережевої інфраструктури.

Переваги:

– Реалістичне моделювання мережі: GNS3 дозволяє вам використовувати реальне мережеве обладнання, таке як маршрутизатори та комутатори, що робить моделювання мережі максимально реалістичним.

– Велика спільнота користувачів: GNS3 має активну спільноту користувачів, що дозволяє отримувати підтримку, ділитися досвідом та використовувати різноманітні ресурси для вдосконалення навичок у мережевому проектуванні та адмініструванні.

– Безкоштовність та відкритий код: GNS3 є безкоштовним та з відкритим вихідним кодом, що робить його доступним для широкого кола користувачів та сприяє розвитку програми за допомогою внесення внесків користувачами.

– Гнучкість конфігурації: Ви можете легко змінювати конфігурації мережевого обладнання, додавати або видаляти пристрої, налаштовувати

параметри мережі та експериментувати з різними мережевими протоколами та сервісами.

– Інтеграція з віртуальними машинами: GNS3 дозволяє інтегрувати віртуальні машини в ваші мережеві топології, що розширює можливості тестування та розгортання мережевих сервісів.

Недоліки:

– Вимоги до обладнання: Для ефективної роботи GNS3 потрібне достатньо потужне обладнання, особливо для великих мережевих топологій, що може бути обмеженням для деяких користувачів.

– Складність використання для початківців: GNS3 може бути складним для новачків у мережевому моделюванні, і деякі користувачі можуть потребувати додаткового часу для вивчення та засвоєння його функціоналу.

– Обмежена підтримка мережевих пристроїв: Хоча GNS3 підтримує багато мережевого обладнання, деякі моделі можуть бути непідтримуваними або не повністю функціональними.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Альтернативними мовами програмування та інструментами розробки для створення програмного забезпечення для побудови та проектування систем топології мережі настіпні.

Python: Python - популярна мова програмування з простим синтаксисом і широким спектром можливостей; існує кілька мережевих бібліотек, таких як NetworkX і Scapy, які часто використовуються для розробки програмного забезпечення для моделювання та управління мережами.

C/C++: C та C++ - це мови програмування, що використовуються для розробки ефективного та швидкого програмного забезпечення, особливо придатні для системного програмування та розробки драйверів. Програмне забезпечення

для управління, хоча воно може бути складнішим у розробці, ніж мови високого рівня, такі як Java та Python.

JavaScript / Node.js: JavaScript - це широко використовувана мова програмування для створення веб-додатків і програмування на стороні сервера за допомогою Node.js. З її допомогою можна розробляти веб-додатки для моделювання та управління мережами, а також створювати інтерактивні веб-інтерфейси для користувачів.

Go - мова програмування, розроблена компанією Google, з простим синтаксисом і багатими можливостями паралельного програмування. Її можна використовувати для розробки ефективних і швидких додатків для моделювання та управління мережами.

Вибір мови програмування Java та фреймворку JavaFX для розробки програмного забезпечення ґрунтувався на кількох важливих факторах.

Java є кросплатформенною мовою програмування і додатки, розроблені на Java, можуть працювати на операційних системах, які підтримують віртуальну машину Java (JVM). Це важливо при роботі з мережевим обладнанням, яке може використовуватися на різних операційних системах.

Крім того, широкий спектр стандартних класів Java і бібліотек сторонніх розробників дозволяє легко розробляти різні програмні функції без необхідності вносити значні зміни. Це означає, що різні функції систем проектування мереж і топології можуть бути реалізовані швидко і ефективно.

JavaFX, з іншого боку, є передовою технологією для створення кросплатформних користувацьких інтерфейсів, з вбудованими можливостями для створення графічних компонентів, анімації та взаємодії з користувачем. Гнучкість і можливість налаштування дозволяють створювати інтерактивні та візуально привабливі інтерфейси, що є критично важливим для комфортного та ефективного використання програмного забезпечення. Для розробки системи було обрано такі інструменти:

IntelliJ IDEA - інтегроване середовище розробки (IDE) для Java, яке надає

широкий спектр можливостей для створення, налагодження та тестування програмного забезпечення.

Бібліотеки для запуску мережевих пристроїв, такі як Apache MINA, Netty та Java Network Launch Protocol (JNLP), які забезпечують ефективний зв'язок з мережевими пристроями.

2.3 Розгорнута постановка завдання

Розробку програмного забезпечення для побудови топології мережі в центрі обробки даних можна розділити на кілька етапів.

Етап аналізу вимог вимагає ретельного вивчення вимог замовника і користувачів до програми. Це означає, що всі вимоги, пов'язані з додатком, повинні бути прочитані і зрозумілі. Необхідно розуміти процеси, які виконує додаток, і дії, які необхідно виконати для їх виконання.

Також на цьому етапі необхідно визначити, які дані використовує додаток і як вони повинні оброблятися. Іншими словами, необхідно визначити дані, які входять у додаток, дані, які виходять з додатку, і дані, необхідні для внутрішніх розрахунків і обробки.

Останнім кроком в аналізі вимог є визначення критеріїв, за якими буде вимірюватися успіх програми. Це означає визначення того, як буде оцінюватися додаток, наприклад, шляхом запиту конкретних критеріїв для оцінки продуктивності, точності, надійності, зручності використання та інших критеріїв, важливих для клієнтів і користувачів додатку.

Етап проектування вимагає детального аналізу архітектури системи з метою визначення топології мережі та визначення ключових напрямків розвитку. На цьому етапі команда розробників розглядає всі аспекти вимог і формулює стратегію розробки додатку.

Першим кроком є проведення детального аналізу архітектури системи. Це включає вивчення потреб користувачів, визначення функціональних вимог і

необхідності виконання конкретних процесів, які повинен виконувати додаток. На основі цього аналізу визначаються основні елементи програми, які реалізуються у вигляді класів, інтерфейсів, методів і функцій.

Потім розробляються класи і встановлюються взаємозв'язки між класами. Під час цього процесу важливо врахувати логічні зв'язки між програмними компонентами і забезпечити їхню взаємодію для ефективної роботи програми.

Процес проектування також визначає алгоритми побудови топології мережі. Ці алгоритми повинні бути добре відпрацьованими та ефективними, щоб забезпечити необхідну функціональність програми.

Для кращого розуміння архітектури програми створюються діаграми класів і зв'язків. Ці діаграми візуалізують взаємодію між класами, полегшують розуміння архітектури додатку і сприяють більш ефективній розробці.

На етапі розробки програмне забезпечення мережевої топології переходить від концептуального проектування до фактичної реалізації програмного продукту. Ця фаза включає наступні кроки:

По-перше, класи, інтерфейси та методи реалізуються на мові програмування, визначеній на етапі проектування. Це означає, що програмний код пишеться з урахуванням усіх архітектурних деталей і функціональних вимог.

Далі створюється топологія мережі за допомогою заздалегідь визначених алгоритмів. Реалізувавши ці алгоритми в програмному коді, додаток може виконувати необхідні процеси генерації топології відповідно до вимог користувача.

Важливим аспектом цього етапу є ретельне кодування відповідно до стандартів програмування та найкращих практик. Це включає підтримку чистого і читабельного коду, використання коментарів для пояснення складних частин програми, уникнення непотрібної складності та застосування ефективних методів для реалізації функціональності.

На етапі тестування виконується ряд дій, щоб переконатися, що топологія мережі є правильною і відповідає вимогам. Етапи цього етапу є наступними:

Тестування додатку: Тестування додатку: Виконуються різні тести для виявлення помилок, невідповідностей і дефектів в додатку. Сюди входить тестування окремих функціональних модулів, тестування інтеграції різних компонентів додатку та системне тестування всього додатку.

Автоматизоване або ручне тестування: Тестування може бути як автоматизованим, так і ручним. Автоматизоване тестування часто використовується для автоматизації повторюваних процесів тестування, тим самим швидше виявляючи помилки і підвищуючи ефективність тестування.

Налагодження та повторне тестування: Налагодження та повторне тестування: Коли виявлено помилку, розробники вносять необхідні зміни до програмного коду, щоб виправити помилку. Потім додаток повторно тестується, щоб підтвердити зміни і переконатися, що додаток працює належним чином.

Процес тестування має важливе значення для забезпечення якості програмного продукту та його сумісності з вимогами користувачів.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

При проектуванні систем, що складають топологію мережі дата-центру, були враховані різні аспекти, що визначають їх характеристики та принципи роботи. Нижче наведено функціональний опис системи, що пояснює її основні характеристики та принципи роботи.

Система автоматично проектує топологію мережі для центру обробки даних. Користувач може вказати вимоги до мережі, такі як кількість серверів, маршрутизаторів і комутаторів, а також визначити параметри зв'язку між ними. На основі цих даних система створює оптимальну топологію мережі, яка враховує ефективне використання ресурсів та забезпечує надійність і масштабованість мережі.

Однією з основних функцій системи є можливість моделювати різні мережеві операції, аналізувати їх ефективність та виявляти потенційні проблеми. Користувачі можуть створювати віртуальні моделі мережі та запускати різні симуляції, щоб спрогнозувати поведінку мережі за різних умов і внести необхідні корективи.

Система надає інструменти для моніторингу та управління мережею в режимі реального часу. Користувачі можуть відстежувати стан і обсяг трафіку мережевих пристроїв, а також налаштовувати і оптимізувати мережу безпосередньо з інтерфейсу системи.

Однією з основних функцій системи є забезпечення захисту даних і безпеки мережі. Система включає в себе механізми аутентифікації, авторизації та контролю доступу, які запобігають несанкціонованому доступу до мережевих ресурсів і забезпечують конфіденційність і цілісність даних.

Система має можливість інтегруватися з іншими системами управління ЦОД, взаємодіяти з іншими компонентами інфраструктури та забезпечувати

уніфіковане управління в межах всієї системи.

Особливу увагу було приділено функції аварійного резервного копіювання та відновлення даних. Система автоматично створює резервну копію конфігурації.

Система автоматично створює резервну копію конфігурації мережевого обладнання та забезпечує можливість її швидкого відновлення в разі потреби.

Таким чином, можливості системи надають широкий спектр інструментів для ефективного проектування, конфігурування та управління мережевою інфраструктурою дата-центру.

Важливою особливістю системи є розширені можливості управління файлами. Користувачі можуть створювати нові файли з різними розширеннями і форматами, зберігати дані конфігурації, результати аналізу мережі та іншу інформацію.

Крім того, передбачені корисні інструменти для відкриття і перегляду файлів, що дозволяють користувачам швидко отримувати доступ до збереженої інформації. Крім того, файли можна редагувати безпосередньо з інтерфейсу програмного забезпечення топології.

Крім того, система може зберігати зміни, внесені в файли, щоб при необхідності можна було відновити попередні версії файлів. Це забезпечує безпеку і надійність в управлінні даними конфігурації та іншою інформацією в системі".

Ця надбудова для керування файлами дозволяє користувачам зберігати, відкривати, редагувати та відновлювати файли, підвищуючи зручність та ефективність керування даними в мережевих інфраструктурах центрів обробки даних.

При розробці системи було враховано низку міркувань, щоб відобразити топологію мережі дата-центру та забезпечити максимальну ефективність і безпеку мережі. Головним з цих міркувань є можливість підтримки різних мережевих протоколів, таких як HTTP і FTP.

Система надає користувачам інструменти для налаштування та оптимізації поведінки цих протоколів. Наприклад, можна налаштувати правила маршрутизації для ефективної маршрутизації HTTP і FTP-трафіку, забезпечуючи оптимальну швидкість і надійність зв'язку між різними частинами мережі.

Крім того, можна налаштувати параметри безпеки для захисту від несанкціонованого доступу до даних, що надсилаються за цими протоколами. Це включає в себе налаштування правил аутентифікації та авторизації, а також контроль доступу до мережевих ресурсів.

Важливою особливістю системи є можливість моніторингу та аналізу поведінки мережевих протоколів у режимі реального часу. Користувачі можуть відстежувати обсяги трафіку, виявляти проблеми та шукати шляхи оптимізації продуктивності мережі і підтримувати її в найкращому стані.

3.2 Розробка структурної схеми

Розробка архітектури системи проектування та побудови мережевих топологій в дата-центрах базується на основних принципах організації мережевої інфраструктури та враховує потреби користувачів у зручних та ефективних інтерфейсах.

Система має модульну архітектуру, яка дозволяє розподілити функції, де кожен компонент відповідає за певний аспект роботи системи. Основними модулями є модуль автоматизованого проектування, модуль моделювання, модуль моніторингу та контролю, модуль безпеки та захисту, модуль інтеграції та модуль резервного копіювання та відновлення.

Кожен модуль взаємодіє з іншими компонентами системи через визначені інтерфейси для забезпечення єдності та цілісності системи. Наприклад, модуль автоматизованого проектування може передавати проектні дані до модуля моделювання для запуску симуляцій та отримання результатів для подальшого аналізу.

Система має інтуїтивно зрозумілий інтерфейс користувача, який дозволяє користувачеві легко використовувати всі функції системи. Цей інтерфейс забезпечує легкий доступ до основних інструментів обробки, конфігурації та аналізу.

Система використовує базу даних для зберігання даних конфігурації мережі, статистики моніторингу, журналів подій та іншої важливої інформації. База даних забезпечує надійне зберігання даних і швидкий доступ до них для подальшого аналізу.

Система розроблена з урахуванням потреб масштабованості, що дозволяє збільшувати пропускну здатність і кількість пристроїв, які вона обслуговує, без істотних змін в структурі або алгоритмах роботи системи.

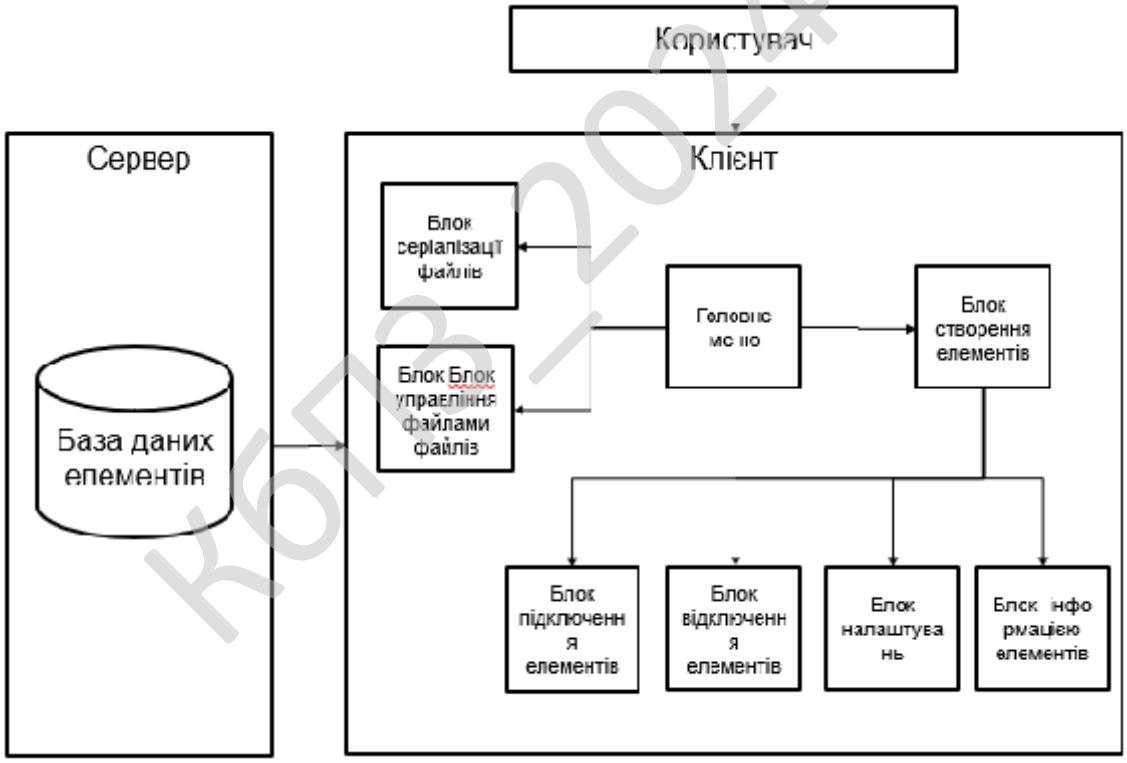

Рисунок 3.1 - Структурна схема

Система має можливість інтеграції з іншими системами управління ЦОД, що дозволяє взаємодіяти з іншими компонентами інфраструктури та забезпечує єдність управління в рамках всієї системи.

Система включає в себе механізми захисту даних, які забезпечують високу надійність за рахунок резервного копіювання та відновлення даних, а також використання сучасних методів захисту від кібератак та витоків інформації.

Тому розробка структурної схеми системи базується на принципах модульності, інтерактивності, дружніх інтерфейсів, масштабованості та безпеки, що забезпечує ефективну та надійну роботу системи в умовах сучасного дата-центру.

Ця схема показує послідовність модулів та їх взаємозв'язки. Користувацький інтерфейс є початковою точкою, через яку користувач взаємодіє з програмним забезпеченням. Потім дані вводяться для аналізу вимог, після чого виконуються модулі планування мережі та моделювання мережі. Результати можуть бути візуалізовані та піддані валідації. Після цього може бути згенерована документація. Нарешті, програмне забезпечення може бути інтегроване з іншими системами для подальшого використання або обміну даними.

3.3 Розробка функціональної схеми

Функціональна схема системи проектування та побудови мережевих топологій в дата-центрах відображає основні функції та процеси, які виконуються системою. Ось загальний опис:

– Аналіз вимог користувачів: Система отримує вхідні дані щодо вимог користувачів щодо мережевої інфраструктури дата-центру.

– Автоматизоване проектування мережі: Система виконує автоматизований процес проектування мережі з урахуванням вхідних вимог та параметрів.

– Моделювання мережі: Система створює модель мережі для проведення симуляцій та оцінки різних варіантів топологій.

– Моніторинг та контроль: Система здійснює моніторинг роботи мережі та забезпечує контроль за її станом та ефективністю.

– Захист та безпека: Система забезпечує захист мережевої інфраструктури від кіберзагроз та витоків даних.

– Інтеграція з іншими системами: Система взаємодіє з іншими системами управління дата-центром для забезпечення єдності та сумісності роботи.

– Резервне копіювання та відновлення: Система здійснює резервне копіювання даних та має механізми відновлення у разі виникнення аварійних ситуацій.

– Інтерфейс користувача: Система надає інтуїтивно зрозумілий інтерфейс для взаємодії з користувачем та управління всіма функціями системи.

– База даних: Система використовує базу даних для зберігання конфігурацій, статистики, журналів подій та іншої важливої інформації.

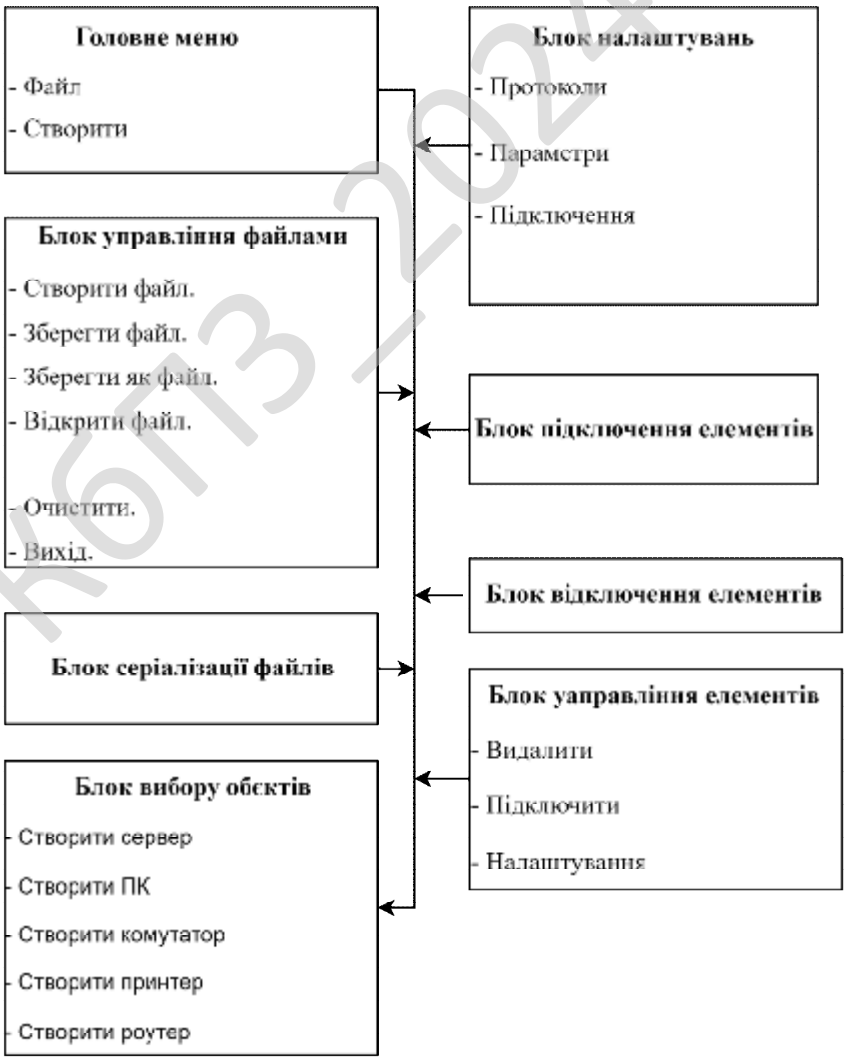

Рисунок 3.2 - Функціональна схема

Ця функціональна схема дозволяє керувати всіма аспектами проектування та експлуатації мережевої інфраструктури дата-центру, забезпечуючи ефективну та безпечну роботу системи.

