МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему:

«Інформаційна технологія проєктування віддаленої телеком-лабораторії для побудови комп'ютерних мереж»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Великодний Д.В.**

Студента групи ІК.м-01 — **Перепелиця Б.В.**

СУМИ 2021

Сумський державний університет
(назва вузу)

Факультет *ЕлІТ* Кафедра *Комп'ютерних наук*
Спеціальність *«122 - Комп'ютерні науки»*

Затверджую:
зав.кафедрою _____
_____
"_____" _____ 20___р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТОВІ

Перепелиці Богдану Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Інформаційна технологія проєктування віддаленої телеком-лабораторії для побудови комп'ютерних мереж.*

затверджую наказом по інституту від "____" _____ 20 ___ р. № _____

2. Термін здачі студентом закінченого проекту (роботи) _____

3. Вхідні данні до проекту (роботи) _____
_____
_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми. Постановка задачі дослідження. 2) Огляд технологічних рішень Cisco з можливостей віддаленого налаштування комп'ютерних мереж. 3) Моделювання мережі в графічних симуляторах. 4) Розробка програмного забезпечення графічного інтерфейсу налаштування мережі віддалено із використанням протоколів віддаленого доступу*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) *2 – Актуальність теми, 3 – Мета роботи, 4 – Завдання роботи, 5 – Протоколи віддаленого доступу, 6 – SSH протокол, 7 – Octal cable, 8 - Розроблена схема в Cisco Packet Tracer, 9 - Список інтерфейсів роутера, 10 - Перевірка SSH підключення, 11 - Графічний інтерфейс адміністратора, 12 - Графічний інтерфейс користувача, 13 – Висновки*

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	*Аналіз проблеми. Постановка задачі дослідження*		
2.	*Огляд технологічних рішень Cisco з можливостей віддаленого налаштування комп'ютерних мереж*		
3.	*Моделювання мережі*		
4.	*Розробка програмного забезпечення графічного інтерфейсу налаштування мережі віддалено із використанням протоколів віддаленого доступу*		
5.	*Оформлення пояснювальної записки до дипломної роботи*		

Студент – дипломник _____
(підпис)

Керівник проекту _____
(підпис)

РЕФЕРАТ

Записка: 74 стор., 24 рис., 3 додатки, 12 джерел.

Об’єкт дослідження – віддалена телеком-лабораторія для побудови комп'ютерних мереж.

Предмет дослідження – особливості налаштування віддаленої мережі із використанням протоколів віддаленого доступу.

Мета роботи – розробка програмного забезпечення, графічний інтерфейс якого дозволить користувачам моделювати телекомунікаційні мережі та налаштовувати їх роботу у режимі онлайн.

Методи дослідження – моделювання схеми з використанням SSH протоколу в симуляторі мереж PacketTracer. Застосування інструментарію HTML, CSS та JavaScript для розробки web-орієнтованого графічного інтерфейсу.

Результати – розроблена web-орієнтована інформаційна система, графічний інтерфейс якої дозволяє вчителю будувати телекомунікаційні мережі у віддаленій телеком-лабораторії та надавати студентам можливість налаштовувати і тестувати їх без безпосередньої присутності, що покращує навчальний досвід студентів. Система надає можливість передивлятися історію змін на кожному з пристроїв та відслідкувати типові помилки чи неточності студентів, що полегшує роботу вчителя. Систему реалізовано у формі web-додатку з використанням мови програмування JavaScript та технології Web-socket

ПРОТОКОЛИ ВІДДАЛЕНОГО ДОСТУПУ, ТЕЛЕКОМ-ЛАБОРАТОРІЯ, OCTAL CABLE, WEB-ОРІЄНТОВАНА СИСТЕМА, SSH, SCSS, ANGULAR, JAVASCRIPT.

ЗМІСТ

ВСТУП

Із розвитком технологій життя людини стає простіше, завдання розв’язуються швидше та також підвищується цінність кожної хвилини. У наші дні в багатьох галузях з’явилася можливість використовувати онлайн ресурси та керувати усім зі свого смартфону або ноутбуку. Тим паче на розвиток «онлайн ери» вплинув короновірус, який змінив наше життя загалом, змусив заклади харчування, магазини, та схожі за напрямом заклади здійснювати доставку, а заклади освіти – перейти на дистанційне навчання.

За даними опитування Державної служби якості освіти, в оцінках учнів якості організації дистанційного навчання спостерігається впевнена позитивна динаміка. Зокрема, 70% учнів задоволені організацією дистанційного навчання. Порівняно з минулим роком цей показник зріс на 15%. Секрет такого підходу полягає у можливості створити гнучку і індивідуальну програму для кожного учня. У традиційній школі, на класних заняттях, таке неможливо. Навіть виконання домашніх робіт оцінюється і перевіряється досить уніфіковано. При дистанційній роботі з учнями вибудовується значно більш продуктивна комунікація: консультації з усіх виникаючих питань, допомога в опрацюванні складної теми, додаткова інформація за обраним напрямом, та інше.

Телекомунікаційні технології це один із таких предметів, де практика грає дуже важливу роль. Теоретична база також є важливою, але без лабораторних робіт неможливо досягти потрібного рівня навичок. Кожен пристрій має свої особливості, плюси та мінуси і тільки на практиці можна зрозуміти як з ним працювати. При розробці тренажера програміст може створити всі необхідні умови для повноцінного виконання практичних робіт, що, хоча і не повноцінно, але замінить заняття в університеті та покращить рівень освіченості студентів.

Мета випускної роботи – розробити тренажер, що дозволить користувачам моделювати телекомунікаційні мережі та налаштовувати їх роботу у режимі онлайн. Наразі, коли все навчання проходить у дистанційному режимі це буде корисним для покращення ефективності навчання, тому **робота є актуальною**.

1. ЛІТЕРАТУРНИЙ ОГЛЯД

1.1. Поняття телекомунікаційних систем

Поняття телекомунікаційної системи – це комплекс програмного та апаратного обладнання, з'єднаного між собою в ланцюг, який передає дані з однієї точки до іншої. Така передача даних можлива завдяки чіткій структурі телекомунікаційної мережі. Текстова, голосова, графічна, відео та аудіо інформація не складає повного списку можливостей телекомунікаційної системи [1].

Сьогодні це дуже поширена технологія, тому що ми живемо у часи, коли інформація, її передача та захист дуже важливі. Побудова інтелектуальної та безпечної мережі дозволяє користувачам не турбуватися про те, що їхні дані будуть втрачені або вкрадені шахраями.

У склад такої системи входять:

- мережеве обладнання, яке включає кінцеві пристрої (персональні комп'ютери, сервери, аудіо та відео пристрої, мережеві принтери, факс-апарати, зчитувачі штрих-кодів та ін) і комунікаційне обладнання (провідне, дротове та (або) бездротове середовище передачі даних), а також такі проміжні пристрої, як мережеві адаптери, модеми, повторювачі, мости, комутатори та ін.);
- засоби підтримки мережевого обладнання. У такій складній системі, якою є телекомунікаційна мережа, необхідно мати широкий арсенал програмного забезпечення, а також стандартні набори (стеки) комунікаційних протоколів, що визначають правила взаємодії мережевих пристроїв [2].

Найпопулярнішою в наш час є мережа TCP/IP, тобто Інтернет. Його унікальність полягає у величезному змісті інформаційних ресурсів та широкому використанні інформаційних технологій, таких як збирання, зберігання, обробка інформації та її подання у вигляді веб-сайтів. Тому Інтернет відомий як глобальна інформаційна мережа.