Схема детально описує кожну з функцій системи та їх роль у процесі проектування та побудови мережевих топологій в дата-центрах.

– Аналіз вимог користувачів: Система отримує вхідні дані щодо вимог користувачів щодо мережевої інфраструктури дата-центру.

– Автоматизоване проектування мережі: Система виконує автоматизований процес проектування мережі з урахуванням вхідних вимог та параметрів.

– Моделювання мережі: Система створює модель мережі для проведення симуляцій та оцінки різних варіантів топологій.

– Моніторинг та контроль: Система здійснює моніторинг роботи мережі та забезпечує контроль за її станом та ефективністю.

– Захист та безпека: Система забезпечує захист мережевої інфраструктури від кіберзагроз та витоків даних.

– Інтеграція з іншими системами: Система взаємодіє з іншими системами управління дата-центром для забезпечення єдності та сумісності роботи.

– Резервне копіювання та відновлення: Система здійснює резервне копіювання даних та має механізми відновлення у разі виникнення аварійних ситуацій.

– Інтерфейс користувача: Система надає інтуїтивно зрозумілий інтерфейс для взаємодії з користувачем та управління всіма функціями системи.

3.4 Розробка діаграми процесів

Розробимо процедуру розробки програмного забезпечення, яке захищає дані методами шифрування. Основна проблема полягає в реалізації методів для синхронізації взаємодій, станів і дій у системі.

Використання діаграми процесу полегшує ідентифікацію послідовності дій, які будуть виконані, і порядку, в якому вони будуть виконуватися.

Натомість системне програмне забезпечення призначене для створення набору інструментів для створення та керування топологій центрів обробки даних з точки зору програмного забезпечення. Це стосується проектування, ефективності та оптимізації комунікаційних мереж.

Програмне забезпечення для мережевої інженерії зазвичай містить інструменти, які полегшують побудову моделі, візуальний аналіз, оцінку пропускної здатності, планування резервування та інші функції, необхідні для підвищення ефективності мережевої інфраструктури.

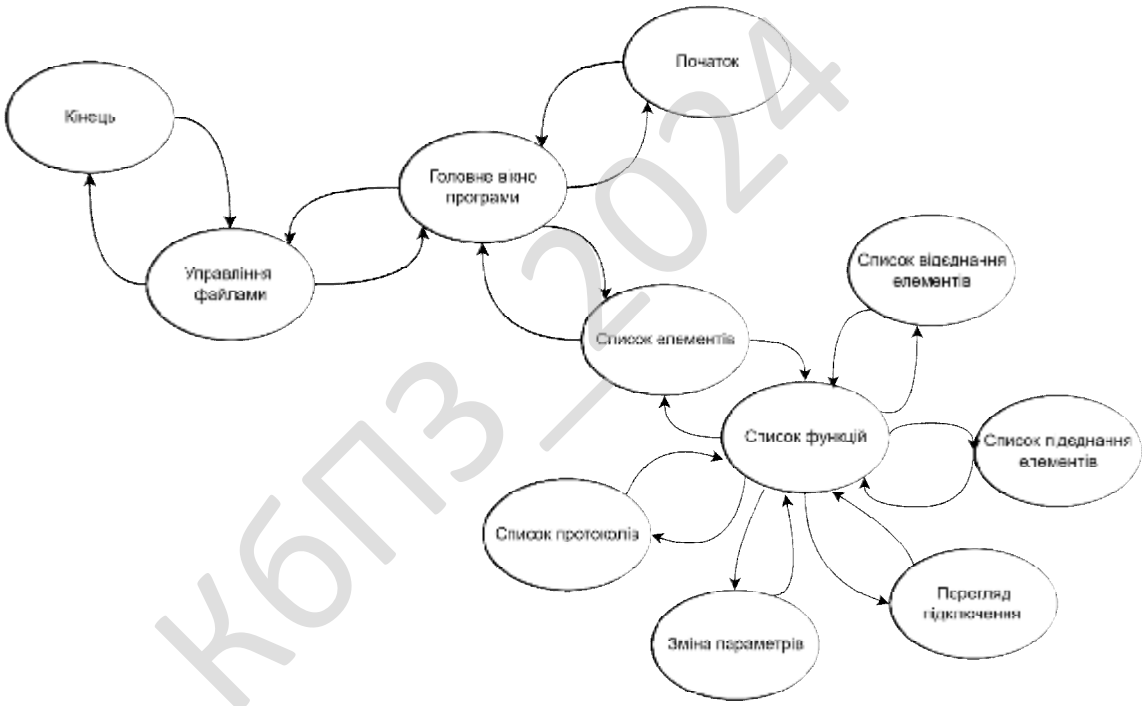

Рисунок 3.3 - Діаграма процесів

Програмне забезпечення може використовуватися в обох сферах, але з різними цілями і фокусом на різних аспектах IT.

Процес, показаний на схемі, є важливим етапом проектування та оптимізації топології мережі центру обробки даних. Створення окремих мережевих компонентів, таких як комп'ютери, комутатори, принтери та маршрутизатори, а також конфігурація компонентів з відповідними мережевими протоколами та

налаштуваннями починається з наступних кроків. Потім ці компоненти з'єднуються один з одним, визначаються елементи мережі та створюється топологія мережі.

Після створення мережі починається етап аналізу даних, на якому оцінюється стан мережі та обирається найкращий алгоритм оптимізації. Обраний алгоритм використовується для створення оптимальної топології мережі та оптимізується для підвищення ефективності та продуктивності.

Всі ці процеси тісно пов'язані між собою. Наприклад, результати аналізу даних впливають на вибір алгоритмів для побудови та подальшої оптимізації топології мережі. Вибір мережевих елементів залежить від результатів протоколу та параметризації. Крім того, успішне з'єднання мережевих компонентів вимагає правильного налаштування їхніх протоколів і параметрів.

Інтеграція цих процесів призводить до створення ефективної та надійної мережевої інфраструктури, яка відповідає потребам центру обробки даних. Кожен крок у цьому процесі важливий і взаємодіє з іншими для досягнення найкращих результатів. При роботі з мережевою топологією центру обробки даних також важливо вміти ефективно працювати з файлами. Файли можна створювати, відкривати і зберігати, забезпечуючи зручний інтерфейс для зберігання важливих мережевих даних і налаштувань для подальшого перегляду.

Створюючи файли, можна зберігати нові конфігурації та інші дані, пов'язані з топологією мережі, для подальшого використання та аналізу. Наприклад, можна створити файл для збереження результатів аналізу мережі або конфігурації протоколу. Коли файл відкрито, користувач може отримати доступ до збережених даних і конфігурацій, щоб переглянути їх, внести зміни або проаналізувати їх далі. Наприклад, файл конфігурації мережі можна відкрити, щоб переглянути і відредагувати налаштування протоколу. Збереження файлу дозволяє зберегти важливі зміни і дані для подальшого використання. Наприклад, після зміни конфігурації мережі ви можете зберегти файл, щоб зберегти зміни і забезпечити надійність.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Програмне забезпечення для проектування та побудови топологій мережі реалізує широкий спектр функцій для створення, збереження та аналізу мережевих топологій. Ось основний функціонал програми:

Опції головного меню:

Створити файл: Дозволяє користувачеві створити новий файл для роботи з мережевою топологією.

```
    createFileItem.setOnAction(event -> {
            FileChooser fileChooser = new FileChooser();
            FileChooser.ExtensionFilter extFilter = new
FileChooser.ExtensionFilter("Сериализованные файлы (*.ser)", "*.ser");
            fileChooser.getExtensionFilters().add(extFilter);
            File file = fileChooser.showSaveDialog(primaryStage);

            if (file != null) {
                try (ObjectOutputStream ignored = new ObjectOutputStream(new
FileOutputStream(file))) {
                    NAME_FILE_NEW = file.getAbsolutePath();
                } catch (IOException e) {
                    e.printStackTrace();

}
            }
        });
```

Алгоритм:

– Вибір файлу для створення за допомогою FileChooser;

– Після вибору файлу програма створює ObjectOutputStream, який дозволяє записувати об'єкти в файл.

– Об'єкт, який потрібно записати, може бути переданий ObjectOutputStream для запису у файл. У цьому прикладі він неявно позначений як ignored.

– Після успішного запису виводиться повідомлення з шляхом файлу, в який було здійснено запис.

Зберегти файл: Зберігає поточний стан мережевої топології у вже існуючий файл.

Зберегти як файл: Зберігає поточний стан мережевої топології у новому файлі.

Відкрити файл: Дозволяє користувачеві відкрити наявний файл з мережевою топологією для подальшого редагування.

```
public static void ImageInfoSerializationExample(String filePath) {
    ArrayList<ImageInfo> arrayList = new ArrayList<>();

    for (ImageInfo imageInfo1 : Constants.imageList) {
        ImageInfo imageInfo = new ImageInfo();

        imageInfo.setType(imageInfo1.getType());
        imageInfo.setName(imageInfo1.getName());
        imageInfo.setX(imageInfo1.getX());
        imageInfo.setY(imageInfo1.getY());
        imageInfo.setMovable(imageInfo1.isMovable());
        imageInfo.setConnectedItems(imageInfo1.getConnectedItem());
        imageInfo.setIdLine(imageInfo1.getIdLine());
        imageInfo.setConnectedLiness(imageInfo1.getConnectedLines());
        imageInfo.setImageView(imageInfo1.getImageView());

        arrayList.add(imageInfo);
    }

    try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(filePath))) {
      outputStream.writeObject(arrayList);
          } catch (IOException e) {
        e.printStackTrace();
        System.err.println("Помилка");
    }
}
```

Цей метод призначений для серіалізації об'єктів ImageInfo у файл. Він створює новий список ArrayList, де будуть зберігатися скопійовані об'єкти ImageInfo. Потім проходиться по кожному об'єкту imageInfo1 зі списку Constants.imageList, копіює його дані у новий об'єкт imageInfo і додає його до arrayList. Після цього відкривається потік ObjectOutputStream, що дозволяє записувати об'єкти у файл, та записується arrayList у цей файл.

Метод призначений для десеріалізації об'єктів ImageInfo з файлу. Він спочатку очищує списки Constants.lineList і Constants.imageList, а потім відкриває потік ObjectInputStream, щоб прочитати об'єкти з файлу. Після цього зчитується об'єкт із файлу, перевіряється, чи є він списком, та кожен об'єкт ImageInfo

додається до imageInfoList. Потім копіюються дані з кожного об'єкту ImageInfo з imageInfoList в новий об'єкт imageInfo, який додається до imageList. Після цього викликається метод drawOpenFile(), який відображає завантажені об'єкти на екрані.

```
public static void ImageInfoDeserializationExample(String filePath) {
    Constants.lineList.clear();
    Constants.imageList.clear();
    List<ImageInfo> imageInfoList = new ArrayList<>();

    try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(filePath))) {
        Object object = inputStream.readObject();

        if (object instanceof List<?> list) {
            // Теперь вы можете безопасно использовать list
            for (Object item : list) {
                if (item instanceof ImageInfo imageInfo) {
                    imageInfoList.add(imageInfo);
                }
            }
        } else {
            System.err.println("Line 73(OpenFile). list is not a ImageInfo");
        }

        for (ImageInfo imageInfo1 : imageInfoList) {
            ImageInfo imageInfo = new ImageInfo();

            // Копируем данные из imageInfo1 в новый объект imageInfo
            imageInfo.setType(imageInfo1.getType());
            imageInfo.setName(imageInfo1.getName());
            imageInfo.setX(imageInfo1.getX());
            imageInfo.setY(imageInfo1.getY());
            imageInfo.setMovable(imageInfo1.isMovable());
            imageInfo.setConnectedItems(imageInfo1.getConnectedItem());
            imageInfo.setIdLine(imageInfo1.getIdLine());
            imageInfo.setConnectedLiness(imageInfo1.getConnectedLines());
            imageInfo.setImageView(imageInfo1.getImageView());

            imageList.add(imageInfo);
        }

        drawOpenFile();
        System.out.println("ImageInfo успешно прочитаны из файла: " +
filePath);

    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
        System.err.println("Ошибка при чтении объектов ImageInfo из файла");
    }
}
```

– Очистити поле: Очищує робочу область від усіх елементів мережевої топології.

– Вийти: Завершує роботу програми.

Опції меню створення елементів:

– Створити сервер: Додає на робочу область новий об'єкт, що відповідає серверу.

– Створити ПК: Додає на робочу область новий об'єкт, що відповідає персональному комп'ютеру.

– Створити роутер: Додає на робочу область новий об'єкт, що відповідає маршрутизатору.

– Створити комутатор: Додає на робочу область новий об'єкт, що відповідає комутатору.

– Створити принтер: Додає на робочу область новий об'єкт, що відповідає принтеру.

Опції контекстного меню елементів:

– Видалити: Видаляє обраний елемент з мережевої топології.

– Інформація: Відображає інформацію про обраний елемент, таку як IP-адреса, MAC-адреса, тощо.

– Підключити: Призначена для побудови з'єднань між обраними елементами мережевої топології.

– Налаштування: Відкриває вікно з налаштуваннями обраного елемента, де можна налаштувати HTTP, FTP протоколи, а також IPv4 і IPv6 конфігурації.

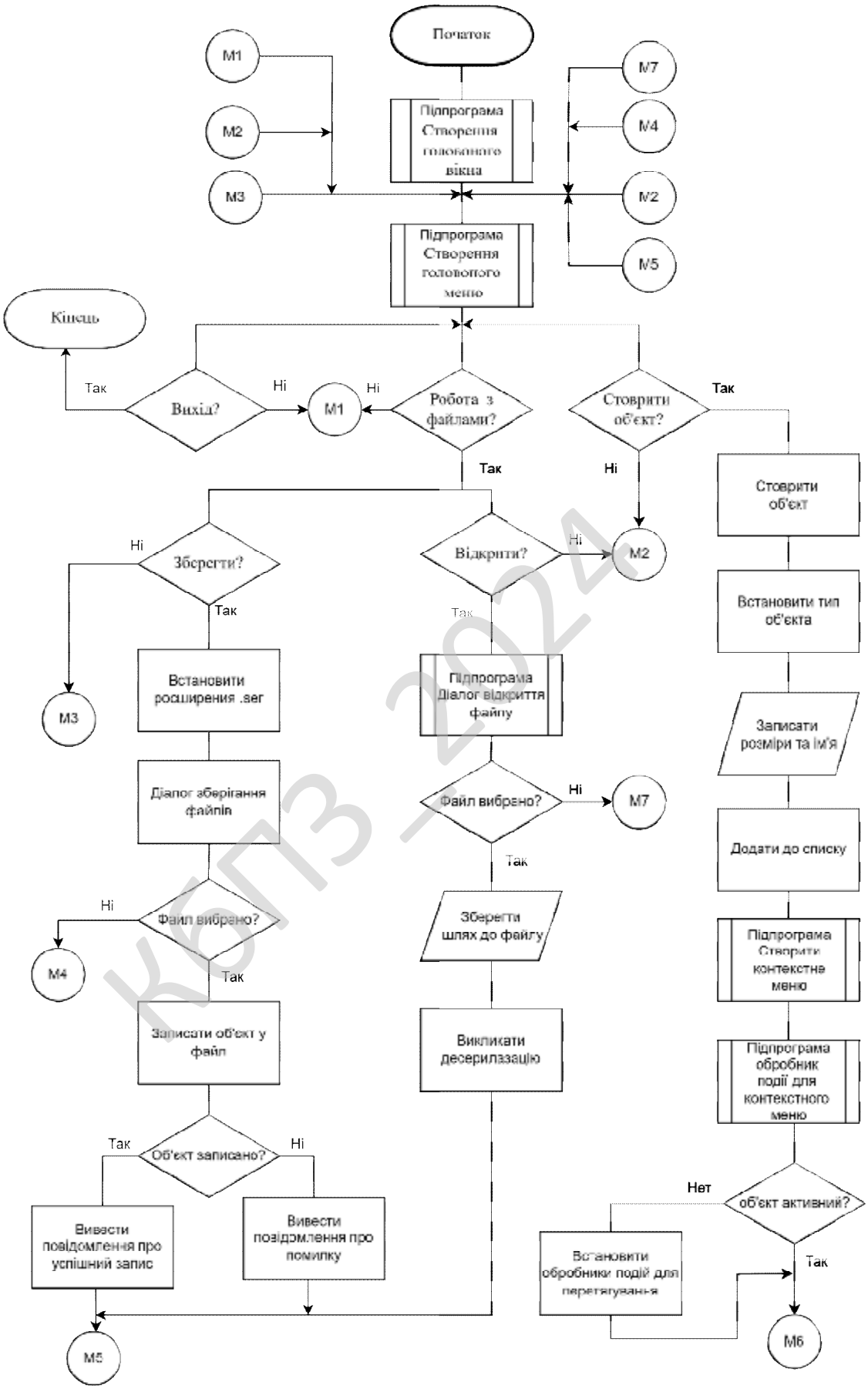

Рисунок 4.1 – Блок схема програми

4.2 Розбір алгоритмів програми

Алгоритм створення елементів мережі:

– Користувач обирає тип елемента з меню.

– Програма створює відповідний об'єкт і розміщує його на робочій області.

Метод createObject. Цей метод створює об'єкти (зображення) на основі шляху path до файлу зображення. Він завантажує зображення з файлу, створює ImageView, встановлює розміри та обмеження, створює об'єкт ImageInfo для збереження даних про зображення, створює контекстне меню та встановлює обробники подій для виводу цього контекстного меню. Потім створює VBox, додає до нього ImageView та Label, встановлює обробники подій для перетягування, і додає цей VBox на інтерфейс користувача.

```
public static void createObject(String path) {
          InputStream imageStream = MenuCreator.class.getResourceAsStream(path);

          String type = parseImageType(path);

          if (imageStream != null) {
              // Завантажуємо зображення з файлу
              Image image = new Image(imageStream);
              ImageView imageView = new ImageView(image);

              // Встановлюємо розміри та обмеження збереження пропорцій
              imageView.setFitWidth(50);
              imageView.setFitHeight(50);
              imageView.setPreserveRatio(true);
              imageView.setFitWidth(100);

              // Створюємо об'єкт ImageInfo та додаємо його до списку
              ImageInfo imageInfo = new ImageInfo();
              imageInfo.setImageView(imageView);
              imageInfo.setType(type);
              imageInfo.setName(type + " " + imageList.size());

              switch (type) {
                  // Створюємо об'єкти відповідно до типу зображення
                  case "comutator" -> imageInfo.setCommutator(new
      Commutator(imageInfo.getName()));
                  case "pc" -> imageInfo.setPC(new PC(imageInfo.getName()));
                  case "printer" -> imageInfo.setPrinter(new
      Printer(imageInfo.getName()));
                  case "router" -> imageInfo.setRouter(new
      Router(imageInfo.getName()));
                  case "server" -> imageInfo.setServer(new
      Server(imageInfo.getName()));
                  default -> {}
              }

              imageList.add(imageInfo);
```

```
        // Створюємо контекстне меню та встановлюємо обробник подій для виводу
контекстного меню
        ContextMenu contextMenu = createContextMenu(imageInfo);
        imageView.setOnContextMenuRequested(event ->
contextMenu.show(imageView, event.getScreenX(), event.getScreenY()));

        // Створюємо VBox, додаємо в нього ImageView та Label, встановлюємо
обробники подій для перетягування
        VBox imageBox = new VBox();
        imageBox.setMaxWidth(50);
        imageBox.setMaxHeight(50);
        imageBox.getChildren().addAll(imageView, new
Label(imageInfo.getName()));

        if (imageInfo.isMovable()) {
            setDragEventHandlers(imageBox, imageInfo);
        }

        // Додаємо об'єкт на інтерфейс користувача
        imageContainer.getChildren().add(imageBox);
    } else {
        System.err.println("Unable to load image: " + path);
    }
}
```

Метод createCircleLine. Цей метод створює кільця (кружки) на кінцях лінії. Він створює два Circle, встановлює їх координати, встановлює кольори заливки, присвоює їм ідентифікатори, додає їх до списку circleList і додає їх на інтерфейс користувача.

```
public static void createCircleLine(Line line){
    // Создаем кружки для краев линии
    Circle circleStart = new Circle(3);
    Circle circleEnd = new Circle(3);

    // Устанавливаем координаты кружков
    circleStart.setCenterX(line.getStartX());
    circleStart.setCenterY(line.getStartY());
    circleEnd.setCenterX(line.getEndX());
    circleEnd.setCenterY(line.getEndY());

    // Устанавливаем цвет заливки кружков по значениям RGB
    circleStart.setFill(Color.rgb(118, 237, 71));
    circleEnd.setFill(Color.rgb(118, 237, 71));

    circleStart.setId(line.getId());
    circleEnd.setId(line.getId());

    circleList.add(circleStart);
    circleList.add(circleEnd);

    imageContainer.getChildren().addAll(circleStart, circleEnd);
}
```

Ці методи разом допомагають створити та відображати об'єкти на інтерфейсі користувача з урахуванням їх типів та зв'язків.