Крім того, в мережі Інтернет, технологічною особливістю якої є пакетний метод передачі інформації, можна організувати різні послуги, зокрема надати користувачам недорогі телекомунікаційні послуги, найбільш поширеною з яких є електронна пошта. Діяльність постачальників послуг зосереджена на організації так званих сервісних вузлів, за допомогою яких користувачі можуть отримувати доступ до різних мереж та інформаційних ресурсів цього вузла, так і віддалених вузлів Інтернету. При цьому постачальники послуг (провайдери) також є споживачами телекомунікаційних послуг (послуг з транспортування інформації), які надаються мережевими операторами.

Доступ користувачів до сервісного вузла, як правило, здійснюється через місцеві мережі операторів, однак окремі провайдери можуть мати власні мережі абонентського доступу. Прийнято розрізняти сервіс-провайдерів різних рівнів: місцевого рівня, регіонального рівня та національного рівня (рис. 1.1).

Вузол обслуговування провайдера на локальному рівні (рівень III) підключений через зовнішній канал, орендований оператором мережі, до так званої мережної точки доступу, де розташовані пристрої доступу до мережі регіонального провайдера.

Регіональний провайдер (Рівень II) зазвичай локалізує кілька точок доступу мережі у своєму регіоні, що дозволяє користувачам локальних провайдерів отримувати доступ як до своїх інформаційних ресурсів (їх вузлів обслуговування), так і зовнішніх інтернет-ресурсів.

Аналогічно регіональний провайдер, орендуючи канал у оператора мережі, підключається до мережі доступу національного провайдера (рівень I). І лише національний провайдер має право підключитися до точки доступу до мережі (NAP). NAP розшифровується як Міжнародні точки доступу до Інтернету[1].