Алгоритм з'єднання елементів мережі:

– Користувач обирає опцію "Підключити" у контекстному меню елемента.

– Відображається вікно з вибором елемента, який потрібно підключити.

– Після вибору, програма робить з'єднання між обраними елементами та візуально відображає лінію з'єднання.

Метод disconnectItem. Метод відключає елементи один від одного. Він видаляє кільця, які позначають кінці лінії, видаляє зв'язки між елементами, перевіряє, чи залишився ще підключений елемент, і видаляє лінії, якщо вони не потрібні. Якщо підключений елемент не залишився, перевіряється наявність інших підключених елементів, і якщо вони є, створюється новий елемент на місці видаленого.

```
public static void disconnectItem(ImageInfo imageInfo) {
          // Отримуємо підключений елемент та список підключених елементів
          ImageInfo connectedImageInfo = imageInfo.getConnectedItem().get(0);
          List<ImageInfo> listItem = imageInfo.getConnectedItem();

          // Формуємо ідентифікатор лінії
          String idLine = imageInfo.getName() + " " + connectedImageInfo.getName();
          String idLine2 = connectedImageInfo.getName() + " " + imageInfo.getName();

          // Видаляємо кільця, які позначають кінці лінії
          circleList.removeIf(circle -> circle.getId().equals(idLine) ||
      circle.getId().equals(idLine2));

          // Від'єднуємо елементи один від одного
          imageInfo.removeConnectItem(connectedImageInfo);
          connectedImageInfo.removeConnectItem(imageInfo);

          // Перевіряємо, чи не залишився підключений елемент
          if (!connectedImageInfo.getConnectedItem().isEmpty()) {
              String newIdLine = "";
              if(!connectedImageInfo.getConnectedLines().isEmpty()){
                  newIdLine = connectedImageInfo.getConnectedLines().get(0).getId();
              }
              connectedImageInfo.setIdLine(newIdLine);
              connectedImageInfo.setMovable(false);
              connectedImageInfo.setDisabled(true);
          }

          // Видаляємо лінії, якщо вони не потрібні
          if (imageInfo.getIdLine().equals("") &&
      !imageInfo.getConnectedLines().isEmpty()) {
              imageInfo.clearConnectedLines();
          }

          // Перевіряємо, чи не залишився підключений елемент
          if (connectedImageInfo.getConnectedLines().isEmpty()){
              // Перевіряємо, чи є підключені елементи до позначеного елемента
              if (!connectedImageInfo.getConnectedItem().isEmpty()){
                  // Шукаємо індекс позначеного елемента
```

```
                int index = -1;
                for (int i = 0; i < imageList.size(); i++) {
                    ImageInfo imgIn = imageList.get(i);
                    if (imgIn.getName().equals(connectedImageInfo.getName())) {
                        index = i;
                        break;
                    }
                }
                // Створюємо новий елемент на місці видаленого
                ImageInfo imageInfo1 = new ImageInfo();
                imageInfo1.setType(connectedImageInfo.getType());
                imageInfo1.setName(connectedImageInfo.getName());
                imageInfo1.setX(connectedImageInfo.getX());
                imageInfo1.setY(connectedImageInfo.getY());
                imageInfo1.setMovable(true);
                imageInfo1.setDisabled(false);
                switch (imageInfo1.getType()) {
                    case "server" -> imageInfo1.setImageView(createImageView(
PATH_SERVER));
                    case "pc" -> imageInfo1.setImageView(createImageView(
PATH_PC));
                    case "comutator" -> imageInfo1.setImageView(createImageView(
PATH_COMUTATOR));
                    case "printer" -> imageInfo1.setImageView(createImageView(
PATH_PRINTER));
                    case "router" -> imageInfo1.setImageView(createImageView(
PATH_ROUTER));
                }
                // Оновлюємо список елементів
                if (index != -1) {
                    imageList.set(index, imageInfo1);
                } else {
                    try {
                        throw new Exception("Index == -1");
                    } catch (Exception e) {
                        throw new RuntimeException(e);
                    }
                }
            }
        }
        // В іншому випадку перевіряємо на неспівпадіння підключених елементів
        else {
            removeInconsistentConnectedItems(imageInfo);
            removeInconsistentConnectedItems(connectedImageInfo);
        }
        // Оновлюємо вміст контейнера
        redrawImageContainer();
    }
```

Метод createImageView. Цей метод створює інтерфейс ImageView зображення з вказаного шляху та встановлює його параметри.

```
    public static ImageView createImageView(String path){
        InputStream imageStream = MenuCreator.class.getResourceAsStream(path);
        assert imageStream != null;
        ImageView imageView = new ImageView(new Image(imageStream));

        imageView.setFitWidth(50);
        imageView.setFitHeight(50);

        imageView.setPreserveRatio(true);
```

```
        imageView.setFitWidth(100);

        return imageView;
    }
```

Обидва ці методи спільно використовуються для реалізації взаємодії з елементами інтерфейсу користувача для відображення, підключення та роз'єднання об'єктів.

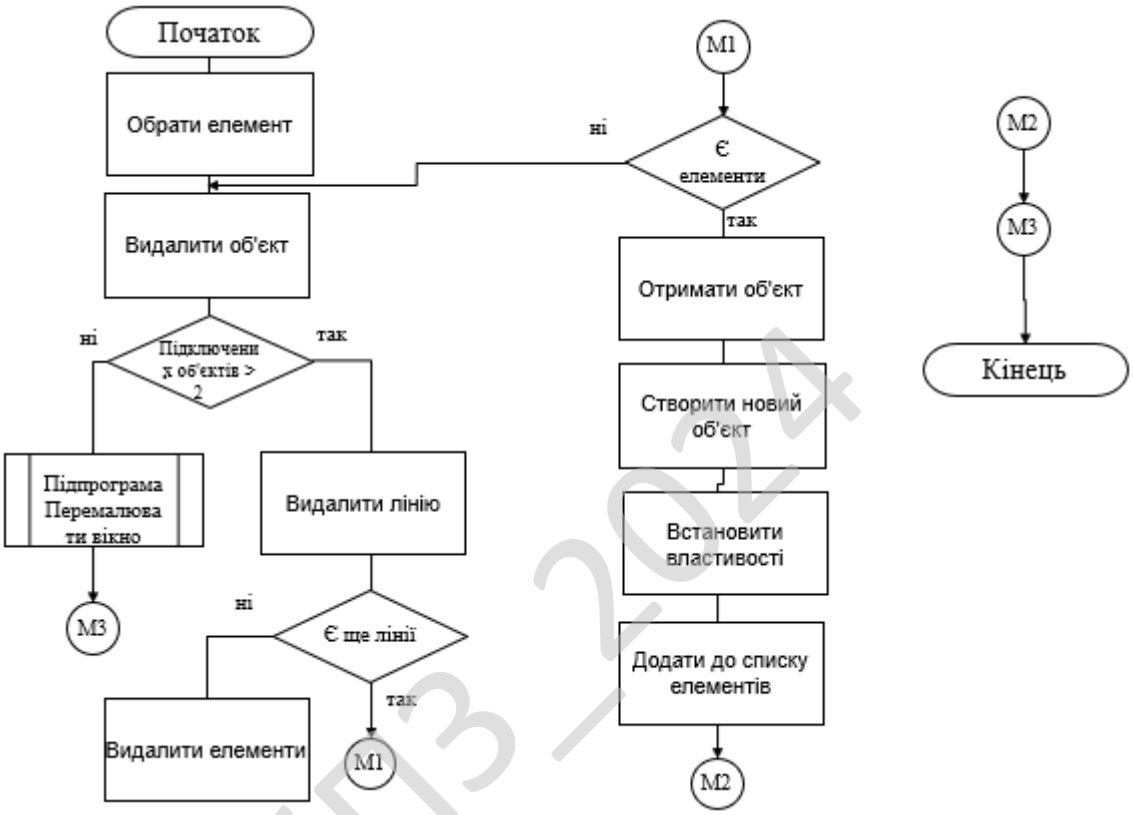

Рисунок 4.2 – Блок схема відключення елементів

4.3 Захист розробленого програмного забезпечення

Оцінка потенційних загроз Перший етап передбачає проведення аналізу ризиків для виявлення потенційних загроз безпеці програмного забезпечення. Сюди входить виявлення вразливостей коду, потенціалу несанкціонованого доступу та можливих атак зловмисників. Оцінка цих потенційних загроз може допомогти розробникам вжити заходів для підвищення безпеки своїх додатків.

Провести аудит програмного коду, щоб виявити потенційні вразливості та

недоліки.

– Використовуйте інструменти автоматизованого сканування вразливостей для виявлення потенційних проблем безпеки.

– Аналізувати зовнішні загрози, такі як потенційні атаки зловмисників.

Використовувати методи шифрування: використовуйте сучасні методи шифрування для захисту конфіденційної інформації, що обробляється програмним забезпеченням. Шифрування захищає від несанкціонованого доступу шляхом перетворення даних у формат, який ніхто не зможе зрозуміти без відповідного ключа.

– Впровадив сучасні алгоритми шифрування, які захищають конфіденційні дані, такі як Advanced Encryption Standard (AES) та Rivest-Shamir-Adleman (RSA).

– Забезпечив належне управління ключами шифрування, щоб убезпечити процес шифрування та дешифрування.

Впровадити механізми автентифікації: для забезпечення безпеки програмного забезпечення важливо впровадити механізми автентифікації користувачів. Це гарантує перевірку особи користувача та обмеження доступу до конфіденційної інформації лише уповноваженим особам з відповідними правами доступу.

– Розробка механізму автентифікації, такі як введення пароля, біометричні методи та двофакторна автентифікація.

– Використовую сервіси автентифікації, такі як Lightweight Directory Access Protocol (LDAP) та OAuth для перевірки особи користувача.

Надати оновлення та виправлення: безпека програмного забезпечення потребує постійного вдосконалення. Тому важливо регулярно випускати оновлення та патчі для усунення виявлених вразливостей. Це допомагає запобігти потенційним атакам і забезпечити стабільну роботу програми.

– Налаштування механізми автоматичного оновлення програмного забезпечення та його компонентів.

– Налагоджений процес застосування патчів для усунення виявлених

вразливостей і недоліків.

Моніторинг та аудит безпеки: постійний моніторинг та аудит безпеки є важливим аспектом забезпечення безпеки програмного забезпечення. Це передбачає спостереження та аудит заходів безпеки для виявлення потенційних порушень або слабких місць у заходах безпеки. Це єдиний спосіб забезпечити надійний рівень безпеки додатків у центрах обробки даних.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Метою програми аналізу топології мережі є підвищення ефективності управління мережевими ресурсами та забезпечення стабільності роботи комп'ютерної інфраструктури. Основними завданнями програми є

Підвищення ефективності управління мережею: Програмне забезпечення швидко і точно аналізує топологію мережі, що дозволяє виявити проблемні ділянки і вчасно вжити відповідних заходів.

Скорочення часу реагування на мережеві проблеми: Швидкий доступ до інформації про топологію мережі дозволяє швидко реагувати на проблеми та уникати несподіваних відключень мережі.

Підвищення надійності мережевої інфраструктури: Аналізуючи топологію мережі, можна виявити і захистити потенційні вразливості, зменшити ймовірність інцидентів і підвищити стабільність роботи мережі.

Підвищення продуктивності IT-персоналу: Оптимізувавши процес аналізу топології мережі, IT-персонал може витрачати менше часу на рутинні завдання і більше часу на стратегічні аспекти управління мережею.

Покращення взаємодії з користувачем: Інтуїтивно зрозумілий інтерфейс програмного забезпечення покращує комунікацію між IT-персоналом та іншими користувачами мережевих ресурсів.

Ці цілі спрямовані на забезпечення оптимальної роботи мережевої інфраструктури та високої ефективності для організації.

Мінімальні системні вимоги для програмного забезпечення мережевої топології такі: Windows 10 або новішої версії, macOS 10.12 або новішої версії, або Linux з версією 4.14 або новішої версії; процесор з тактовою частотою 1 ГГц або вище, архітектура x86-64 для Windows і Linux, і x86 або Apple Silicon для macOS повинні бути доступні. Для оптимальної продуктивності потрібно мінімум 2 ГБ оперативної пам'яті і 500 МБ вільного місця на жорсткому диску.

Відеокарта повинна підтримувати OpenGL 2.0 або новішу версію, а роздільна здатність екрану має бути не менше 1280 x 720 пікселів. Для запуску програми потрібні миша і клавіатура.

Розглянемо інтерфейс програми. Головне вікно розробленого програмного забезпечення приведено на рисунку 5.1. На горі зображено головні опції програми.

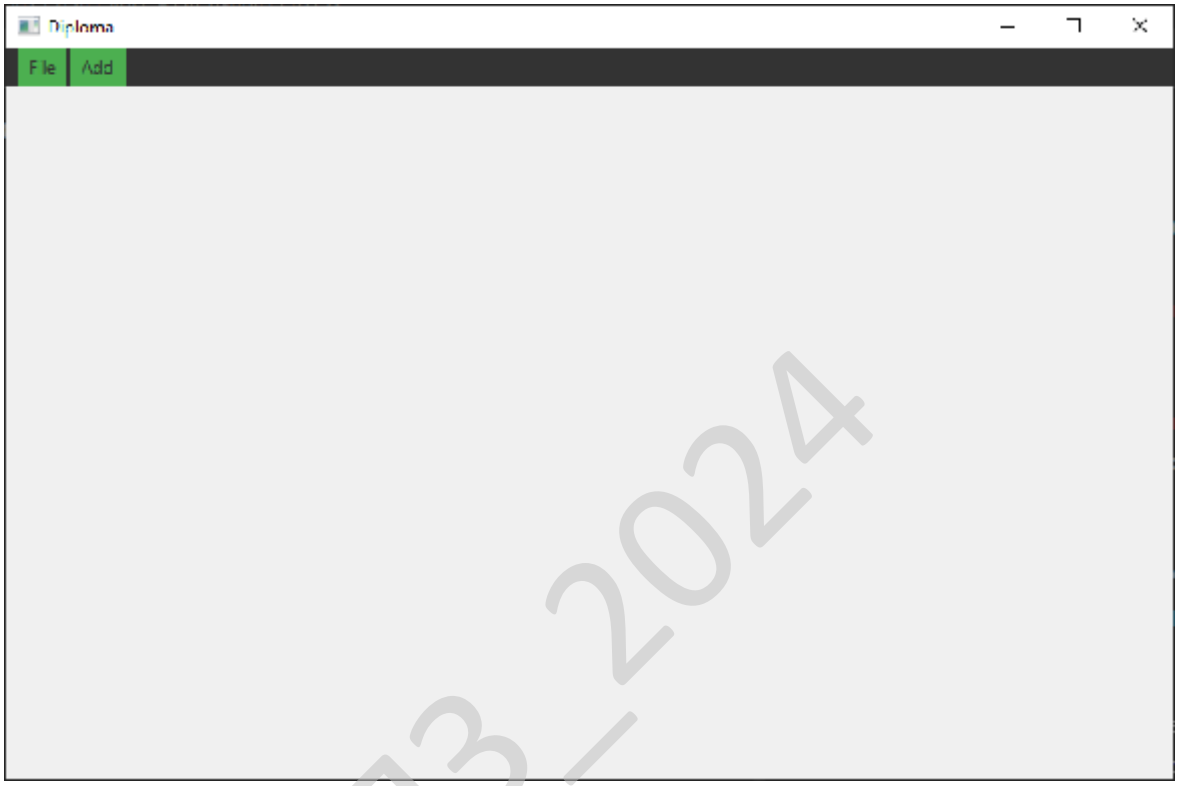

Рисунок 5.1 - Головне вікно програми

Для роботи призначенні кнопки:

- «File».
- «Add».

Після створення елементів (рисунок 5.2) відображаються елементи.

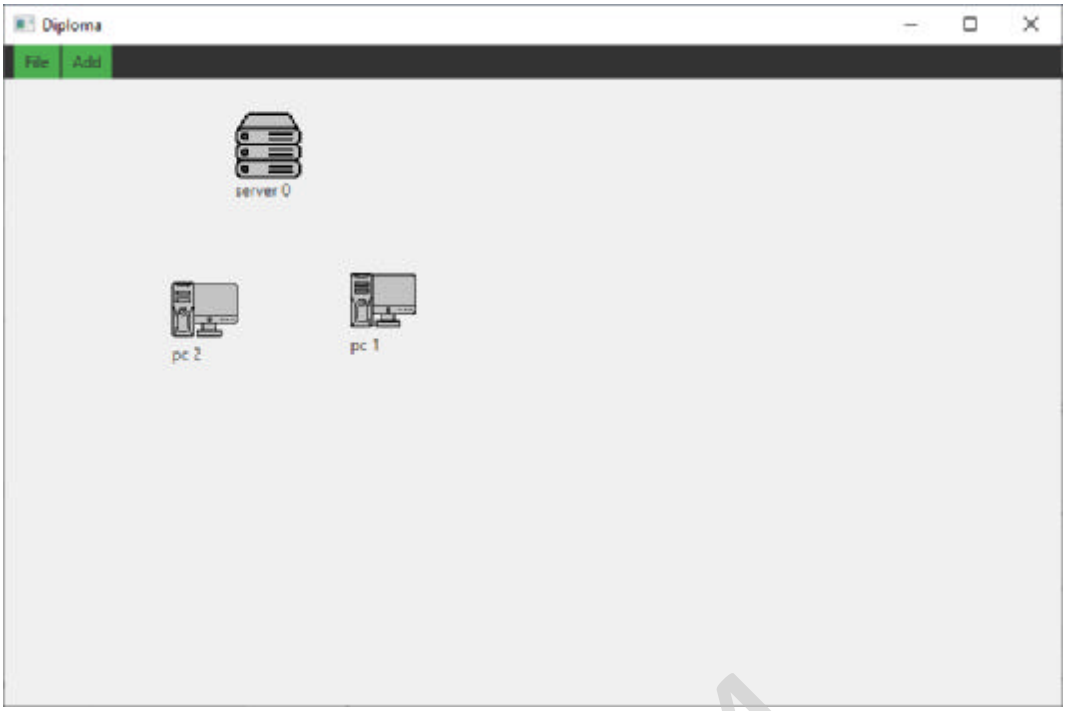

Рисунок 5.2 - Створення елементів

На рисунку 5.3 відображено з’єднання елементів мережі

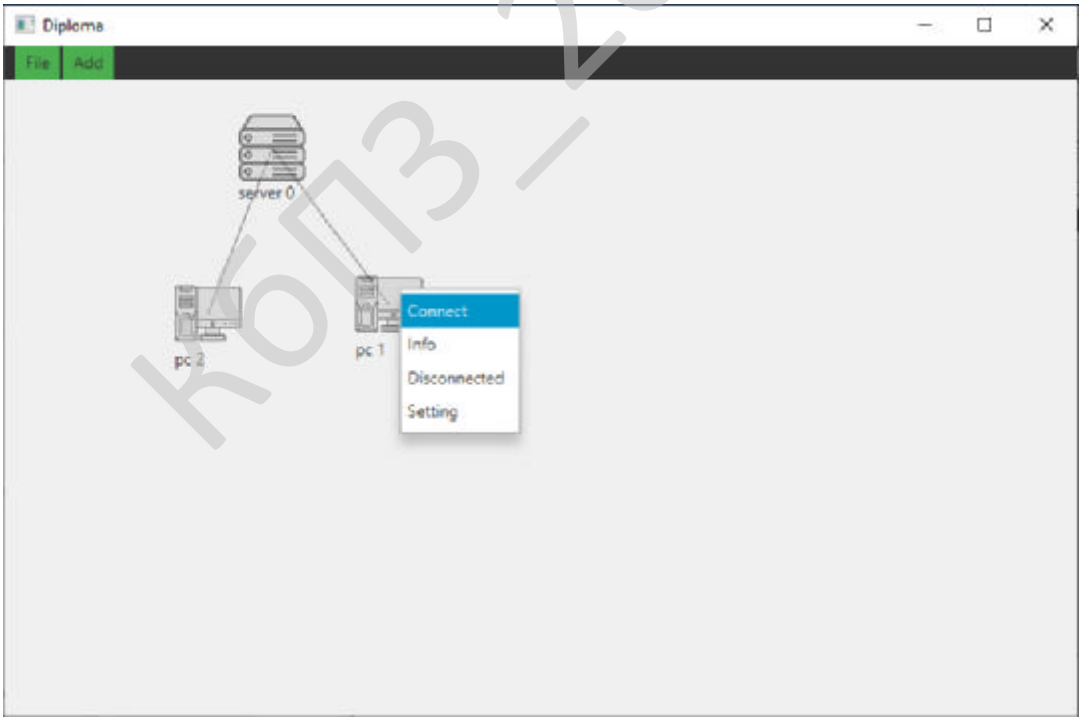

Рисунок5.3 - З’єднання елементів

При натисканні на об’єкт правою кнопкою в контекстному меню є:

- Connect.

- Info.
- Disconnect.
- Setting.

Connect дозволяє підключати елементи один до одного або один до багатьох. При наявному підключенню до елемента його колір змінюється також перестає працювати функцію переміщення підключеного елемента.

На рисунку 5.4 зображено вікно налаштування для об’єкта:

– Services – призначений для налаштування протоколів.
– Desktop - призначений для налаштування вибраного об’єкта.
– Connection – для входу в браузер або термінал.

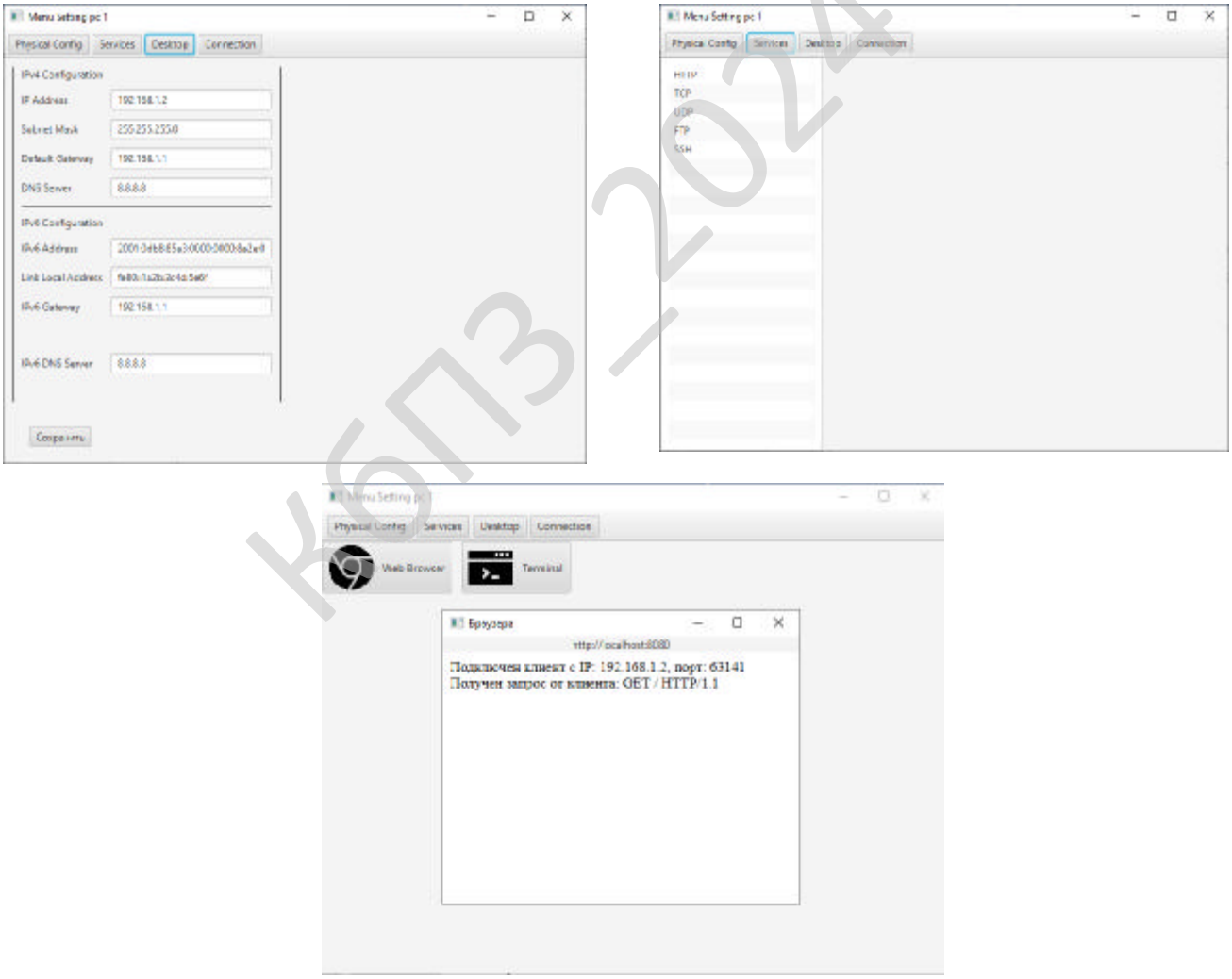

Рисунок 5.4 - Вікно налаштування

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної роботи, призначено для системи кібербезпеки для захищеного обміну даними у мережі Інтернет.

Для вирішення поставленої мети було проведено:

– Дослідження існуючих систем захищеного обміну даних для комунікування по мережі Інтернет.

– Розробку алгоритмів системи кібербезпеки для захищеного обміну даними у мережі Інтернет.

– Реалізацію програмного забезпечення системи кібербезпеки для захищеного обміну даними у мережі Інтернет.

Реалізовані під час виконання кваліфікаційної роботи алгоритми дозволяють успішно вирішувати завдання шифрування та передачі даних у мережі Інтернет.

Розроблене програмне забезпечення представляє собою веб-додаток, що можна використовувати практично на будь-якому сучасному комп'ютері, підключеному до інтернету, та при наявності встановленого Інтернет-браузера.

Програмне забезпечення розроблялося у об'єктно-орієнтованій парадигмі, що робить його гнучким та зручним для підтримки та масштабування.

Для розробки програмного забезпечення системи кібербезпеки захищеного обміну даними у мережі Інтернет використовувалися мови програмування JavaScript та PHP. Дані мови програмування дозволили ефективно вирішити поставлену мету та задачі кваліфікаційної роботи.

Для шифрування даних у розробленому програмному забезпеченні було використано алгоритм асиметричного шифрування RSA, що дозволило захистити як дані, так і ключі шифрування. Адже RSA дозволяє здійснювати захищений обмін ключами шифрування, розділяючи їх на відкриті та закриті ключі для кожного користувача.

Запропоноване програмне забезпечення складається з клієнтського та серверного додатків. У п’ятому розділі пояснювальної записки надаються усі необхідні рекомендації з встановлення серверного програмного забезпечення та інструкція для використання клієнтського програмного забезпечення.

Для захисту розробленого програмного забезпечення було обрано вільну ліцензію Creative Commons.

Програмне забезпечення системи кібербезпеки для захищеного обміну даними є важливим елементом в будь-якому сучасному діловому середовищі, а також для особистого використання з метою захищеного обміну приватними даними у мережі Інтернет. Враховуючи все більшу кількість і складність кіберзагроз, використання такого роду програмного забезпечення стає не просто рекомендацією, а необхідністю. Тож, розроблене у цій кваліфікаційній роботі програмне забезпечення має важливе призначення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Комп’ютерні мережі [навчальний посібник] / А.Г. Микитишин, М.М. Митник, П.Д. Стухляк, В.В. Пасічник – Львів, «Магнолія 2006», 2013. – 256 с.
2. Комп’ютерні мережі [Текст]: 2-ге оновл. і доп. вид. / Є. Буров; ред. В. Пасічник. – Л.: БаК, 2003. – 584 с.
3. Програмне забезпечення компютерних мереж С.Забара,М.Дехтярук-Київ, «Університет Україна», 2012 р.-353 с.
4. Організація комп’ютерних мереж [Електронний ресурс]: підручник: для студ. спеціальності 121 «Інженерія програмного забезпечення» та 122 «Комп’ютерні науки»/ КПІ ім. Ігоря Сікорського; Ю.А. Тарнавський, І.М. Кузьменко. – Київ : КПІ ім. Ігоря Сікорського, 2018. – 259с.
5. Оліфер В.Г. Комп'ютерні мережі. Принципи, технології, протоколи: Підручник для вузів. 5-е изд. / В.Г. Оліфер, Н.А. Оліфер - СПб: Пітер, 2016. - 992 с.
6. Stallings W. Data and Computer Communications 10th - Pearson, 2013. - 912 p.
7. Таненбаум. Е. Комп'ютерні мережі. - Пітер, 2003. - 992 с.
8. Пятибратов А.П. Обчислювальні машини, мережі та телекомунікаційні системи. / П'ятибратов А.П., Гудино Л.П., Кириченко О.А. - М.: ЄАОІ, 2009. - 292 с.
9. Кулаков Ю.А., Луцький Г.М. Комп'ютерні мережі: Навчальний посібник, К.: Юніор, 1998. - 350 с.
10. Кулаков Ю.А., Луцький Г.М. Локальні мережі: Навчальний посібник, К.: Юніор, 1998. - 336 с.
11. Нанс Б. Комп'ютерні мережі: Пер. з англ. - М.: БІНОМ, 1996. - 400 с.
12. Волл Д. Використання World Wide Web. 2-е видання: Пер. з англ. - К: Діалектика, 1997. - 432с.
13. Хонікатт Д. Використання Internet. 2-е видання: Пер. з англ. - К: Діалектика, 1997. - 304с.
14. Шатт С. Світ комп'ютерних мереж: Пер. з англ. - К.: BHV, 1996. - 288с.

15. Larry L. Peterson, Bruce S. Davie. Computer Networks: A Systems Approach / The Morgan Kaufman series in Networking - 1999. - 776 p. 12

16. David G. Messerschmitt. Networked Applications: A Guide to the New Computing Infrastructure - The Morgan Kaufman series in Networking, 1999 - 396p. Іванов, П., Петров, В. (2020). Методи програмного моделювання топологій мереж у центрах обробки даних. Журнал "Інформаційні технології в управлінні", 12(3), 45-56.Сидоренко, О. (2019). Аналіз програмних продуктів для автоматизованого проектування мереж топологій у центрах обробки даних. Конференція "Інноваційні технології в інформаційному секторі".

17. Гаврилюк, М., Ковальчук, О. (2018). Роль програмного забезпечення у побудові оптимальних топологій мереж для центрів обробки даних. Міжнародний журнал "Інтегровані технології".

18. Ковальчук, А., Ігнатенко, С. (2017). Сучасні програмні засоби для аналізу та вибору топологій мереж у центрах обробки даних. Конгрес "Інформаційні технології та комп'ютерні системи".

19. Петренко, Н., Іванчук, Л. (2016). Використання програмних систем у проектуванні та оптимізації мереж для центрів обробки даних. Журнал "Комп'ютерні науки та інформаційні технології".

20. Шевченко, К.. (2015). Програмне забезпечення для автоматизованого створення та аналізу топологій мереж у центрах обробки даних. Конференція "Інноваційні рішення в ІТ".

21. Іванова, О., Павлов, Д. (2014). Огляд програмних засобів для моделювання, візуалізації та аналізу топологій мереж у центрах обробки даних. Міжнародний симпозіум "ІТ-інновації".

22. Ковальчук, С., Петренко, О. (2013). Сучасні програмні рішення для планування та оптимізації топологій мереж у центрах обробки даних. Журнал "Інформаційні технології в бізнесі".

23. Сидорова, Н., Коваленко, П. (2012). Розвиток програмного забезпечення для автоматизованого аналізу та моделювання мереж топологій у центрах обробки даних. Конференція "ІТ-інфраструктура".

24. Петрова, І., Сидоренко, О. (2011). Порівняльний аналіз програмних засобів для розробки топологій мереж у центрах обробки даних. Журнал "Інноваційні технології".

25. Ковальчук, О., Іваненко, М. (2010). Переваги використання програмного забезпечення у побудові мереж топологій для центрів обробки даних. Конгрес "ІТ-інфраструктура".

26. Шевченко, А., Ковальчук, О. (2009). Перспективи розвитку програмних продуктів для планування та аналізу топологій мереж у центрах обробки даних. Конференція "ІТ-інновації".

27. Іваненко, Т., Петров, В. (2008). Інноваційні програмні рішення для побудови та оптимізації мереж топологій у центрах обробки даних. Міжнародний форум "ІТ-технології".

28. Петренко, О., Сидоров, В. (2007). Програмне забезпечення для автоматизованого аналізу та моделювання топологій мереж у центрах обробки даних. Журнал "Інформаційні системи".

29. Сидоренко, І., Ковальчук, О. (2006). Сучасні програмні продукти для розробки та оцінки топології.

30. Методичні вказівки до виконання й захисту випускної випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти за спеціальністю 123 «Комп'ютерна інженерія». / проф. Смірнов О.А., проф. Мелешко Є.В., доц. Гермак В.С., доц. Буравченко К.О., доц. Якименко Н.М., доц. Смірнов С.А., доц.. Доренський О.П., доц. Смірнова Т.В. – Кропивницький: ЦНТУ, 2022. - 68 с.

31. Medium – [Електронний ресурс]. Режим доступу : https://medium.com/nuances-of-programming/плюсы-и-минусыпрограммирования-

на-java-2861f4c2a0d5 (дата запиту – 16.05.2020) – Плюси та мінуси програмування на Java.

32. ProgLang – [Електронний ресурс]. Режим доступу : http://proglang.su/java/introduction-to-programming#istoriya-sozdaniyayazyka-java (дата запиту – 16.05.2020) – Огляд мови програмування Java

33. Javarush – [Електронний ресурс]. Режим доступу : https://javarush.ru/groups/posts/510-preimujshestva-ispoljhzovanija-spring (дата запиту – 16.05.2020) – Переваги використання Spring

34. ItProger – [Електронний ресурс]. Режим доступу : https://tproger.ru/articles/maven-short-intro/ (дата запиту – 16.05.2020) – Коротке знайомство із Maven

35. Хабр – [Електронний ресурс]. Режим доступу : https://habr.com/ru/post/435144/ (дата запиту – 16.05.2020) – Введення в Spring Boot. Створення простого REST API на Java.

36. Proselyte – [Електронний ресурс]. Режим доступу : https://proselyte.net/tutorials/junit/introduction/ (дата запиту – 16.05.2020) – Керівництво по JUnit. Введення. Зм. Арк. № докум. Підпис Дата Арк. 91 ІАЛЦ.467200.003 ПЗ

37. Хабр – [Електронний ресурс]. Режим доступу : https://habr.com/ru/post/434798/ (дата запиту – 16.05.2020) – Swagger – розумна документація.

38. Хабр – [Електронний ресурс]. Режим доступу : https://habr.com/ru/post/438870/ (дата запиту – 16.05.2020) – Lombok повертає велич Java.

39. Codeflow – [Електронний ресурс]. Режим доступу : https://www.codeflow.site/ru/article/mapstruct (дата запиту – 16.05.2020) – Коротке керівництво по MapStruct.

40. Bauman National Library – [Електронний ресурс]. Режим доступу : https://ru.bmstu.wiki/index.php?title=Spring_Framework&mobileaction=toggle_view_desktop (дата запиту – 16.05.2020) – Spring Framework.

41. Tutorials Point – [Електронний ресурс]. Режим доступу : https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm(да та запиту – 16.05.2020) – Spring Boot Introduction.

42. Java Study – [Електронний ресурс]. Режим доступу : http://javastudy.ru/junit/junit-hello-world/ (дата запиту – 16.05.2020) – JUnit – введення в юніт-тести

43. Оліфер В. Г., Оліфер Н. О. Комп'ютерні мережі. Принципи, технології, протоколи. - 4-е изд. СПб.:Питер, 2010. - С. 438. - 4500 прим. - ISBN 978-5-49807-389-7.

44. Комп'ютерні мережі. 4-е изд./ І. Таненбаум. - СПб.: Пітер, 2003. - 992 с.

45. Стрихалюк Б. М. Теорія побудови та протоколи інфокомунікаційних мереж: Конспект лекцій. - Львів: Львівська політехніка, 2017. - 121 с.

46. Р. Стівенс Протоколи TCP/IP. Практичний посібник. - Спб.: БХВ, 2003

47. Дуглас Камер Мережі TCP/IP, том 1. Принципи, протоколи та структура = Internetworking with TCP/IP, Vol. 1: Principles, Protocols and Architecture. - М.: "Вільямс", 2003. - С. 880. - ISBN 0-13-018380-6.

48. Сліпченко В. Г. Локальні комп'ютерні мережі. Проектування, використання та програмування: навч. посіб. / В. Г. Сліпченко, В. І. Гайдаржи, В. А. Лабжинський. – Київ: ІВЦ «Політехніка», 2002. – 184 с.

49. Net Cracker 4.1. User Manual. Нормативні матеріали: 259 [Електронний ресурс]. – Режим доступу : http://soft-landia.ru/netcracker.html. – Назва з екрана.

50. Cisco Packet Tracer. Лабораторная работа 5: [Електронний ресурс].– Режим доступу: https:/studfiles.net/donntu/145/folder:11411/#5682479

Додаток А

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи проектування та побудови топології мереж у центрах обробки даних.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу № 133-02 від 01.04.2024 року, видане на кафедрі кібербезпеки та програмного забезпечення.

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи проектування та побудови топології мереж у центрах обробки даних.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– систему проектування та побудови топології мереж у центрах обробки даних;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel Core i7/8 ГБ /1 Tb/ GeForce GT 1030 2GB або сумісні з ним.

5.8.2 Мова програмування

Програму розроблено на мові програмування Java.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД.

7 Перелік документів, що розробляються

– Структурна схема системи. 1 аркуш
– Функціональна схема системи. 1 аркуш
– Діаграма процесів. 1 аркуш
– Блок-схема роботи програми. 2 аркуша
– Пояснювальна записка. 55 аркушів

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 19.05.2024 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 07.06.2024 р.

Додаток Б

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи

за першим (бакалаврським) рівнем вищої освіти

_____ А.С. Коваленко

Програмне забезпечення системи проектування та побудови топології мереж у центрах оборки даних

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 44

Літера: РП

Кропивницький – 2024 року

//Constants.java – Клас констатів

```
package com.example.demo1;

import com.example.demo1.clasess.ImageInfo;
import com.example.demo1.protocols.UserCredentials;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.layout.Pane;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Line;

import java.util.ArrayList;
import java.util.List;

public class Constants {
    public static final String PATH_SERVER = "/assets/server.png";
    public static final String PATH_PC = "/assets/pc.png";
    public static final String PATH_COMUTATOR = "/assets/comutator.png";
    public static final String PATH_PRINTER = "/assets/printer.png";
    public static final String PATH_ROUTER = "/assets/router.png";
    public static final String Path_STYLE = "/assets/infoStage.css";

    public static final ArrayList<Circle> circleList = new ArrayList<>();
    public static final List<ImageInfo> imageList = new ArrayList<>();
    @SuppressWarnings("exports")
    public static final List<Line> lineList = new ArrayList<>();

    public static String NAME_FILE_NEW = "diplom.ser";
    @SuppressWarnings("exports")
    public static Pane imageContainer;



    public static final String PATH_GOOGLE = "/assets/web.png";
    public static final String PATH_TERMINAL = "/assets/terminal.png";


    public static ArrayList<String> ipServer = new ArrayList<>();
    public static ArrayList<String> ipCommutator = new ArrayList<>();
    public static ArrayList<String> ipPc = new ArrayList<>();
    public static ArrayList<String> ipPrinter = new ArrayList<>();
    public static ArrayList<String> ipRouter = new ArrayList<>();

    public static ObservableList<UserCredentials> listUserFtp =
FXCollections.observableArrayList();



}
```

//TextInputDialogFtp.java – Клас для протоколу FTP

```
package com.example.demo1.clasess;

import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.layout.VBox;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

import java.io.File;
```

```
import java.io.FileWriter;
import java.io.IOException;

public class TextInputDialogFtp extends Stage {
    private static TextArea textArea;

    public TextInputDialogFtp() {
        initializeUI();
    }

    public void initializeUI() {
        VBox root = new VBox();
        root.setPadding(new Insets(10));

        textArea = new TextArea();
        textArea.setPrefSize(300, 200);

        Button saveButton = new Button("Save");

        saveButton.setOnAction(event -> {
            String text = textArea.getText();

            FileChooser fileChooser = new FileChooser();
            fileChooser.setInitialDirectory(new
File("src/main/resources/assets/files"));

            fileChooser.getExtensionFilters().addAll(
                    new FileChooser.ExtensionFilter("Text Files", "*.txt"),
                    new FileChooser.ExtensionFilter("All Files", "*.*")
            );

            File selectedFile = fileChooser.showSaveDialog(null);

            if (selectedFile != null) {
                try {
                    FileWriter writer = new FileWriter(selectedFile);
                    writer.write(text);
                    writer.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            close();
        });


        root.getChildren().addAll(textArea, saveButton);

        Scene scene = new Scene(root);
        setScene(scene);
    }
}
```