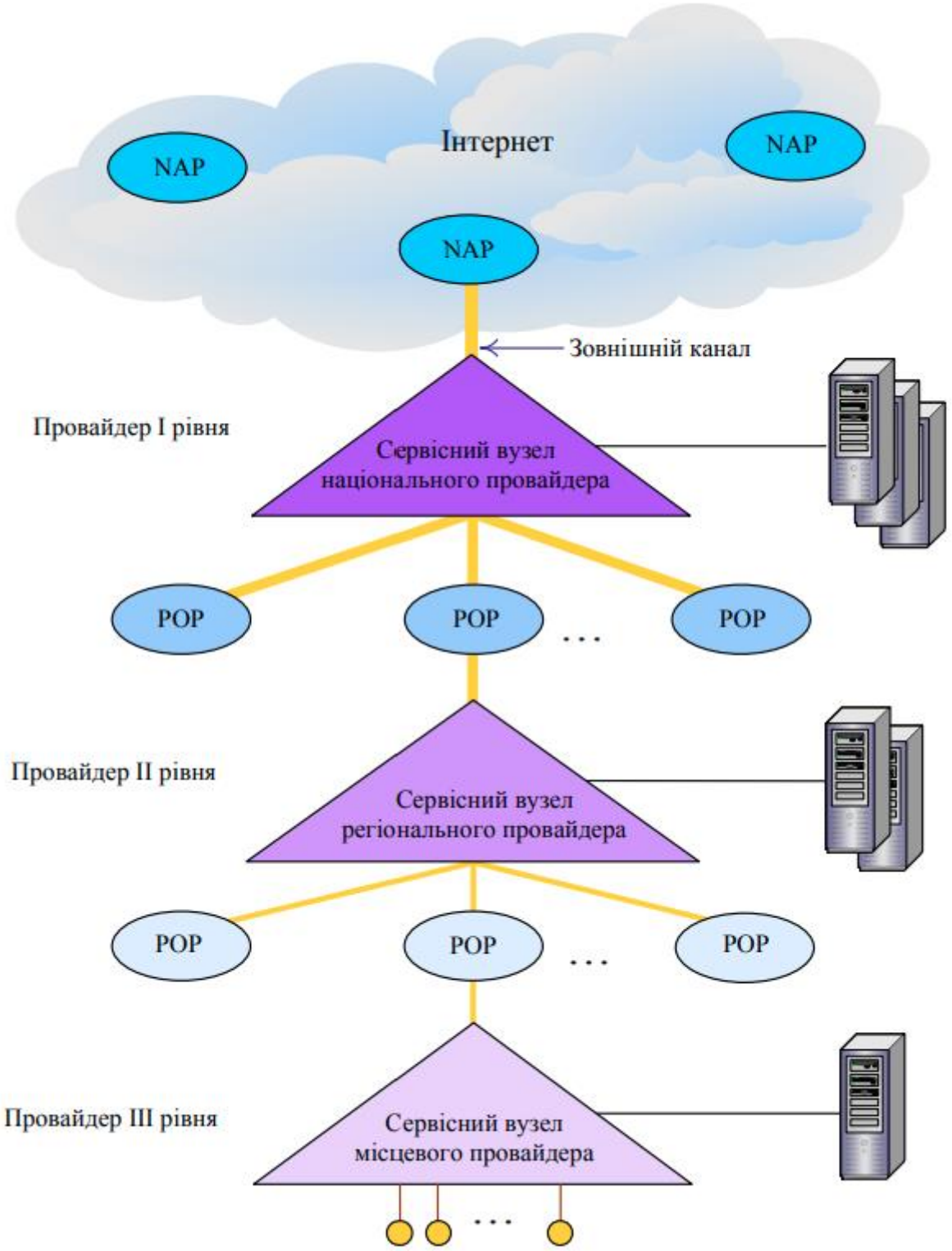

Рисунок 1.1 – Інтернет-сервіс-провайдинг

Корпоративні мережі або приватні мережі – це мережі, що належать установам та компаніям, чиї бізнес-інтереси виходять за межі ринку телекомунікацій.

Особливістю приватних мереж є те, що всі мережеві ресурси використовуються виключно співробітниками компанії, яка має мережу. Термін "приватна" мережа також відноситься до закритої мережі для конфіденційного зв'язку. Маючи це на увазі, термін "приватна мережа" найчастіше використовується для позначення мереж великих компаній з офісами у різних містах, країнах і навіть на континентах. Невеликі корпоративні мережі завжди вважаються приватними [3].

Об'єднуючи комп'ютери в мережу, компанія може оптимізувати свою інформаційну інфраструктуру (запуск програм, додатків, баз даних тощо. буд.), що, своєю чергою, підвищує ефективність всього процесу.

Залежно від розміру виробничої одиниці, в якій працює мережа, розрізняють мережі робочих груп, відомчі мережі, мережі університетського містечка та мережі компаній.

Мережі робочих груп зазвичай характеризуються невеликою кількістю робочих станцій (до 10) та використовуються невеликими групами співробітників компанії, що виконують загальне виробниче завдання. У цьому випадку метою мережі є поділ дорогого периферійного обладнання та даних, спільне використання програм та забезпечення універсального зв'язку для внутрішнього та зовнішнього зв'язку (рис. 1.2) [1].

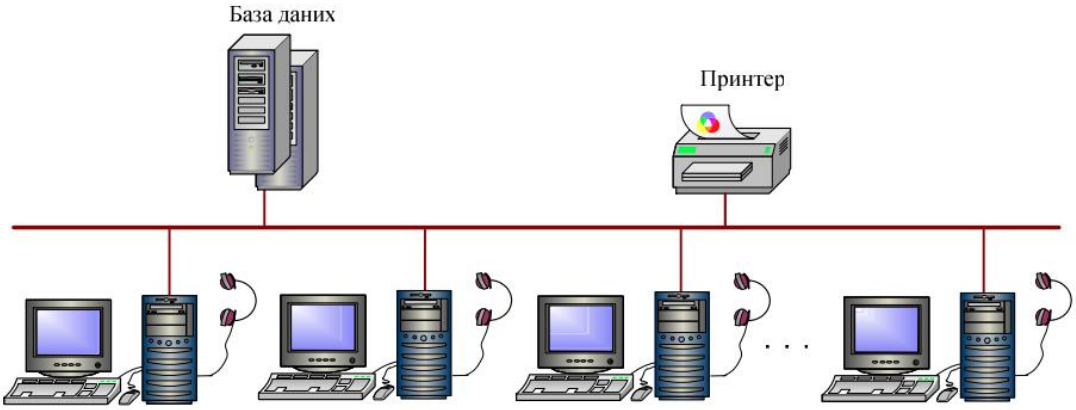

Рисунок 1.2 – Мережа робочої групи

Мережі відділів можуть об'єднувати від 30 до 100 робочих місць та розраховані на роботу працівників відділу. Ці співробітники зазвичай вирішують ряд пов'язаних завдань, наприклад, беруть участь у планово-фінансовій діяльності підприємства, ведуть облік матеріально-технічних цінностей тощо. буд. будинок. Мережа дозволяє працювати в режимі роздачі лазерним принтерам, модемам, інформаційним ресурсам та мережним програмам.

Інтеграція комп'ютера та телефону призвела до появи нових функцій, властивих сучасним відомчим мережам. На робочі місця співробітників додано спеціалізовані телефони, підключені до послідовних портів персональних комп'ютерів (ПК). Крім того, з'явилася можливість емулювати телефон за допомогою карт розширення стандартного інтерфейсу прикладного програмування телефону (TAPI). [4].

Завдяки новим стандартам факс, як важливий елемент ділового життя будь-якого офісу чи відділу, також був інтегрований у телефонну та комп'ютерну систему. У зв'язку з переходом на високошвидкісні технології з'явилася можливість підключення до мережі широкосмугового обладнання, що гарантує організацію відеоконференцзв'язку (рис. 1.3).

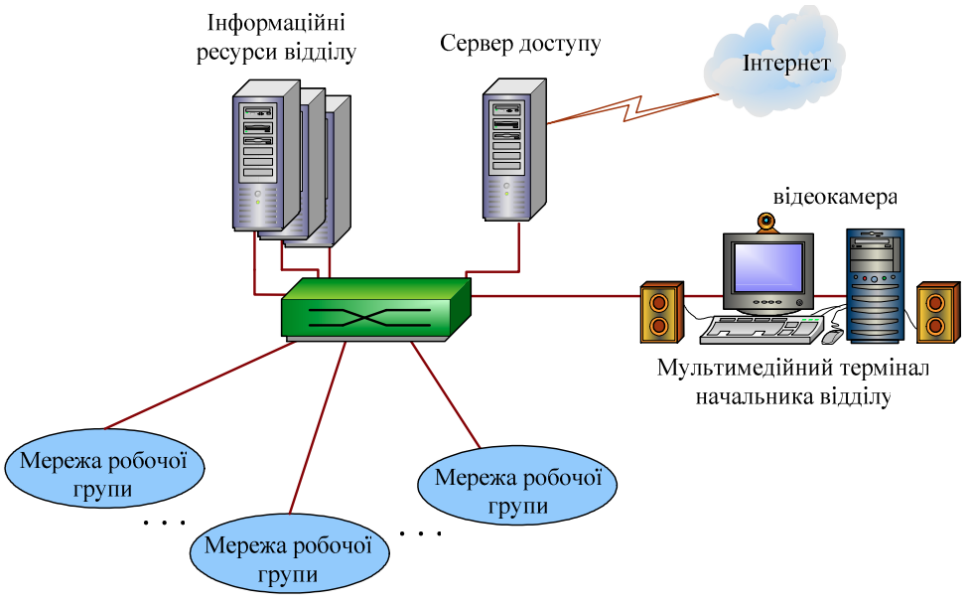

Рисунок 1.3 – Мережа відділу

Телекомунікаційні мережі характеризуються показниками, що в цілому відображають можливості та ефективність передачі інформації. Можливість передачі з телекомунікаційної мережі залежить від рівня її функціональності у часі, тощо.

Стан мережі пов'язаний з концепціями надійності та живучості. Відмінність між цими поняттями в основному пов'язана з різницею причин і факторів, що порушують нормальне функціонування мережі, та специфікою порушень [5].