//ConnectObject.java– Клас для з’єднання елементів

```
package com.example.demo1.object;

import com.example.demo1.MenuCreator;
import com.example.demo1.clasess.ImageInfo;
import javafx.scene.control.ChoiceDialog;
import javafx.scene.control.ListView;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.shape.Line;
```

```
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;


import static com.example.demo1.Constants.*;
import static com.example.demo1.Redraw.getImageInfoByType;
import static com.example.demo1.Redraw.redrawImageContainer;

public class ConnectObject {

    public static void disconnectItem(ImageInfo imageInfo) {
        ImageInfo connectedImageInfo = imageInfo.getConnectedItem().get(0);
        List<ImageInfo> listItem = imageInfo.getConnectedItem();

        String idLine = imageInfo.getName() + " " +
connectedImageInfo.getName();
        String idLine2 = connectedImageInfo.getName() + " " +
imageInfo.getName();


        if (listItem.size() < 2) {
            circleList.removeIf(circle -> circle.getId().equals(idLine) ||
circle.getId().equals(idLine2));

            imageInfo.removeConnectItem(connectedImageInfo);
            connectedImageInfo.removeConnectItem(imageInfo);


            if (!connectedImageInfo.getConnectedItem().isEmpty()) {
                String newIdLine = "";

                if(!connectedImageInfo.getConnectedLines().isEmpty()){
                        newIdLine =
connectedImageInfo.getConnectedLines().get(0).getId();
                }
                connectedImageInfo.setIdLine(newIdLine);

                connectedImageInfo.setMovable(false);
                connectedImageInfo.setDisabled(true);
            }
            if (imageInfo.getIdLine().equals("") &&
!imageInfo.getConnectedLines().isEmpty()) {
                imageInfo.clearConnectedLines();
            }

            if (connectedImageInfo.getConnectedLines().isEmpty()){
                if (!connectedImageInfo.getConnectedItem().isEmpty()){
                    int index = -1; // Ініціалізуємо змінну-лічильник

                    for (int i = 0; i < imageList.size(); i++) {
                        ImageInfo imgIn = imageList.get(i);

                        if
(imgIn.getName().equals(connectedImageInfo.getName())) {
                            index = i; // Встановлюємо індекс, якщо знаходимо
відповідний елемент
                            break; // Перериваємо цикл, оскільки елемент
знайдено
                        }
                    }

                    ImageInfo imageInfo1 = new ImageInfo();
                    imageInfo1.setType(connectedImageInfo.getType());
                    imageInfo1.setName(connectedImageInfo.getName());
                    imageInfo1.setX(connectedImageInfo.getX());
                    imageInfo1.setY(connectedImageInfo.getY());
                    imageInfo1.setMovable(true);
```

```
                    imageInfo1.setDisabled(false);


                    switch (imageInfo1.getType()) {
                        case "server" ->
imageInfo1.setImageView(createImageView( PATH_SERVER));
                        case "pc" -> imageInfo1.setImageView(createImageView(
PATH_PC));
                        case "comutator" ->
imageInfo1.setImageView(createImageView( PATH_COMUTATOR));
                        case "printer" ->
imageInfo1.setImageView(createImageView( PATH_PRINTER));
                        case "router" ->
imageInfo1.setImageView(createImageView( PATH_ROUTER));
                    }


                    if (index != -1) {
                        imageList.set(index, imageInfo1);
                    }else {
                        try {
                            throw new Exception("Index == -1");
                        } catch (Exception e) {
                            throw new RuntimeException(e);
                        }
                    }
                }

            }
            else {
                removeInconsistentConnectedItems(imageInfo);
                removeInconsistentConnectedItems(connectedImageInfo);
            }
        } else {
            disconnectedDialog(imageInfo);
        }

        redrawImageContainer();
    }

    public static ImageView createImageView(String path){
        InputStream imageStream = MenuCreator.class.getResourceAsStream(path);
        assert imageStream != null;
        ImageView imageView = new ImageView(new Image(imageStream));

        // Встановлюємо бажані розміри
        imageView.setFitWidth(50);
        imageView.setFitHeight(50);

        // Обмеження зміни розміру (незмінний)
        imageView.setPreserveRatio(true);
        imageView.setFitWidth(100);

        return imageView;
    }

    private static void disconnectedDialog(ImageInfo imageInfo) {
        // Створюємо списки для вибору елементів з imageContainer
        ListView<String> listView2 = disconnectedImageNameListView(imageInfo);

        // Створюємо діалог вибору
        ChoiceDialog<String> dialog = new ChoiceDialog<>(null,
listView2.getItems());
        dialog.setTitle("Від'єднані елементи");
        dialog.setHeaderText("Виберіть елементи для від'єднання");
        dialog.setContentText("Виберіть елементи для від'єднання:");
        dialog.setHeight(250);

        // Очікуємо вибір користувачем і обробляємо результат
```

```
        Optional<String> result = dialog.showAndWait();

        result.ifPresent(selectedItem -> {
            ImageInfo selectedImage = getImageInfoByType(selectedItem); //
елемент, який вибрали зі списку

            String idLine = imageInfo.getName() + " " + selectedImage.getName();
            String idLine2 = selectedImage.getName() + " " +
imageInfo.getName();

            circleList.removeIf(circle -> circle.getId().equals(idLine) ||
circle.getId().equals(idLine2));

            selectedImage.removeConnectItem(imageInfo);

            lineList.removeIf(line -> line.getId().equals(idLine) ||
line.getId().equals(idLine2));

            imageInfo.removeConnectItem(selectedImage);


            if (!selectedImage.getConnectedLines().isEmpty()) {
                selectedImage.setMovable(false);
                selectedImage.setDisabled(true);
            }

            imageInfo.setMovable(false);
            imageInfo.setDisabled(true);

            if (!imageInfo.getConnectedLines().isEmpty()) {
                String id = imageInfo.getConnectedLines().get(0).getId();
                imageInfo.setIdLine(id);
            }

            if (selectedImage.getIdLine().equals("") &&
selectedImage.getConnectedLines().isEmpty()) {
                selectedImage.clearConnectedLines();
            } else if (!selectedImage.getConnectedLines().isEmpty() &&
selectedImage.getIdLine().equals("")) {
                String id = selectedImage.getConnectedLines().get(0).getId();
                selectedImage.setIdLine(id);
            }


            if (selectedImage.getConnectedLines().size() !=
selectedImage.getConnectedItem().size()){
                removeInconsistentConnectedItems(selectedImage);
                removeInconsistentConnectedItems(imageInfo);
            }

        });
    }



    public static void removeInconsistentConnectedItems(ImageInfo selectedImage)
{
        ArrayList<ImageInfo> arrayList = selectedImage.getConnectedItem();
        ArrayList<Line> arrayListLine = selectedImage.getConnectedLines();

        ArrayList<String> nameItems = new ArrayList<>();
        ArrayList<String> nameLine = new ArrayList<>();

        for (ImageInfo imageInfo1 : arrayList) {
            nameItems.add(imageInfo1.getName());
        }

        for (Line line : arrayListLine) {
            String[] str = line.getId().split("\\s+");
```

```
            nameLine.add(str[0] + " " + str[1]);
            nameLine.add(str[2] + " " + str[3]);
        }

        ArrayList<String> originalNameItems = new ArrayList<>(nameItems);

        nameItems.retainAll(nameLine);

        // Знаходимо елемент, який видаляється
        originalNameItems.removeAll(nameItems);

        if (!originalNameItems.isEmpty()) {
            String removedElement = originalNameItems.get(0);

            selectedImage.returnItemConnectedList(removedElement);
        }
    }

    public static void connectItemDialog(ImageInfo imageInfo) {
        // Створюємо списки для вибору елементів з imageContainer
        ListView<String> listView2 = createDeviceListView(imageInfo);

        // Створюємо діалог вибору
        ChoiceDialog<String> dialog = new ChoiceDialog<>(null,
listView2.getItems());
        dialog.setTitle("Підключення елементів");
        dialog.setHeaderText("Виберіть елементи для підключення");
        dialog.setContentText("Виберіть елементи для підключення:");
        dialog.setHeight(250);

        // Очікуємо вибір користувачем і обробляємо результат
        Optional<String> result = dialog.showAndWait();

        result.ifPresent(selectedItem -> {
            ImageInfo selectedImage2 = getImageInfoByType(selectedItem);

            imageInfo.setConnectedItem(selectedImage2);
            selectedImage2.setConnectedItem(imageInfo);

            connectItemLine(imageInfo, selectedImage2);

            // Очистити і перерисувати контейнер
            redrawImageContainer();
        });
    }


    public static void connectItemLine(ImageInfo imageInfo, ImageInfo
imageInfo2) {
        double startX = imageInfo.getX() + 25;
        double startY = imageInfo.getY() + 25;

        double endX = imageInfo2.getX() + 25;
        double endY = imageInfo2.getY() + 25;

        Line line = new Line(startX, startY, endX, endY);

        line.setId(imageInfo.getName() + " " + imageInfo2.getName());

        imageInfo.setIdLine(line.getId());
        imageInfo2.setIdLine(line.getId());

        imageInfo.setLineConnectedItem(line);
        imageInfo2.setLineConnectedItem(line);

        line.setStyle("-fx-opacity: 0.5;");

        lineList.add(line); // Додаємо лінію до списку
```

```
        imageContainer.getChildren().add(line); // Додаємо лінію до контейнера


        // Зробити елементи неактивними
        imageInfo.setMovable(false);
        imageInfo2.setMovable(false);

        imageInfo.setDisabled(true);
        imageInfo2.setDisabled(true);

    }

    private static ListView<String> createDeviceListView(ImageInfo imageInfo) {
        ListView<String> listView = new ListView<>();
        Optional<Class<?>> deviceClassOptional =
imageInfo.getDevice().map(Object::getClass);
        String className =
deviceClassOptional.map(Class::getSimpleName).orElse("");

        imageList.stream()
                .filter(imageInfo1 ->
!imageInfo1.getName().equals(imageInfo.getName()))
                .filter(imageInfo1 -> {
                    String classImage =
imageInfo1.getDevice().map(Object::getClass).map(Class::getSimpleName).orElse("
");
                    if (className.equalsIgnoreCase("PC")) {
                        return !classImage.equalsIgnoreCase("PC");
                    } else if (className.equalsIgnoreCase("Printer")) {
                        return !classImage.equalsIgnoreCase("Router") &&
!classImage.equalsIgnoreCase("Printer");
                    }else if (className.equalsIgnoreCase("Router")){
                        return !classImage.equalsIgnoreCase("Router") &&
!classImage.equalsIgnoreCase("Printer");
                    }else {
                        return true;  // Для інших випадків, нічого не
виключаємо
                    }
                })
                .forEach(imageInfo1 ->
listView.getItems().add(imageInfo1.getName()));

        return listView;
    }

    private static ListView<String> disconnectedImageNameListView(ImageInfo
imageInfo){
        ListView<String> listView = new ListView<>();

        for (ImageInfo img: imageInfo.getConnectedItem()){
            listView.getItems().add(img.getName());
        }

        return listView;
    }

}
```

//MovedObject.java – класс для переміщення елементів

```
package com.example.demo1.object;

import com.example.demo1.clasess.ImageInfo;
import javafx.scene.layout.VBox;

import java.util.concurrent.atomic.AtomicBoolean;
```

```
public class MovedObject {
    /**
     * пересунуто
     */
    public static void setDragEventHandlers(VBox imageBox, ImageInfo imageInfo)
{
        AtomicBoolean moved = new AtomicBoolean(true);
        if (!imageInfo.isMovable()) moved.set(false);

        if (moved.get()) {
            imageBox.setOnMousePressed(mouseEvent -> {
                // Збереження початкових координат
                imageBox.setUserData(new double[]{mouseEvent.getSceneX(),
mouseEvent.getSceneY()});
            });

            imageBox.setOnMouseDragged(mouseEvent -> {
                // Обчислення зміни координат та переміщення зображення
                double[] userData = (double[]) imageBox.getUserData();
                double deltaX = mouseEvent.getSceneX() - userData[0];
                double deltaY = mouseEvent.getSceneY() - userData[1];

                imageBox.setTranslateX(imageBox.getTranslateX() + deltaX);
                imageBox.setTranslateY(imageBox.getTranslateY() + deltaY);
                imageBox.setUserData(new double[]{mouseEvent.getSceneX(),
mouseEvent.getSceneY()});

                imageInfo.setX(imageBox.getTranslateX());
                imageInfo.setY(imageBox.getTranslateY());

            });
        } else {
            System.out.println("Movable == false");
        }
    }

}
```

//ObjectCreator.java – класс для створення елеметів

```
   package com.example.demo1.object;

import com.example.demo1.MenuCreator;
import com.example.demo1.clasess.*;
import javafx.scene.control.ContextMenu;
import javafx.scene.control.Label;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Line;

import java.io.InputStream;

import static com.example.demo1.Constants.*;
import static com.example.demo1.MenuCreator.createContextMenu;
import static com.example.demo1.object.MovedObject.setDragEventHandlers;

public class ObjectCreator {

    public static void createObject(String path) {
        InputStream imageStream = MenuCreator.class.getResourceAsStream(path);

        String type = parseImageType(path);

        if (imageStream != null) {
```

```
            Image image = new Image(imageStream);
            ImageView imageView = new ImageView(image);

            // Встановлюємо бажані розміри
            imageView.setFitWidth(50);
            imageView.setFitHeight(50);

            // Обмеження зміни розміру (незмінний)
            imageView.setPreserveRatio(true);
            imageView.setFitWidth(100);

            // Створення об'єкта ImageInfo і додавання його до списку
            ImageInfo imageInfo = new ImageInfo();
            imageInfo.setImageView(imageView);
            imageInfo.setType(type);
            imageInfo.setName(type + " " + imageList.size());

            switch (type) {
                case "comutator" -> imageInfo.setCommutator(new
Commutator(imageInfo.getName()));
                case "pc" -> imageInfo.setPC(new PC(imageInfo.getName()));
                case "printer" -> imageInfo.setPrinter(new
Printer(imageInfo.getName()));
                case "router" -> imageInfo.setRouter(new
Router(imageInfo.getName()));
                case "server" -> imageInfo.setServer(new
Server(imageInfo.getName()));
                default -> {}
            }

            imageList.add(imageInfo);

            // Встановлення обробника події для контекстного меню
            ContextMenu contextMenu = createContextMenu(imageInfo);
            imageView.setOnContextMenuRequested(event ->
contextMenu.show(imageView, event.getScreenX(), event.getScreenY()));

            // Створюємо VBox та додаємо до нього ImageView та Label
            VBox imageBox = new VBox();
            imageBox.setMaxWidth(50);
            imageBox.setMaxHeight(50);

            imageBox.getChildren().addAll(imageView, new
Label(imageInfo.getName()));

            if (imageInfo.isMovable()) {
                setDragEventHandlers(imageBox, imageInfo);
            }

            // Додаємо зображення на VBox
            imageContainer.getChildren().add(imageBox);
        } else {
            System.err.println("Не вдалося завантажити зображення: " + path);
        }
    }

    public static String parseImageType(String imageName) {
        int lastSlashIndex = imageName.lastIndexOf('/');// Знаходимо останній
символ '/'
        int lastDotIndex = imageName.lastIndexOf('.'); // Якщо є '/', знаходимо
останній символ '.'
        int startIndex = (lastSlashIndex >= 0) ? lastSlashIndex + 1 : 0;//
Визначаємо початок підстроки
        int endIndex = (lastDotIndex >= 0) ? lastDotIndex :
imageName.length();// Визначаємо кінець підстроки (якщо є '.')

        return imageName.substring(startIndex, endIndex);// Витягаємо підстроку
    }
```

```
    public static void createCircleLine(Line line){
        // Створюємо кола для країв лінії
        Circle circleStart = new Circle(3);
        Circle circleEnd = new Circle(3);

        // Встановлюємо координати колів
        circleStart.setCenterX(line.getStartX());
        circleStart.setCenterY(line.getStartY());
        circleEnd.setCenterX(line.getEndX());
        circleEnd.setCenterY(line.getEndY());

        // Встановлюємо коліру заливки колів по значенням RGB
        circleStart.setFill(Color.rgb(118, 237, 71));
        circleEnd.setFill(Color.rgb(118, 237, 71));

        circleStart.setId(line.getId());
        circleEnd.setId(line.getId());

        circleList.add(circleStart);
        circleList.add(circleEnd);

        imageContainer.getChildren().addAll(circleStart, circleEnd);
    }
}
```

//BrowserSimulator.java – клас симулятор браузера

```
package com.example.demo1.protocols;

import com.example.demo1.clasess.ImageInfo;
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;
import javafx.scene.web.WebView;
import javafx.stage.Stage;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class BrowserSimulator extends Application {
    private static final int PORT = 8080;

    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) { }

    public static void runServer(ImageInfo imageInfo) {
        WebView webView = new WebView();

        // Додаємо заголовок
        Label titleLabel = new Label("http://localhost:" + PORT);

        // Розташовуємо елементи на сцені
        VBox root = new VBox(titleLabel, webView);
        root.setAlignment(Pos.CENTER); // Центруємо по вертикалі
        titleLabel.setAlignment(Pos.CENTER); // Центруємо заголовок по
горизонталі

        Scene scene = new Scene(root, 400, 300);

        Stage primaryStage = new Stage();
```

```
        primaryStage.setTitle("Браузера");
        primaryStage.setScene(scene);
        primaryStage.show();

        // Завантажуємо сторінку
        webView.getEngine().load("http://localhost:" + PORT);

        new Thread(() -> {
            try {
                ServerSocket serverSocket = new ServerSocket(PORT);
                System.out.println("Сервер запущений і чекає на підключення
клієнта...");

                while (true) {
                    Socket clientSocket = serverSocket.accept();
                    int clientPort = clientSocket.getPort();

                    BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                    PrintWriter out = new
PrintWriter(clientSocket.getOutputStream(), true);

                    String request = in.readLine();

                    String ipClient = "Підключений клієнт з IP: " +
imageInfo.getIPAddress() + ", порт: " + clientPort;
                    String requestClient = "Отримано запит від клієнта: " +
request;

                    if (imageInfo.getProtocolsConnect().equals("HTTP"))
out.println("HTTP/1.1 200 OK");
                    else out.println("HTTPS/1.1 200 OK");

                    out.println("Content-Type: text/html; charset=UTF-8"); //
Встановлення кодування
                    out.println("\r\n");

                    out.println("<html><head><title>Відповідь від
сервера</title></head><body>");
                    out.println("<div>" + ipClient + "</div>");
                    out.println("<div>" + requestClient + "</div>");
                    out.println("</body></html>");

                    System.out.println("З'єднання з клієнтом закрито.");

                    in.close();
                    out.close();
                    clientSocket.close();
                    serverSocket.close();
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }).start();
    }
}
```

//FileItem.java – класс для роботи з фаловим менеджером

```
package com.example.demo1.protocols;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
```

```
public class FileItem {
    private final StringProperty fileName;
    private final StringProperty fileSize;

    public FileItem(String fileName, String fileSize) {
        this.fileName = new SimpleStringProperty(fileName);
        this.fileSize = new SimpleStringProperty(fileSize);
    }

    public static String getFormattedFileSize(long size) {
        if (size < 1024) {
            return size + " bytes";
        } else if (size < 1024 * 1024) {
            return String.format("%.2f KB", size / 1024.0);
        } else if (size < 1024 * 1024 * 1024) {
            return String.format("%.2f MB", size / (1024.0 * 1024));
        } else {
            return String.format("%.2f GB", size / (1024.0 * 1024 * 1024));
        }
    }

    public String getFileName() {
        return fileName.get();
    }

    public void setFileName(String fileName) {
        this.fileName.set(fileName);
    }

    public StringProperty fileNameProperty() {
        return fileName;
    }

    public String getFileSize() {
        return fileSize.get();
    }

    public void setFileSize(String fileSize) {
        this.fileSize.set(fileSize);
    }

    public StringProperty fileSizeProperty() {
        return fileSize;
    }
}
```


//FTPClient.java – класс для кліента FTP пртоколу

```
package com.example.demo1.protocols;

import java.io.*;

public class FTPClient {
    public static void main(String[] args) {
        org.apache.commons.net.ftp.FTPClient ftpClient = new
org.apache.commons.net.ftp.FTPClient();
        String server = "ftp.example.com";
        int port = 21;
        String user = "ім'я_користувача";
        String pass = "пароль";

        try {
            ftpClient.connect(server, port);
            ftpClient.login(user, pass);

            // Перейти в робочий каталог на сервері (якщо потрібно)
            ftpClient.changeWorkingDirectory("/віддалений/каталог");

            // Завантажити файл на сервер
```

```
            File file = new File("локальний_файл.txt");
            InputStream inputStream = new FileInputStream(file);
            boolean uploaded = ftpClient.storeFile("віддалений_ім'я_файлу.txt",
inputStream);
            inputStream.close();
            if (uploaded) {
                System.out.println("Файл успішно завантажено на сервер");
            }

            // Закрити з'єднання з сервером
            ftpClient.logout();
            ftpClient.disconnect();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

//FTPServer.java – класс для серверу FTP пртоколу