Надійність мережі зв'язку характеризується здатністю забезпечувати зв'язок, зберігаючи у часі значення встановлених показників якості за умов експлуатації.

Вона відбиває вплив на працездатність мережі, переважно, внутрішніх чинників: випадкові відмови технічного устаткування, викликані процесами старіння, дефекти виробничих технологій чи помилки обслуговуючого персоналу. Показники надійності - це, наприклад, взаємозв'язок між часом безвідмовної роботи мережі та загальним часом її роботи, можливість бездоганного зв'язку тощо.

Важливим показником є кількість незалежних каналів зв'язку, які можна визначити між кількома точками мережі.

Надійність мережі зв'язку характеризується здатністю підтримувати повну або часткову функціональність під впливом деструктивних причин, що виникають поза мережею та призводять до відмови або значного пошкодження деяких її елементів (точок та ліній зв'язку). Є два типи причин: спонтанні та навмисні.

Надійність мережі можна охарактеризувати такими показниками, які визначають: ймовірність того, що обмежений обсяг інформації може бути переданий між даною парою мережевих точок після дії деструктивних факторів; мінімальна кількість точок, мережевої лінії (або обох), вихід з ладу яких призведе до порушення можливості підключення до мережі по відношенню до будь-якої пари точок; середня кількість точок, що залишаються підключеними при розриві кількох ліній зв'язку тощо [6].

Пропускна спроможність мережі. У випадках, коли мережа не може задовольнити (нести) необхідне навантаження, говоримо про обсяг навантаження, що виконується в мережі. Величина навантаження, що реалізується мережею, визначає її пропускну здатність і в деяких випадках може бути визначена кількісно апріорі. Наприклад, ви можете визначити значення максимального потоку інформації між двома точками (джерело-колектор) або смугу пропускання перетину мережі, яка є вузьким місцем під час поділу мережі між джерелом та колектором на дві частини. Оцінка пропускної спроможності мережі значною мірою пов'язана з параметрами якості обслуговування, оскільки реалізація заданого навантаження має виконуватись відповідно до заданих параметрів якості.

Якість обслуговування визначається набором характеристик, що вказують на рівень відповідності телекомунікаційної мережі чинним стандартам та вимогам користувачів [7].

Економічність та ціна. Мережа електрозв'язку є економічно вигідною, оскільки витрати на організацію та забезпечення операцій відшкодовуються за рахунок доходів від послуг, що надаються користувачам. Основними економічними характеристиками мережі є сукупні витрати (на рівні мережі), які визначають її вартість з урахуванням експлуатації та управління.

1.2. Протоколи віддаленого доступу

SSH ("Secure Shell") - це протокол віддаленого адміністрування, призначений для віддаленого керування операційними системами та тунелювання TCP-з'єднань. Використання цього протоколу дозволяє використовувати різні алгоритми шифрування, що дозволяє безпечно працювати практично в будь-якому незахищеному середовищі: робота з комп'ютером через командну оболонку, передача будь-якого типу даних (наприклад, відео та аудіо файлів) зашифрованим каналом [8].

Перша версія протоколу з’явилася в 1995 році, а вже в 1996 році була представлена його покращена версія, яка стала основою для подальшого розвитку продукту. Сьогодні є SSH-сервер та SSH-клієнт для всіх мережевих

операційних систем, а сам протокол SSH є одним із найпопулярніших рішень для віддаленого керування системою та передачі важливої інформації.

SSH – це протокол, який використовує модель клієнт-сервер для аутентифікації віддалених систем та забезпечує шифрування даних, якими обмінюються через віддалений доступ.

За замовчуванням протокол використовує порт TCP-22: на ньому сервер (хост) очікує вхідного з'єднання і після отримання команди та виконання автентифікації організує запуск клієнта, відкривши оболонку, вибрану користувачем. При необхідності користувач може змінити порт, що використовується [8].

Щоб створити з'єднання SSH, клієнт повинен ініціювати з'єднання з сервером, забезпечуючи безпечне з'єднання та підтверджуючи свою особу (ідентичність перевіряється за попередніми записами, що зберігаються у файлі RSA, та особистими даними користувача, необхідними для аутентифікації).

Використання SSH підключення має ряд переваг:

- безпечна робота на віддаленому ПК із використанням командної оболонки;
- використання різних алгоритмів шифрування (симетричного, асиметричного та хешування);
- можливість безпечного використання будь-якого мережевого протоколу, що дозволяє передавати захищеним каналом файли будь-якого розміру.

Для забезпечення доступу до SSH користувачу необхідні SSH-клієнт та SSH-сервер. Кожна операційна система має власний набір програм, які забезпечують підключення. Отже, для Linux це lsh (сервер та клієнт), openssh (сервер та клієнт). Для Mac OS часто використовується Nifty Telnet SSH. А в ОС Windows для реалізації підключення за протоколом SSH найчастіше використовується програма PuTTY. Також ініціювати SSH підключення можна за допомогою стандартної консолі Windows.

Для запуску протоколу SSH використовуються два основних методи аутентифікації: за паролем та ключем. У разі використання аутентифікації за

паролем користувач використовує особистий логін та пароль для встановлення з'єднання. Якщо використовується ключ, публічний (на пристрої, до якого вони підключатимуться) та приватний (на пристрої, з якого буде виконано з'єднання), ключі будуть попередньо згенеровані для кожного окремого користувача. Ці файли не передаються під час аутентифікації, система лише перевіряє, чи є у власника відкритого ключа закритий ключ [9].

SSH - один із найбезпечніших протоколів для реалізації віддаленого доступу до ПК. Сучасні алгоритми шифрування та широкий спектр інструментів налаштування протоколів роблять його найбільш популярним варіантом для віддаленого адміністрування комп'ютера та безпечної передачі даних.

Служба віддаленого управління **telnet** (Teletype Network) – одна з найстаріших мережевих технологій в Інтернеті (першою специфікацією був RFC 158 від 19.05.1971). Поточна специфікація Telnet - RFC 854/STD 8 - Telnet Protocol Specification. За промовчанням telnet-сервер приймає вхідні з'єднання через порт TCP 23 [10].

Протокол Telnet спочатку був розроблений для використання у гетерогенних мережах. Він заснований на концепції мережевого віртуального терміналу (NVT) – механізму абстракції спеціальних функцій введення/виведення різних апаратних та програмних платформ. Наприклад, UNIX використовує LF (код 13) як розрив рядка, тоді як MS-DOS та Windows використовують пару символів CR-LF (коди 10 та 13). Мережевий віртуальний термінал NVT пропонує єдиний набір символів, який використовується для перетворення клієнтських та серверних кодів [10].

Хоча сеанс telnet розрізняє клієнтську та серверну сторони, протокол є симетричним, і обидві сторони взаємодіють через NVT, обмінюючись даними двох типів:

- дані додатків (тобто дані, що передаються від користувача до серверної програми та навпаки);
- параметри протоколу Telnet, які служать для з'ясування можливостей та переваг сторін.

Дані програми проходять по протоколу без змін, тобто на виході другого віртуального терміналу ми бачимо те, що було подано на вхід першого. З погляду протоколу, ці дані є просто послідовність байтів, які за умовчанням належать набору ASCII. Якщо встановлено значення Binary, рядок будь-яких даних у двійковому поданні.

Telnet підтримує чотири режими передачі даних:

- Напівдуплексний режим. За умовчанням NVT - це напівдуплексний пристрій, який вимагає від сервера виконання спеціальної команди (GO AHEAD, GA) до того, як введення користувача буде прийнято. Введення користувача повертається локально з клавіатури NVT на принтер NVT, тому від клієнта на сервер відправляються тільки повні рядки.
- Посимвольний. Кожен введений символ надсилається на сервер окремо від інших. Сервер повертає (опція ECHO – RFC 857) більшість символів, і ці символи відображаються на екрані клієнта.
- Блочний. Всі дані відправляються єдиним монолітом.
- Лінійний режим (лінійний режим). У цьому випадку цей термін відноситься до фактичної опції лінійного режиму (RFC 1184). Цей варіант обговорюється між клієнтом та сервером та усуває будь-які недоліки в онлайн-режимі за один раз.