```
package com.example.demo1.protocols;

import org.apache.commons.net.ftp.FTPClient;
import org.apache.commons.net.ftp.FTP;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

public class FTPServer {
    public static void main(String[] args) {
        FTPClient ftpClient = new FTPClient();
        int port = 21;
        String user = "ім'я_користувача";
        String pass = "пароль";

        try {
            ftpClient.connect("localhost", port);
            ftpClient.login(user, pass);

            // Перейти в робочий каталог на сервері (якщо потрібно)
            ftpClient.changeWorkingDirectory("/віддалений/каталог");

            // Відправити файл на сервер
            File file = new File("локальний_файл.txt");
            InputStream inputStream = new FileInputStream(file);
            ftpClient.setFileType(FTP.BINARY_FILE_TYPE);
            ftpClient.storeFile("віддалений_ім'я_файлу.txt", inputStream);
            inputStream.close();

            // Закрити з'єднання з сервером
            ftpClient.logout();
            ftpClient.disconnect();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

//ftpSetting.java – класс для налаштування FTP протоколу

```
 package com.example.demo1.settings.services;

import com.example.demo1.clasess.TextInputDialogFtp;
```

```
import com.example.demo1.protocols.FileItem;
import com.example.demo1.protocols.UserCredentials;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

import java.io.File;
import java.util.Arrays;

import static com.example.demo1.Constants.listUserFtp;
import static com.example.demo1.protocols.FileItem.getFormattedFileSize;

public class FtpSetting {

    /**
     * Метод для додавання нового користувача FTP.
     * Створює нове вікно для додавання користувача та пароля,
     * надає поля для введення імені користувача і пароля,
     * а також кнопки для додавання та відміни операції.
     * Після додавання користувача закриває вікно.
     */
    private static void clickAddFtp() {
        // Створення нового вікна для додавання користувача та пароля
        Stage addUserStage = new Stage();
        addUserStage.setTitle("Додати користувача");

        // Створення компонентів для введення імені користувача і пароля
        Label usernameLabel = new Label("Ім'я користувача:");
        TextField usernameField = new TextField();
        Label passwordLabel = new Label("Пароль:");
        TextField passwordField = new TextField();

        // Створення кнопок для додавання та відміни операції
        Button addUserButton = new Button("Додати");

        addUserButton.setOnAction(e -> {
            // Додавання користувача
            String username = usernameField.getText();
            String password = passwordField.getText();

            // Виведення інформації про доданого користувача в консоль
(тимчасово)
            System.out.println("Додано користувача: " + username + ", пароль: "
+ password);

            // Додавання користувача в список
            listUserFtp.add(new UserCredentials(username, password));

            // Закриття вікна після додавання користувача
            addUserStage.close();
        });

        // Кнопка для відміни операції додавання користувача
        Button cancelButton = new Button("Скасувати");
        cancelButton.setOnAction(e -> addUserStage.close());

        // Розташування компонентів на сцені
        VBox layout = new VBox(10, usernameLabel, usernameField, passwordLabel,
passwordField, addUserButton, cancelButton);
        layout.setAlignment(Pos.CENTER);
        layout.setPadding(new Insets(10));

        // Створення сцени та встановлення на вікно
        Scene scene = new Scene(layout);
```

```
        addUserStage.setScene(scene);

        // Показати вікно додавання користувача
        addUserStage.show();
    }

    /**
     * Метод для налаштування компонентів FTP протоколу.
     * Створює розділ FTP на головній сторінці, додає контейнери
     * для налаштувань користувачів та файлів, і встановлює їх в кореневий
вузол.
     *
     * @param root Кореневий вузол, в який додаються компоненти FTP.
     */
    public static void clickFtpProtocols(BorderPane root) {
        // Назва розділу FTP
        Label ftpLabel = new Label("FTP");
        ftpLabel.setPadding(new Insets(10));
        ftpLabel.setStyle("-fx-font-size: 18px;");

        // Створення контейнера для налаштувань користувача
        VBox userSetupContainer = blockUserSetup();

        // Створення контейнера для налаштувань файлів
        VBox filesSetupContainer = blockFilesSetup();

        // Додавання компонентів в кореневий вузол
        VBox container = new VBox(10, ftpLabel, userSetupContainer,
filesSetupContainer);
        container.setPadding(new Insets(10));
        root.setCenter(container);
    }

    /**
     * Метод для налаштування компонентів, пов'язаних з файлами.
     * Створює блок з файлами, таблицю для відображення файлів і кнопки
     * для додавання і видалення файлів, а також встановлює обробники
     * подій для цих кнопок.
     *
     * @return Контейнер, що містить компоненти для роботи з файлами.
     */
    private static VBox blockFilesSetup() {
        // Блок з файлами
        Label filesLabel = new Label("Файли");
        filesLabel.setStyle("-fx-font-weight: bold;");
        filesLabel.setPadding(new Insets(10));

        // Таблиця для відображення файлів
        TableView<FileItem> filesTable = new TableView<>();

        TableColumn<FileItem, String> fileNameColumn = new TableColumn<>("Ім'я
файлу");
        TableColumn<FileItem, String> fileSizeColumn = new TableColumn<>("Розмір
файлу");

        fileNameColumn.setCellValueFactory(cellData ->
cellData.getValue().fileNameProperty());
        fileSizeColumn.setCellValueFactory(cellData ->
cellData.getValue().fileSizeProperty());

        // Додавання стовпців в таблицю
        filesTable.getColumns().add(fileNameColumn);
        filesTable.getColumns().add(fileSizeColumn);

        // Встановлення заповнювача для випадку відсутності файлів
        filesTable.setPlaceholder(new Label("Немає файлів"));

        filesTable.setPrefHeight(200);
        filesTable.setMinHeight(100);
```

```
        // Кнопки для додавання і видалення файлів
        Button addFileButton = new Button("Додати");
        Button removeFileButton = new Button("Видалити");

        // Встановлення фіксованої ширини для кнопок
        addFileButton.setPrefWidth(100);
        removeFileButton.setPrefWidth(100);
        addFileButton.setPadding(new Insets(5));
        removeFileButton.setPadding(new Insets(5));

        // Обробник натискання на кнопку додавання файлу
        addFileButton.setOnAction(event -> {
            TextInputDialogFtp dialog = new TextInputDialogFtp();
            dialog.showAndWait();

            // Після закриття діалогового вікна, перезавантажте вміст таблиці
            File folder = new File("src/main/resources/assets/files");
            File[] listOfFiles = folder.listFiles();

            // Перевіряємо, що список файлів не порожній
            if (listOfFiles != null) {
                // Шукаємо лише нові файли, які не були додані раніше
                for (File file : listOfFiles) {
                    if (file.isFile()) {
                        boolean alreadyExists = false;
                        for (FileItem item : filesTable.getItems()) {
                            if (item.getFileName().equals(file.getName())) {
                                alreadyExists = true;
                                break;
                            }
                        }
                        if (!alreadyExists) {
                            // Отримуємо розмір файлу у рядковому форматі з
урахуванням точності
                            String fileSize =
getFormattedFileSize(file.length());
                            // Створюємо об'єкт FileItem для поточного файлу
                            FileItem fileItem = new FileItem(file.getName(),
fileSize);
                            // Додаємо об'єкт FileItem в дані таблиці
                            filesTable.getItems().add(fileItem);
                        }
                    }
                }
                filesTable.refresh(); // Оновлюємо таблицю після додавання
файлів
            } else {
                System.out.println("Немає файлів у каталозі.");
            }
        });

        // Обробник натискання на кнопку видалення файлу
        removeFileButton.setOnAction(event -> {
            // Отримуємо вибраний елемент в таблиці
            FileItem selectedFileItem =
filesTable.getSelectionModel().getSelectedItem();

            if (selectedFileItem != null) {
                // Видаляємо вибраний файл з таблиці
                filesTable.getItems().remove(selectedFileItem);

                // Видаляємо вибраний файл з папки
                String fileName = selectedFileItem.getFileName();
                File fileToDelete = new File("src/main/resources/assets/files/"
+ fileName);

                if (fileToDelete.delete()) {
```

```
                    System.out.println("Файл " + fileName + " успішно
видалено.");
                } else {
                    System.out.println("Не вдалося видалити файл " + fileName);
                }
            } else {
                System.out.println("Файл не обрано.");
            }
        });

        // Контейнери для компонентів, пов'язаних з файлами
        VBox filesContainer = new VBox(10, filesLabel, filesTable);
        HBox buttonContainer = new HBox(10, addFileButton, removeFileButton);

        filesContainer.setPadding(new Insets(10));
        buttonContainer.setPadding(new Insets(10));
        buttonContainer.setAlignment(Pos.CENTER_RIGHT);

        // Заповнення таблиці даними про файли
        populateFilesTable(filesTable);

        return new VBox(10, filesContainer, buttonContainer);
    }

    /**
     * Метод для налаштування компонентів, пов'язаних з користувачами.
     * Створює блок для налаштування користувачів, таблицю для відображення
     * користувачів і кнопки для додавання і видалення користувачів.
     * Встановлює обробники подій для цих кнопок.
     *
     * @return Контейнер, що містить компоненти для роботи з користувачами.
     */
    private static VBox blockUserSetup() {
        // Блок User Setup
        Label userSetupLabel = new Label("Налаштування користувача");
        userSetupLabel.setStyle("-fx-font-weight: bold;");
        userSetupLabel.setPadding(new Insets(10));

        // Таблиця для відображення користувачів
        TableView<UserCredentials> credentialsTable = new TableView<>();
        TableColumn<UserCredentials, String> usernameColumn = new
TableColumn<>("Ім'я користувача");
        usernameColumn.setCellValueFactory(cellData ->
cellData.getValue().usernameProperty());

        TableColumn<UserCredentials, String> passwordColumn = new
TableColumn<>("Пароль");
        passwordColumn.setCellValueFactory(cellData ->
cellData.getValue().passwordProperty());

        // Кнопки для додавання і видалення користувачів
        Button addButton = new Button("Додати");
        Button removeButton = new Button("Видалити");

        // Встановлення фіксованої ширини для кнопок
        addButton.setPrefWidth(100);
        removeButton.setPrefWidth(100);

        HBox buttonsBox = new HBox(10, addButton, removeButton);
        buttonsBox.setAlignment(Pos.CENTER_RIGHT);
        buttonsBox.setPadding(new Insets(10));

        // Обробник натискання на кнопку додавання користувача
        addButton.setOnAction(event -> clickAddFtp());

        // Обробник натискання на кнопку видалення користувача
        removeButton.setOnAction(event -> {
            UserCredentials selectedItem =
credentialsTable.getSelectionModel().getSelectedItem();
```

```
            if (selectedItem != null) {
                credentialsTable.getItems().remove(selectedItem);
                listUserFtp.remove(selectedItem);
            }
        });

        // Додавання стовпців в таблицю користувачів
        credentialsTable.setItems(listUserFtp);
        credentialsTable.getColumns().addAll(Arrays.asList(usernameColumn,
passwordColumn));

        return new VBox(10, userSetupLabel, credentialsTable, buttonsBox);
    }

    /**
     * Метод для заповнення таблиці даними про файли.
     * Перебирає файли у папці та додає їх до таблиці.
     *
     * @param filesTable Таблиця, в яку додаються файли.
     */
    private static void populateFilesTable(TableView<FileItem> filesTable) {
        File folder = new File("src/main/resources/assets/files");
        File[] listOfFiles = folder.listFiles();

        // Перевіряємо, що список файлів не порожній
        if (listOfFiles != null) {
            for (File file : listOfFiles) {
                if (file.isFile()) {
                    // Отримуємо розмір файлу у рядковому форматі з урахуванням
точності
                    String fileSize = getFormattedFileSize(file.length());
                    // Створюємо об'єкт FileItem для поточного файлу
                    FileItem fileItem = new FileItem(file.getName(), fileSize);
                    // Додаємо об'єкт FileItem в дані таблиці
                    filesTable.getItems().add(fileItem);
                }
            }
            filesTable.refresh(); // Оновлюємо таблицю після додавання файлів
        } else {
            System.out.println("Немає файлів у каталозі.");
        }
    }
}
```

//httpSetting.java – класс для налаштування пртоколу HTTP

```
package com.example.demo1.settings.services;

import com.example.demo1.clasess.ImageInfo;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.control.*;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.shape.Line;

import java.util.ArrayList;
import java.util.List;

import static com.example.demo1.object.ObjectCreator.createCircleLine;

public class httpSetting {

    /**
     * Метод для налаштування компонентів HTTP протоколу.
```

```
     *
     * @param imageInfo Об'єкт ImageInfo, що містить інформацію про протоколи та
підключені елементи.
     * @param root      Кореневий вузол, до якого додаються компоненти HTTP
протоколу.
     */
    public static void clickHttpProtocols(ImageInfo imageInfo, BorderPane root)
{
        // Створення групи для радіокнопок протоколу HTTP
        ToggleGroup toggleGroupHttp = new ToggleGroup();

        // Створення радіокнопок для протоколу HTTP
        RadioButton httpOnRadioButton = new RadioButton("Увімкнено");
        RadioButton httpOffRadioButton = new RadioButton("Вимкнено");

        // Налаштування радіокнопок та додавання їх до групи
        configureRadioButton(httpOnRadioButton, httpOffRadioButton,
toggleGroupHttp, imageInfo.getHTTPConnect());

        // Створення групи для радіокнопок протоколу HTTPS
        ToggleGroup toggleGroupHttps = new ToggleGroup();

        // Створення радіокнопок для протоколу HTTPS
        RadioButton httpsOnRadioButton = new RadioButton("Увімкнено");
        RadioButton httpsOffRadioButton = new RadioButton("Вимкнено");

        // Налаштування радіокнопок та додавання їх до групи
        configureRadioButton(httpsOnRadioButton, httpsOffRadioButton,
toggleGroupHttps, imageInfo.getHTTPSConnect());

        // Створення контейнера для радіокнопок
        VBox radioButtonsContainer = new VBox(
                createLabeledRadioButtons("HTTP", httpOnRadioButton,
httpOffRadioButton),
                createLabeledRadioButtons("HTTPS", httpsOnRadioButton,
httpsOffRadioButton)
        );

        // Створення ChoiceBox для відображення назв підключених елементів
        ChoiceBox<String> connectedItemsChoiceBox = new ChoiceBox<>();

connectedItemsChoiceBox.getItems().addAll(getConnectedItemsNames(imageInfo.getCo
nnectedItem()));
        connectedItemsChoiceBox.setPadding(new Insets(0, 0, 0, 0));

        // Кнопка "Зберегти" для збереження обраних налаштувань
        Button saveButton = new Button("Зберегти");
        saveButton.setOnAction(e -> handleSaveButtonActionRadio(imageInfo,
toggleGroupHttp, toggleGroupHttps, connectedItemsChoiceBox.getValue()));
        saveButton.setPadding(new Insets(10));

        // Створення контейнера для всіх компонентів та його додавання до
кореневого вузла
        VBox container = new VBox(radioButtonsContainer,
connectedItemsChoiceBox, saveButton);
        container.setAlignment(Pos.CENTER);
        container.setSpacing(10);
        root.setCenter(container);
    }

    /**
     * Налаштовує радіокнопки та додає їх до групи.
     *
     * @param onRadioButton  Радіокнопка для увімкненого стану.
     * @param offRadioButton Радіокнопка для вимкненого стану.
     * @param toggleGroup    Група для управління станом радіокнопок.
     * @param isConnected    Прапорець, що вказує на поточний стан протоколу
(увімкнено/вимкнено).
     */
```

```
    private static void configureRadioButton(RadioButton onRadioButton,
RadioButton offRadioButton, ToggleGroup toggleGroup, boolean isConnected) {
        onRadioButton.setToggleGroup(toggleGroup);
        offRadioButton.setToggleGroup(toggleGroup);
        if (isConnected) {
            onRadioButton.setSelected(true);
        } else {
            offRadioButton.setSelected(true);
        }
    }

    /**
     * Створює групу радіокнопок з міткою.
     *
     * @param label          Текст мітки для групи радіокнопок.
     * @param onRadioButton  Радіокнопка для увімкненого стану.
     * @param offRadioButton Радіокнопка для вимкненого стану.
     * @return Група радіокнопок з міткою.
     */
    private static HBox createLabeledRadioButtons(String label, RadioButton
onRadioButton, RadioButton offRadioButton) {
        Label labelComponent = new Label(label);
        labelComponent.setPadding(new Insets(0, 10, 0, 0));
        HBox radioButtons = new HBox(labelComponent, onRadioButton,
offRadioButton);
        radioButtons.setAlignment(Pos.CENTER);
        return radioButtons;
    }

    /**
     * Обробляє дії після натискання кнопки "Зберегти" з урахуванням обраних
станів RadioButtons.
     *
     * @param imageInfo             Об'єкт ImageInfo, в якому встановлюються
стани протоколів.
     * @param toggleGroupHttp       Група для управління станом RadioButtons
протоколу HTTP.
     * @param toggleGroupHttps      Група для управління станом RadioButtons
протоколу HTTPS.
     * @param selectedConnectedItem Обраний елемент з ChoiceBox.
     */
    public static void handleSaveButtonActionRadio(ImageInfo imageInfo,
ToggleGroup toggleGroupHttp, ToggleGroup toggleGroupHttps, String
selectedConnectedItem) {
        boolean isHttpOn = ((RadioButton)
toggleGroupHttp.getSelectedToggle()).getText().equals("Увімкнено");
        boolean isHttpsOn = ((RadioButton)
toggleGroupHttps.getSelectedToggle()).getText().equals("Увімкнено");

        imageInfo.setHttpSelected(isHttpOn);
        imageInfo.setHttpsSelected(isHttpsOn);

        saveSelectedConnectionItem(selectedConnectedItem, imageInfo);
    }

    /**
     * Отримує список імен підключених елементів зі списку ImageInfo.
     *
     * @param connectedItems Список елементів типу ImageInfo.
     * @return Список імен підключених елементів.
     */
    public static List<String> getConnectedItemsNames(List<ImageInfo>
connectedItems) {
        List<String> names = new ArrayList<>();
        for (ImageInfo item : connectedItems) {
            names.add(item.getName());
        }
        return names;
    }
```

```
    /**
     * Зберігає обраний елемент з ChoiceBox у ImageInfo.
     *
     * @param selectedConnectionItem Обраний елемент.
     * @param imageInfo              Об'єкт ImageInfo, в який зберігається
обраний елемент.
     */
    private static void saveSelectedConnectionItem(String
selectedConnectionItem, ImageInfo imageInfo) {

        for (ImageInfo imageInfo1 : imageInfo.getConnectedItem()) {
            if (imageInfo1.getName().equals(selectedConnectionItem)) {
                imageInfo.setHttpItemConnect(imageInfo1);
                imageInfo1.setHttpItemConnect(imageInfo);
                break;
            }
        }

        System.out.println("Обраний елемент: " + selectedConnectionItem);

        String idLine = selectedConnectionItem + " " + imageInfo.getName();
        String idLine2 = imageInfo.getName() + " " + selectedConnectionItem;

        for (Line line : imageInfo.getConnectedLines()) {
            if (line.getId().equals(idLine) || line.getId().equals(idLine2)) {
                createCircleLine(line);
                break;
            }
        }
    }
}
```

//services.java – класс меню налаштувань кнопки sevices

```
   package com.example.demo1.settings.services;

import com.example.demo1.clasess.ImageInfo;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.geometry.Insets;
import javafx.scene.control.ListView;
import javafx.scene.control.ToolBar;
import javafx.scene.layout.BorderPane;

import static com.example.demo1.settings.services.ftpSetting.clickFtpProtocols;
import static
com.example.demo1.settings.services.httpSetting.clickHttpProtocols;

public class services {

    /**
     * Відображає вікно налаштувань для розділу "Сервіси".
     *
     * @param imageInfo Об'єкт класу ImageInfo.
     * @param root      Кореневий BorderPane.
     * @param toolBar   Панель інструментів.
     */
    public static void servicesSetting(ImageInfo imageInfo, BorderPane root,
ToolBar toolBar) {
        root.getChildren().clear();
        root.setTop(toolBar);

        // Створюємо ListView для елементів у лівій області
        ListView<String> servicesList = new ListView<>();
```

```
        servicesList.setPadding(new Insets(10, 10, 10, 10));

        // Додаємо елементи до ListView
        ObservableList<String> services =
FXCollections.observableArrayList("HTTP", "TCP", "UDP", "FTP", "SSH");
        servicesList.setItems(services);

        // Встановлюємо дію при виборі елемента

servicesList.getSelectionModel().selectedItemProperty().addListener((observable,
oldValue, newValue) -> {
            selectedService(newValue, root, imageInfo); // передаємо newValue
замість nameValue
        });

        servicesList.setMaxWidth(200);

        // Додаємо servicesList до кореневого вузла вашого основного вікна
        root.setLeft(servicesList);
    }

    /**
     * Обробляє вибір сервісу в розділі "Сервіси".
     *
     * @param newValue  Обраний сервіс.
     * @param root      Кореневий BorderPane.
     * @param imageInfo Об'єкт класу ImageInfo.
     */
    private static void selectedService(String newValue, BorderPane root,
ImageInfo imageInfo) {
        if (newValue.equals("HTTP")) {
            clickHttpProtocols(imageInfo, root);
        }

        if (newValue.equals("FTP")) {
            clickFtpProtocols(imageInfo, root);
        }
    }

}
```

//ConnectionType.java – класс для меню зєднання елементів

```
package com.example.demo1.settings;

import com.example.demo1.clasess.ImageInfo;
import com.example.demo1.protocols.BrowserSimulator;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.ToolBar;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.shape.Line;
import javafx.stage.Stage;

import java.util.HashMap;
import java.util.Objects;

import static com.example.demo1.Constants.PATH_GOOGLE;
import static com.example.demo1.Constants.PATH_TERMINAL;
import static com.example.demo1.ErrorApplication.errorConnectionItemWeb;
```

```
import static com.example.demo1.ErrorApplication.errorIpAddress;
import static com.example.demo1.settings.services.services.servicesSetting;


public class ConnectionType {
    private static final HashMap<String, String> setting = new HashMap<>();

    /**
     * Відкриває діалогове вікно налаштувань для елемента ImageInfo.
     *
     * @param imageInfo Екземпляр класу ImageInfo, для якого відкривається вікно
налаштувань.
     */
    public static void dialogSettingItem(ImageInfo imageInfo) {
        // Створюємо BorderPane
        BorderPane root = new BorderPane();

        // Створюємо горизонтальну панель інструментів (ToolBar)
        ToolBar toolBar = new ToolBar();

        // Створюємо кнопки для меню
        Button physicalConfigButton = new Button("Physical Config");
        Button servicesButton = new Button("Services");
        Button desktopButton = new Button("Desktop");
        Button connectionButton = new Button("Connection");

        // Додаємо кнопки в панель інструментів
        toolBar.getItems().addAll(physicalConfigButton, servicesButton,
desktopButton, connectionButton);

        // Встановлюємо ToolBar в верхню область BorderPane
        root.setTop(toolBar);

        // Створюємо сцену
        Scene scene = new Scene(root, 700, 520);

        // Створюємо новий Stage (вікно)
        Stage stage = new Stage();

        // Встановлюємо сцену на Stage
        stage.setScene(scene);

        // Обробник події для кнопки "Desktop"
        desktopButton.setOnAction(event -> {
            root.getChildren().clear();
            root.setTop(toolBar);
            root.setCenter(desktopSetting(imageInfo));
        });

        servicesButton.setOnAction(event -> {
            servicesSetting(imageInfo, root, toolBar);
        });

        connectionButton.setOnAction(event -> {
            clickConnectionButton(imageInfo, root, toolBar);
        });

        // Встановлюємо заголовок Stage
        stage.setTitle("Menu Setting " + imageInfo.getName());

        // Показуємо Stage
        stage.show();
    }

    /**
     * Створює і повертає сітку налаштувань для розділу "Desktop".
     *
     * @param imageInfo Екземпляр класу ImageInfo.
     * @return GridPane для розділу "Desktop".
```

```
     */
    public static GridPane desktopSetting(ImageInfo imageInfo) {
        GridPane grid = new GridPane();
        grid.setHgap(10);
        grid.setVgap(10);
        grid.setPadding(new Insets(10, 10, 10, 10));


        // Лінія-роздільник зліва
        Line leftLine = new Line(0, 0, 0, 400);
        grid.add(leftLine, 0, 0, 1, 11);

        // Конфігурація IPv4
        grid.add(new Label("IPv4 Configuration"), 1, 0, 2, 1);

        // Додаємо лівий відступ для міток
        GridPane.setMargin(new Label("IP Address"), new Insets(10, 0, 0, 10));
        grid.add(new Label("IP Address"), 1, 1);
        TextField ipAddressField = new TextField(imageInfo.getIPAddress());
        grid.add(ipAddressField, 2, 1);

        GridPane.setMargin(new Label("Subnet Mask"), new Insets(10, 0, 0, 10));
        grid.add(new Label("Subnet Mask"), 1, 2);
        TextField subnetMaskField = new TextField(imageInfo.getSubnetMask());
        grid.add(subnetMaskField, 2, 2);

        GridPane.setMargin(new Label("Default Gateway"), new Insets(10, 0, 0,
10));
        grid.add(new Label("Default Gateway"), 1, 3);
        TextField defaultGatewayField = new
TextField(imageInfo.getDefaultGateway());
        grid.add(defaultGatewayField, 2, 3);

        GridPane.setMargin(new Label("DNS Server"), new Insets(10, 0, 0, 10));
        grid.add(new Label("DNS Server"), 1, 4);
        TextField dnsServerField = new TextField(imageInfo.getDNSServer());
        grid.add(dnsServerField, 2, 4);

        // Лінія-роздільник справа
        Line rightLine = new Line(0, 0, 0, 400);
        grid.add(rightLine, 3, 0, 1, 11);

        // Лінія-роздільник
        Line line = new Line(0, 0, 300, 0);
        grid.add(line, 1, 5, 2, 1);

        // Конфігурація IPv6
        grid.add(new Label("IPv6 Configuration"), 1, 6, 2, 1);

        GridPane.setMargin(new Label("IPv6 Address"), new Insets(10, 0, 0, 10));
        grid.add(new Label("IPv6 Address"), 1, 7);
        TextField ipv6AddressField = new TextField(imageInfo.getIPv6Address());
        grid.add(ipv6AddressField, 2, 7);

        GridPane.setMargin(new Label("Link Local Address"), new Insets(10, 0, 0,
10));
        grid.add(new Label("Link Local Address"), 1, 8);
        TextField linkLocalAddressField = new
TextField(imageInfo.getLinkLocalAddress());
        grid.add(linkLocalAddressField, 2, 8);

        GridPane.setMargin(new Label("IPv6 Gateway"), new Insets(10, 0, 0, 10));
        grid.add(new Label("IPv6 Gateway"), 1, 9);
        TextField ipv6GatewayField = new TextField(imageInfo.getIPv6Gateway());
        grid.add(ipv6GatewayField, 2, 9);

        GridPane.setMargin(new Label("IPv6 DNS Server"), new Insets(10, 0, 0,
10));
        grid.add(new Label("IPv6 DNS Server"), 1, 10);
```

```
        TextField ipv6DnsServerField = new
TextField(imageInfo.getIPv6DNSServer());
        grid.add(ipv6DnsServerField, 2, 10);

        // Кнопка "Зберегти"
        Button saveButton = new Button("Зберегти");
        GridPane.setMargin(saveButton, new Insets(10, 0, 0, 10));
        grid.add(saveButton, 1, 12, 2, 1);

        // Обробник події для кнопки "Зберегти"
        saveButton.setOnAction(event -> {
            // Отримуємо значення з текстових полів
            String ipAddress = ipAddressField.getText();
            String subnetMask = subnetMaskField.getText();
            String defaultGateway = defaultGatewayField.getText();
            String dnsServer = dnsServerField.getText();

            String ipv6Address = ipv6AddressField.getText();
            String linkLocalAddress = linkLocalAddressField.getText();
            String ipv6Gateway = ipv6GatewayField.getText();
            String ipv6DnsServer = ipv6DnsServerField.getText();

            // Зберігаємо значення в HashMap
            setting.put("IPAddress", ipAddress);
            setting.put("SubnetMask", subnetMask);
            setting.put("DefaultGateway", defaultGateway);
            setting.put("DNSServer", dnsServer);

            setting.put("IPv6Address", ipv6Address);
            setting.put("LinkLocalAddress", linkLocalAddress);
            setting.put("IPv6Gateway", ipv6Gateway);
            setting.put("IPv6DNSServer", ipv6DnsServer);

            // Встановлюємо значення в ImageInfo
            imageInfo.setConnectionSettings(setting);

            System.out.println("Дані збережено в HashMap!");
            System.out.println(setting);
        });

        return grid;
    }

    /**
     * Обробляє натискання кнопки "Підключення".
     *
     * @param imageInfo Екземпляр класу ImageInfo.
     * @param root      Кореневий BorderPane.
     * @param toolBar   Панель інструментів.
     */
    private static void clickConnectionButton(ImageInfo imageInfo, BorderPane
root, ToolBar toolBar) {
        // Створюємо кнопку з зображенням та назвою
        Button connectionButton1 = createImageButton(imageInfo, PATH_GOOGLE,
"Web Browser");

        // Створюємо другу кнопку
        Button connectionButton2 = createImageButton(imageInfo, PATH_TERMINAL,
"Terminal");

        // Створюємо горизонтальний контейнер для розміщення кнопок
        HBox hBox = new HBox(10);
        hBox.getChildren().addAll(connectionButton1, connectionButton2);

        root.getChildren().clear();
        root.setTop(toolBar);
        root.setCenter(hBox);
    }
```

```
    /**
     * Створює кнопку з зображенням і текстом.
     *
     * @param imageInfo  Екземпляр класу ImageInfo.
     * @param imagePath  Шлях до зображення.
     * @param buttonText Текст кнопки.
     * @return Кнопка з зображенням і текстом.
     */
    private static Button createImageButton(ImageInfo imageInfo, String
imagePath, String buttonText) {
        // Завантажуємо зображення
        Image image = new
Image(Objects.requireNonNull(ConnectionType.class.getResourceAsStream(imagePath)
));

        ImageView imageView = new ImageView(image);
        imageView.setFitWidth(50);
        imageView.setFitHeight(50);

        // Створюємо кнопку з зображенням і текстом
        Button button = new Button(buttonText, imageView);

        // Налаштовуємо стиль для кнопки (необов'язково)
        button.setStyle("-fx-font-size: 12; -fx-graphic-text-gap: 10;");

        // Налаштовуємо обробник події для кнопки (за потреби)
        button.setOnAction(event -> {
            if (buttonText.equals("Web Browser")) {
                if (imageInfo.getConnectedItem().isEmpty()) {
                    errorConnectionItemWeb();
                } else if (imageInfo.getIPAddress() == null) {
                    errorIpAddress();
                    dialogSettingItem(imageInfo);
                } else {
                    BrowserSimulator.runServer(imageInfo);
                }
            }

        });

        return button;
    }
}
```

//ErrorApplication.java – класс для оборки та повідомлень помилок

```
пакет com.example.demo1;

імпорт javafx.scene.control.Alert;

клас ErrorApplication {

    публічний статичний void errorIpAddress() {
        // Створюємо спливаюче вікно з попередженням
        Alert alert = новий Alert(Alert.AlertType.WARNING);
        alert.setTitle("Увага");
        alert.setHeaderText(null);
        alert.setContentText("IP-адресу не вказано!");

        // Показуємо спливаюче вікно та очікуємо, доки користувач його закриє
        alert.showAndWait();
    }

    публічний статичний void errorConnectionItemWeb() {
        Alert alert = новий Alert(Alert.AlertType.WARNING);
        alert.setTitle("Увага");
        alert.setHeaderText(null);
        alert.setContentText("Відсутні підключені елементи!");
```

```
        alert.showAndWait();
    }
}
```

//HelloApplication.java – головний класс програми

```
пакет com.example.demo1;

імпорт javafx.application.Application;
імпорт javafx.scene.Scene;
імпорт javafx.scene.control.MenuBar;
імпорт javafx.scene.layout.BorderPane;
імпорт javafx.scene.layout.Pane;
імпорт javafx.stage.Stage;

import java.util.Objects;

клас HelloApplication розширює Application {

    публічний статичний void main(String[] args) {
        launch();
    }


    публічний void start(Stage primaryStage) {
        // Створення кореневого макету з BorderPane
        BorderPane root = новий BorderPane();

        // Ініціалізація контейнера для зображень
        Pane imageContainer = новий Pane();

        // Створення меню та додавання його до кореневого макету
        MenuBar menuBar = MenuCreator.createMenu(primaryStage, imageContainer);
        root.setTop(menuBar);

        // Додавання контейнера з зображеннями на робочу область
        root.setCenter(imageContainer);

        // Створення сцени та встановлення на ній кореневого макету
        Scene scene = новий Scene(root, 800, 500);

        // Встановлення сцени в вікні додатка
        primaryStage.setTitle("Дипломна робота");
        primaryStage.setScene(scene);

        // Використання стилів на сцені

scene.getStylesheets().add(Objects.requireNonNull(getClass().getResource("/asset
s/style.css")).toExternalForm());
        scene.getRoot().setId("root");  // Встановлення ідентифікатора для
кореневого контейнера
        menuBar.setId("menuBar");  // Встановлення ідентифікатора для верхньої
панелі меню
        imageContainer.setId("imageContainer");  // Встановлення ідентифікатора
для контейнера з зображеннями

        // Показ вікна додатка
        primaryStage.show();
    }
}
```

//MenuCreator.java – калсс для створення головного меню

```
package com.example.demo1;

import com.example.demo1.clasess.ImageInfo;
```

```
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.Pane;
import javafx.scene.layout.VBox;
import javafx.stage.FileChooser;
import javafx.stage.Modality;
import javafx.stage.Stage;
import javafx.stage.StageStyle;

import java.io.*;
import java.util.Objects;

import static com.example.demo1.settings.ConnectionType.dialogSettingItem;
import static com.example.demo1.OpenFile.*;
import static com.example.demo1.object.ConnectObject.connectItemDialog;
import static com.example.demo1.object.ConnectObject.disconnectItem;
import static com.example.demo1.Constants.*;
import static com.example.demo1.object.ObjectCreator.createObject;

public class MenuCreator {
    @SuppressWarnings("exports")
    public static MenuBar createMenu(Stage primaryStage, Pane container) {
        imageContainer = container;

        MenuBar menuBar = new MenuBar();

        // Створення меню "File"
        Menu fileMenu = new Menu("Файл");

        MenuItem createFileItem = new MenuItem("Створити файл");
        MenuItem openFileItem = new MenuItem("Відкрити файл");
        MenuItem saveAsFileItem = new MenuItem("Зберегти як");
        MenuItem saveFileItem = new MenuItem("Зберегти файл");
        MenuItem clearItem = new MenuItem("Очистити");
        MenuItem exitItem = new MenuItem("Вийти");

        fileMenu.getItems().addAll(createFileItem, openFileItem, saveAsFileItem,
saveFileItem, new SeparatorMenuItem(), clearItem, exitItem);

        // Обробник подій для пункту меню "Створити файл"
        createFileItem.setOnAction(event -> {
            FileChooser fileChooser = new FileChooser();
            FileChooser.ExtensionFilter extFilter = new
FileChooser.ExtensionFilter("Серіалізовані файли (*.ser)", "*.ser");
            fileChooser.getExtensionFilters().add(extFilter);
            File file = fileChooser.showSaveDialog(primaryStage);

            if (file != null) {
                try (ObjectOutputStream ignored = new ObjectOutputStream(new
FileOutputStream(file))) {
                    // Тут можна використовувати ваш об'єкт, наприклад,
listImageInfo
                    NAME_FILE_NEW = file.getAbsolutePath();
                    System.out.println("Об'єкт успішно записаний у файл: " +
file.getAbsolutePath());
                } catch (IOException e) {
                    e.printStackTrace();
                    // Обробка винятку, пов'язаного з записом у файл
                }
            }
        });

        // Обробник подій для пункту меню "Вийти"
        exitItem.setOnAction(event -> primaryStage.close());

        // Обробник подій для пункту меню "Відкрити файл"
        openFileItem.setOnAction(event -> {
            FileChooser fileChooser = new FileChooser();
```

```
            fileChooser.setTitle("Відкрити файл");
            File file = fileChooser.showOpenDialog(primaryStage);

            if (file != null) {
                NAME_FILE_NEW = file.getAbsolutePath();
                ImageInfoDeserializationExample(file.getAbsolutePath());
            }

        });

        // Обробник подій для пункту меню "Очистити"
        clearItem.setOnAction(event -> {
            imageContainer.getChildren().clear();
            lineList.clear();
            imageList.clear();
        });

        // Обробник подій для пункту меню "Зберегти файл"
        saveFileItem.setOnAction(event -> {
            ImageInfoSerializationExample(NAME_FILE_NEW);
        });

        // Обробник подій для пункту меню "Зберегти як"
        saveAsFileItem.setOnAction(event -> {
            FileChooser fileChooser = new FileChooser();

            // Встановлення фільтра розширень для файлів .ser
            FileChooser.ExtensionFilter extFilter = new
FileChooser.ExtensionFilter("Serialized Files (*.ser)", "*.ser");
            fileChooser.getExtensionFilters().add(extFilter);

            File file = fileChooser.showSaveDialog(primaryStage); //
Використовуйте showSaveDialog

            if (file != null) {
                NAME_FILE_NEW = file.getAbsolutePath();
                ImageInfoSerializationExample(file.getAbsolutePath());
            }
        });


        // Додавання меню "File" в MenuBar
        menuBar.getMenus().add(fileMenu);

        // Додавання меню "Додати"
        Menu addMenu = new Menu("Додати");
        MenuItem createServerItem = new MenuItem("Створити сервер");
        MenuItem createPCItem = new MenuItem("Створити ПК");
        MenuItem createCommutatorItem = new MenuItem("Створити комутатор");
        MenuItem createPrinterItem = new MenuItem("Створити принтер");
        MenuItem createRouterItem = new MenuItem("Створити маршрутизатор");

        addMenu.getItems().addAll(createServerItem, createPCItem,
createCommutatorItem, createPrinterItem, createRouterItem);

        // Додавання меню "Додати" в MenuBar
        menuBar.getMenus().add(addMenu);

        createServerItem.setOnAction(event -> { createObject(PATH_SERVER); });
        createPCItem.setOnAction(event -> { createObject(PATH_PC); });
        createCommutatorItem.setOnAction(event -> {
createObject(PATH_COMUTATOR); });
        createPrinterItem.setOnAction(event -> { createObject(PATH_PRINTER); });
        createRouterItem.setOnAction(event -> { createObject(PATH_ROUTER); });


        return menuBar;
    }
```

```
    @SuppressWarnings("exports")
    public static ContextMenu createContextMenu(ImageInfo imageInfo) {
        ContextMenu contextMenu = new ContextMenu();

        if (imageInfo.isMovable()) {
            MenuItem deleteItem = new MenuItem("Видалити");
            MenuItem infoItem = new MenuItem("Інформація");
            MenuItem propertiesItem = new MenuItem("Властивості");
            MenuItem connectItem = new MenuItem("Підключити");
            MenuItem settingItem = new MenuItem("Налаштування");

            contextMenu.getItems().addAll(deleteItem, infoItem, propertiesItem,
connectItem, settingItem);

            // Обробник подій для пункту "Видалити"
            deleteItem.setOnAction(event -> {
                VBox parentBox = (VBox) imageInfo.getImageView().getParent();

                // Передбачаємо, що Label завжди знаходиться після ImageView
                int labelIndex =
parentBox.getChildren().indexOf(imageInfo.getImageView()) + 1;
                if (labelIndex < parentBox.getChildren().size()) {
                    parentBox.getChildren().remove(labelIndex);
                }

                parentBox.getChildren().remove(imageInfo.getImageView());
                imageList.remove(imageInfo);
            });

            infoItem.setOnAction(event -> {
                showPropertiesDialog(imageInfo);
            });

            connectItem.setOnAction(event -> {
                connectItemDialog(imageInfo);
            });

            settingItem.setOnAction(event -> {
                dialogSettingItem(imageInfo);
            });
        } else {
            MenuItem connectItem = new MenuItem("Підключити");
            MenuItem infoItem = new MenuItem("Інформація");
            MenuItem disconnectItem = new MenuItem("Відключити");
            MenuItem settingItem = new MenuItem("Налаштування");

            contextMenu.getItems().addAll(connectItem, infoItem, disconnectItem,
settingItem);

            connectItem.setOnAction(event-> {
                connectItemDialog(imageInfo);
            });

            infoItem.setOnAction(event -> {
                showPropertiesDialog(imageInfo);
            });

            disconnectItem.setOnAction(event -> {
                disconnectItem(imageInfo);
            });

            settingItem.setOnAction(event -> {
                dialogSettingItem(imageInfo);
            });

        }
        return contextMenu;
```

```
    }

    public static void showPropertiesDialog(ImageInfo imageInfo) {
        Stage dialogStage = new Stage();
        dialogStage.initModality(Modality.APPLICATION_MODAL);
        dialogStage.initStyle(StageStyle.UTILITY);
        dialogStage.setTitle("Інформація");

        Label info = new Label(imageInfo.getInfo());

        VBox vbox = new VBox(10);
        vbox.setPadding(new Insets(10));
        vbox.getChildren().addAll(info);

        Scene dialogScene = new Scene(vbox, 600, 400);

dialogScene.getStylesheets().add(Objects.requireNonNull(MenuCreator.class.getRes
ource(Path_STYLE)).toExternalForm());

        // Застосування стилів до вікна
        dialogStage.setScene(dialogScene);
        dialogScene.getRoot().setId("infoStage");

        // Заборонити зміну розміру вікна
        dialogStage.setResizable(false);

        dialogStage.showAndWait();
    }
}
```

//OpenFile.java – клаас для відкриття та збереження файлів

```
package com.example.demo1;

import com.example.demo1.clasess.ImageInfo;

import java.io.*;
import java.util.*;

import static com.example.demo1.Constants.*;
import static com.example.demo1.Constants.imageList;
import static com.example.demo1.Redraw.redrawImageContainer;
import static com.example.demo1.object.ConnectObject.connectItemLine;
import static com.example.demo1.object.ConnectObject.createImageView;

/**
 * Утилітарний клас для відкриття та збереження файлів проекту.
 */
public class OpenFile {

    public static void ImageInfoSerializationExample(String filePath) {
        ArrayList<ImageInfo> arrayList = new ArrayList<>();

        for (ImageInfo imageInfo1 : Constants.imageList) {
            ImageInfo imageInfo = new ImageInfo();

            // Копіюємо дані з imageInfo1 в новий об'єкт imageInfo
            imageInfo.setType(imageInfo1.getType());
            imageInfo.setName(imageInfo1.getName());
            imageInfo.setX(imageInfo1.getX());
            imageInfo.setY(imageInfo1.getY());
            imageInfo.setMovable(imageInfo1.isMovable());
            imageInfo.setConnectedItems(imageInfo1.getConnectedItem());
            imageInfo.setIdLine(imageInfo1.getIdLine());
            imageInfo.setConnectedLiness(imageInfo1.getConnectedLines());
            imageInfo.setImageView(imageInfo1.getImageView());

            arrayList.add(imageInfo);
```

```
        }


        try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(filePath))) {
            // Записуємо об'єкт в файл
            outputStream.writeObject(arrayList);
            System.out.println("ImageInfo успішно записаний в файл: " +
filePath);
        } catch (IOException e) {
            e.printStackTrace();
            System.err.println("Помилка при записі об'єкта ImageInfo в файл");
        }
    }

    public static void ImageInfoDeserializationExample(String filePath) {
        Constants.lineList.clear();
        Constants.imageList.clear();
        List<ImageInfo> imageInfoList = new ArrayList<>();

        try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(filePath))) {
            Object object = inputStream.readObject();

            if (object instanceof List<?> list) {
                // Тепер ви можете безпечно використовувати list
                for (Object item : list) {
                    if (item instanceof ImageInfo imageInfo) {
                        imageInfoList.add(imageInfo);
                    }
                }
            } else {
                System.err.println("Line 73(OpenFile). list is not a
ImageInfo");
            }

            for (ImageInfo imageInfo1 : imageInfoList) {
                ImageInfo imageInfo = new ImageInfo();

                // Копіюємо дані з imageInfo1 в новий об'єкт imageInfo
                imageInfo.setType(imageInfo1.getType());
                imageInfo.setName(imageInfo1.getName());
                imageInfo.setX(imageInfo1.getX());
                imageInfo.setY(imageInfo1.getY());
                imageInfo.setMovable(imageInfo1.isMovable());
                imageInfo.setConnectedItems(imageInfo1.getConnectedItem());
                imageInfo.setIdLine(imageInfo1.getIdLine());
                imageInfo.setConnectedLiness(imageInfo1.getConnectedLines());
                imageInfo.setImageView(imageInfo1.getImageView());

                imageList.add(imageInfo);
            }


            drawOpenFile();
            System.out.println("ImageInfo успішно зчитано з файлу: " +
filePath);

        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
            System.err.println("Помилка при читанні об'єктів ImageInfo з
файлу");
        }
    }

    public static void drawOpenFile() {
        for (ImageInfo imageInfo : imageList) {
            switch (imageInfo.getType()) {
                case "pc" -> imageInfo.setImageView(createImageView(PATH_PC));
```

```
                case "server" ->
imageInfo.setImageView(createImageView(PATH_SERVER));
                case "printer" ->
imageInfo.setImageView(createImageView(PATH_PRINTER));
                case "comutator" ->
imageInfo.setImageView(createImageView(PATH_COMUTATOR));
                case "router" ->
imageInfo.setImageView(createImageView(PATH_ROUTER));
            }
        }

        Set<String> connectedPairs = new HashSet<>();


        for (ImageInfo imageInfo : imageList) {
            for (ImageInfo imageInfo1 : imageInfo.getConnectedItem()) {
                String pair = imageInfo.getName() + " " + imageInfo1.getName();
                String pair2 = imageInfo1.getName() + " " + imageInfo.getName();

                // Перевіряємо, чи не з'єднували ми вже ці два елементи
                if (!connectedPairs.contains(pair)) {
                    if (!connectedPairs.contains(pair2)) {
                        connectItemLine(imageInfo1, imageInfo);
                    } else {
                        connectItemLine(imageInfo, imageInfo1);
                    }

                    // Додаємо пару в множину, щоб уникнути повторного з'єднання
                    connectedPairs.add(pair);
                }
            }
        }

        redrawImageContainer();
    }

}
```

//Redraw.java – класс для обновлення головного вікна

```
package com.example.demo1;

import com.example.demo1.clasess.ImageInfo;
import javafx.scene.control.ContextMenu;
import javafx.scene.control.Label;
import javafx.scene.image.ImageView;
import javafx.scene.layout.VBox;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Line;


import static com.example.demo1.Constants.*;
import static com.example.demo1.MenuCreator.createContextMenu;
import static com.example.demo1.object.MovedObject.setDragEventHandlers;

public class Redraw {
    /**
     * Перемалювання
     */
    public static void redrawImageContainer() {
        // Очистити контейнер
        imageContainer.getChildren().clear();


        // Додати лінії знову в контейнер
        for (Line line : lineList) {
            imageContainer.getChildren().add(line);
        }
```

```
        for (Circle circle: circleList){
            circle.setStyle("-fx-opacity: 1.0;");
            imageContainer.getChildren().add(circle);
        }

        // Додати залишкові елементи з imageList
        for (ImageInfo imageInfo : imageList) {
            ImageView imageView = imageInfo.getImageView();
            Label label = new Label(imageInfo.getName());

            VBox imageBox = new VBox();

            imageBox.setMaxWidth(50);
            imageBox.setMaxHeight(50);

            // Встановити поточні координати
            imageBox.setTranslateX(imageInfo.getX());
            imageBox.setTranslateY(imageInfo.getY());

            // Встановлення обробника подій для контекстного меню
            ContextMenu contextMenu = createContextMenu(imageInfo);
            imageBox.setOnContextMenuRequested(event ->
contextMenu.show(imageView, event.getScreenX(), event.getScreenY()));

            setDragEventHandlers(imageBox, imageInfo);

            imageBox.getChildren().addAll(imageView, label);

            // Додати в контейнер
            imageContainer.getChildren().add(imageBox);
        }

    }

    public static ImageInfo getImageInfoByType(String type) {
        return imageList.stream().filter(imageInfo ->
imageInfo.getName().equals(type)).findFirst().orElse(null);
    }

}
```

//ImageInfo.java – класс для обєктів робочого стола програми

```
package com.example.demo1.clasess;

import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.shape.Line;

import java.io.*;
import java.util.*;

import static com.example.demo1.Constants.*;

public class ImageInfo implements Serializable {
    @Serial
    private static final long serialVersionUID = 6556054599723165981L; //
унікальне значення для серелізації

    private transient ImageView imageView = new ImageView();
    private String type;
    private double x, y;
    private boolean movable = true;
    private String name;
```

```
    private  ArrayList<ImageInfo> listConnectedItem = new ArrayList<>();
    private String idLine = "";
    private  transient ArrayList<Line> connectedLines = new ArrayList<>();
    private  transient Commutator commutator;
    private transient  PC pc;
    private  transient Printer printer;
    private transient  Router router;
    private  transient Server server;

    private String IPAddress = "127.0.0.1";
    private String SubnetMask = "255.255.255.0";
    private String DefaultGateway = "192.168.1.1";
    private String DNSServer = "8.8.8.8";

    private String IPv6Address = "2001:0db8:85a3:0000:0000:8a2e:0370:7334";
    private String LinkLocalAddress = "fe80::1a2b:3c4d:5e6f";
    private String IPv6Gateway = "192.168.1.1";
    private String IPv6DNSServer = "8.8.8.8";

    private ImageInfo httpConnectItem;

    private boolean httpConnect;

    private boolean httpsConnect;

    @Serial
    private void writeObject(ObjectOutputStream out) throws IOException {
        out.defaultWriteObject();

        // Зберігаємо стан ImageView
        if (imageView.getImage() != null) {
            Image image = imageView.getImage();
            String imageUrl = image.getUrl();
            out.writeObject(imageUrl);
        } else {
            out.writeObject(null);
        }

    }

    @Serial
    private void readObject(ObjectInputStream in) throws IOException,
ClassNotFoundException {
        in.defaultReadObject();

        // Відновлюємо стан ImageView
        String imageUrl = (String) in.readObject();
        if (imageUrl != null) {
            Image image = new Image(imageUrl);
            imageView = new ImageView(image);
        }

    }


    public ImageInfo() {
        // Порожній конструктор
    }

    public ImageInfo(ImageView imageView, String type, double x, double y,
Commutator commutator, PC pc, Printer printer, Router router, Server server) {
        this.imageView = imageView;
        this.type = type;
        this.x = x;
        this.y = y;
        this.server = server;
        this.commutator = commutator;
        this.pc = pc;
```

```
        this.printer = printer;
        this.router = router;
        this.connectedLines = new ArrayList<>();
        this.listConnectedItem = new ArrayList<>();
    }

    public String getInfo() {
        StringJoiner joiner = new StringJoiner(", ");

        for (ImageInfo connectedItem : listConnectedItem) {
            joiner.add(connectedItem.getName());
        }

        StringJoiner joinerLine = new StringJoiner(", ");

        for (Line line : connectedLines) {
            joinerLine.add(line.getId());
        }

        return "Тип: " + type + "\n" +
                "Назва: " + name + "\n" +
                "X: " + x + "\n" +
                "Y: " + y + "\n" +
                "Рухомий: " + movable + "\n" +
                "Підключені елементи: " + joiner + "\n" +
                "Id лінії: " + idLine + "\n" +
                "Підключені лінії: " + joinerLine + "\n" +
                "Підключення за протоколом HTTP: " + httpConnect + "\n" +
                "Підключений елемент за протоколом HTTP: " +
httpConnectItem.getName() + "\n" +
                "IP-адреса: " + IPAddress;
    }

    @Override
    public String toString() {
        StringJoiner joiner = new StringJoiner(", ");

        for (ImageInfo connectedItem : listConnectedItem) {
            joiner.add(connectedItem.getName());
        }

        StringJoiner joinerLine = new StringJoiner(", ");

        if (connectedLines == null) joinerLine.add("null");
        else {
            for (Line line : connectedLines) {
                joinerLine.add(line.getId());
            }
        }

        return "Тип: " + type + "\n" +
                "Назва: " + name + "\n" +
                "X: " + x + "\n" +
                "Y: " + y + "\n" +
                "Рухомий: " + movable + "\n" +
                "Підключені елементи: " + joiner + "\n" +
                "Id лінії: " + idLine + "\n" +
                "Підключені лінії: " + joinerLine;
    }

    public ImageView getImageView() {
        if (imageView == null){
            imageView = new ImageView();
        }
        return imageView;
    }

    public void setImageView(ImageView imageView) {
        this.imageView = imageView;
```

```
}

public String getType() {
    return type;
}

public void setType(String type) {
    this.type = type;
}

public double getX() {
    return x;
}

public void setX(double x) {
    this.x = x;
}

public double getY() {
    return y;
}

public void setY(double y) {
    this.y = y;
}

public boolean isMovable() {
    return movable;
}

public void setMovable(boolean movable) {
    this.movable = movable;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public void setDisabled(boolean disabled) {
    if (imageView == null){
        imageView = new ImageView();
    }
    imageView.setDisable(disabled);
    if (disabled) {
        imageView.setStyle("-fx-opacity: 0.5;");
    } else {
        imageView.setStyle("");
    }
}

public ArrayList<ImageInfo> getConnectedItem() {
    return listConnectedItem;
}

public void returnItemConnectedList(String name) {
    Iterator<ImageInfo> iterator = listConnectedItem.iterator();
    while (iterator.hasNext()) {
        ImageInfo imageInfo = iterator.next();
        if (imageInfo.getName().equals(name)) {
            iterator.remove();
            break;
        }
    }
}
```

```
    public void setConnectedItem(ImageInfo connectedItem) {
        this.listConnectedItem.add(connectedItem);
    }

    public void setConnectedItems(ArrayList<ImageInfo> imageInfoArrayList){
        this.listConnectedItem = imageInfoArrayList;
    }

    public void setIdLine(String id) {
        this.idLine = id;
    }

    public String getIdLine(){
        return idLine;
    }

    public void setLineConnectedItem(Line line) {
        if (this.connectedLines == null){
            this.connectedLines = new ArrayList<>();
        }

        connectedLines.add(line);
    }

    public ArrayList<Line> getConnectedLines() {
        return connectedLines;
    }

    public void removeConnectItem(ImageInfo imageInfo){
        imageInfo.setDisabled(false);
        imageInfo.setMovable(true);

        String idLine = imageInfo.getName() + " " + this.name;
        String idLine2 = this.name + " " + imageInfo.getName();

        connectedLines.removeIf(line -> line.getId().equals(idLine) ||
line.getId().equals(idLine2));
        lineList.removeIf(line -> line.getId().equals(idLine) ||
line.getId().equals(idLine2));

        imageInfo.setIdLine("");
        listConnectedItem.remove(imageInfo);

    }

    public void clearConnectedLines(){
        this.idLine = "";
        this.connectedLines.clear();
    }

    public void setServer(Server server){

        if (ipServer.isEmpty()) {
            this.IPAddress = "127.0.0.1";
        }else{
            // Розділяємо IP-адресу на окремі частини по крапках
            String[] parts = ipServer.get(ipServer.size() - 1).split("\\.");

            // Отримуємо останню частину IP-адреси і перетворюємо її в ціле
число
            // Збільшуємо останню частину на одиницю
            int lastPart = Integer.parseInt(parts[3]) + 1;

            parts[3] = String.valueOf(lastPart);

            this.IPAddress = String.join(".", parts);
        }
```

```
        this.server = server;
    }

    public void setCommutator(Commutator commutator) {

        if (ipCommutator.isEmpty()) {
            this.IPAddress = "192.168.1.200";
        }else{
            // Розділяємо IP-адресу на окремі частини по крапках
            String[] parts = ipCommutator.get(ipCommutator.size() -
1).split("\\.");

            // Отримуємо останню частину IP-адреси і перетворюємо її в ціле
число
            // Збільшуємо останню частину на одиницю
            int lastPart = Integer.parseInt(parts[3]) + 1;

            parts[3] = String.valueOf(lastPart);

            this.IPAddress = String.join(".", parts);
        }

        this.commutator = commutator;
    }

    public void setPC(PC pc) {
        if (ipPc.isEmpty()) {
            this.IPAddress = "192.168.1.2";
        }else{
            // Розділяємо IP-адресу на окремі частини по крапках
            String[] parts = ipPc.get(ipPc.size() - 1).split("\\.");

            // Отримуємо останню частину IP-адреси і перетворюємо її в ціле
число
            // Збільшуємо останню частину на одиницю
            int lastPart = Integer.parseInt(parts[3]) + 1;

            parts[3] = String.valueOf(lastPart);

            this.IPAddress = String.join(".", parts);
        }

        this.pc = pc;
    }

    public void setPrinter(Printer printer) {

        if (ipPrinter.isEmpty()) {
            this.IPAddress = "192.165.0.1";
        }else{
            // Розділяємо IP-адресу на окремі частини по крапках
            String[] parts = ipPrinter.get(ipPrinter.size() - 1).split("\\.");

            // Отримуємо останню частину IP-адреси і перетворюємо її в ціле
число
            // Збільшуємо останню частину на одиницю
            int lastPart = Integer.parseInt(parts[3]) + 1;

            parts[3] = String.valueOf(lastPart);

            this.IPAddress = String.join(".", parts);
        }

        this.printer = printer;
    }

    public void setRouter(Router router) {

        if (ipRouter.isEmpty()) {
```

```
            this.IPAddress = "192.168.0.1";
        }else{
            // Розділяємо IP-адресу на окремі частини по крапках
            String[] parts = ipRouter.get(ipRouter.size() - 1).split("\\.");

            // Отримуємо останню частину IP-адреси і перетворюємо її в ціле
число
            // Збільшуємо останню частину на одиницю
            int lastPart = Integer.parseInt(parts[3]) + 1;

            parts[3] = String.valueOf(lastPart);

            this.IPAddress = String.join(".", parts);
        }

        this.router = router;
    }

    public <T> Optional<T> getDevice() {
        return Optional.ofNullable(commutator)
                .map(device -> (T) device)
                .or(() -> Optional.ofNullable(pc).map(device -> (T) device))
                .or(() -> Optional.ofNullable(printer).map(device -> (T)
device))
                .or(() -> Optional.ofNullable(router).map(device -> (T) device))
                .or(() -> Optional.ofNullable(server).map(device -> (T)
device));
    }

    public  void setConnectedLiness(ArrayList<Line> lines){
        connectedLines = lines;
    }

    public void setConnectionSettings(HashMap<String, String> settings) {
        // Встановлюємо значення IPv4
        setIPAddress(settings.get("IPAddress"));
        setSubnetMask(settings.get("SubnetMask"));
        setDefaultGateway(settings.get("DefaultGateway"));
        setDNSServer(settings.get("DNSServer"));

        // Встановлюємо значення IPv6
        setIPv6Address(settings.get("IPv6Address"));
        setLinkLocalAddress(settings.get("LinkLocalAddress"));
        setIPv6Gateway(settings.get("IPv6Gateway"));
        setIPv6DNSServer(settings.get("IPv6DNSServer"));
    }

    // Геттери
    public String getIPAddress() {
        return IPAddress;
    }

    public String getSubnetMask() {
        return SubnetMask;
    }

    public String getDefaultGateway() {
        return DefaultGateway;
    }

    public String getDNSServer() {
        return DNSServer;
    }

    public String getIPv6Address() {
        return IPv6Address;
    }

    public String getLinkLocalAddress() {
```

```
    return LinkLocalAddress;
}

public String getIPv6Gateway() {
    return IPv6Gateway;
}

public String getIPv6DNSServer() {
    return IPv6DNSServer;
}

public boolean getHTTPConnect(){
    return httpConnect;
}

public boolean getHTTPSConnect(){
    return httpsConnect;
}

// Сеттери
public void setIPAddress(String IPAddress) {
    this.IPAddress = IPAddress;
}

public void setSubnetMask(String SubnetMask) {
    this.SubnetMask = SubnetMask;
}

public void setDefaultGateway(String DefaultGateway) {
    this.DefaultGateway = DefaultGateway;
}

public void setDNSServer(String DNSServer) {
    this.DNSServer = DNSServer;
}

public void setIPv6Address(String IPv6Address) {
    this.IPv6Address = IPv6Address;
}

public void setLinkLocalAddress(String LinkLocalAddress) {
    this.LinkLocalAddress = LinkLocalAddress;
}

public void setIPv6Gateway(String IPv6Gateway) {
    this.IPv6Gateway = IPv6Gateway;
}

public void setIPv6DNSServer(String IPv6DNSServer) {
    this.IPv6DNSServer = IPv6DNSServer;
}

public void setHttpItemConnect(ImageInfo httpConnectItem) {
    this.httpConnectItem = httpConnectItem;
}

public ImageInfo getHttpItemConnect(){
    return httpConnectItem;
}

public void setHttpSelected(boolean selected) {
    this.httpConnect = selected;
}

public void setHttpsSelected(boolean selected) {
    this.httpsConnect = selected;
}

public String getProtocolsConnect(){
```

```
        if (httpsConnect) return "HTTPS";
        if (httpConnect) return "HTTP";
        else httpConnect = true;

        return "HTTP";
    }
}
```

//infoStage.css – стилі для діалового вікна

```
#infoStage {
    -fx-background-color: #0e79cf; /* Background color */
    -fx-font-size: 14px; /* Font size */
    -fx-padding: 10; /* Padding */
}

#typeLabel{
    -fx-color: #101010; /* Font color */
    -fx-font-size: 32px; /* Font size */
}
```

//style.css – допоміжні стилі

```
/* Style for the root container */
#root {
    -fx-background-color: #f0f0f0; /* Background color */
}

/* Style for the top menu bar */
#menuBar {
    -fx-background-color: #333333; /* Background color */
    -fx-text-fill: white; /* Text color */
}

/* Style for the container with images */
#imageContainer {
    /* Add your custom styles here */
}

/* Style for menu buttons */
.menu-button {
    -fx-background-color: #4CAF50; /* Background color */
    -fx-text-fill: white; /* Text color */
}

/* Style for context menu items */
.context-menu-item {
    -fx-background-color: #2196F3; /* Background color */
    -fx-text-fill: white; /* Text color */
}
```