З точки зору безпеки протокол telnet уразливий для будь-якого типу атак, для яких уразливий його транспорт, тобто протокол TCP. Протокол не передбачає використання шифрування даних або автентифікації. Отже, telnet можна використовувати у повністю контрольованій мережі або з безпекою мережевого рівня (різні реалізації VPN). Через недостатню надійність telnet давно не використовують як інструмент управління операційною системою.

Іноді клієнти telnet використовують для доступу до інших символьних протоколів на основі транспорту TCP (HTTP, IRC, SMTP, POP3 та ін.). Однак використання клієнта telnet у цій якості може призвести до помилок, пов'язаних із неправильною інтерпретацією параметрів протоколу сервером.

Протокол **rlogin** (Remote login, RFC 1282, порт сервера - 513) дозволяє користувачам робочих станцій UNIX підключатися до віддалених серверів UNIX в Інтернеті і працювати так, як при підключенні терміналу безпосередньо до машини. Крім rlogin, існує безліч інших подібних протоколів: rsh (віддалена оболонка), rcp (віддалена копія). Утиліти rlogin, rsh та rcp часто називаються командами r [11].

Вперше представлений у 4.2BSD UNIX, rlogin (та інші команди r) колись були надзвичайно популярні у цьому середовищі. Як інструмент термінального доступу Rlogin дуже схожий на telnet, але через його тісну інтеграцію з операційною системою його використання в інших системах обмежено.

У Rlogin відсутні багато параметрів, специфічних для telnet, зокрема, режим узгодження між клієнтом і сервером. Однак rlogin забезпечує принаймні мінімальну безпеку, засновану на довірчих відносинах між хостами: на сервері rlogin у спеціальних системних файлах (зазвичай /etc/hosts.equiv та $HOME/.rhosts) адміністратор може перерахувати комп'ютери, доступ з яких до даного сервера дозволений лише після вводу паролю.

Користувачі інших комп'ютерів (не вказані в цих файлах) можуть підключитися до сервера лише після введення пароля (який, як і в telnet, передається у вигляді відкритого тексту). Але довірчі відносини між хостами також не є панацеєю і можуть бути встановлені лише в ізольованих мережах. Справа в тому, що методи несанкціонованого доступу, такі як заміна IP-адрес (IP-спуфінг) та доменних імен (DNS-спуфінг), роблять r-сервіси в Інтернеті незахищеними. Отже, сучасні UNIX-подібні дистрибутиви операційних систем не включають справжні команди r, а замінюють їх посиланнями на їх аналоги SSH: scp, sftp і т.д.

1.3. Октальний кабель

Октальний кабель використовується для підключення до консольних портів від маршрутизатора Cisco 2509 або 2511 до лабораторних маршрутизаторів Cisco CCNA, CCNP або CCIE. Вам знадобиться один вісімковий кабель для маршрутизатора 2509 і два вісімкові кабелі для

маршрутизатора 2511. Це позбавить від необхідності вручну переміщувати консольний кабель між маршрутизаторами у сертифікаційній лабораторії Cisco.

У кабелях CAB-OCTAL-ASYNC використовується 68-контактний роз'єм та розривний кабель, а на кожному 68-контактному роз'ємі є вісім асинхронних портів із згорнутим кабелем RJ-45. Ви можете підключити кожен асинхронний порт скрученого кабелю RJ-45 до консолі або порту Aux пристрою. Цей кабель можна використовувати для асинхронних модулів високої щільності NM-16A або NM-32A, доступних для маршрутизаторів серій 2600 і 3600.

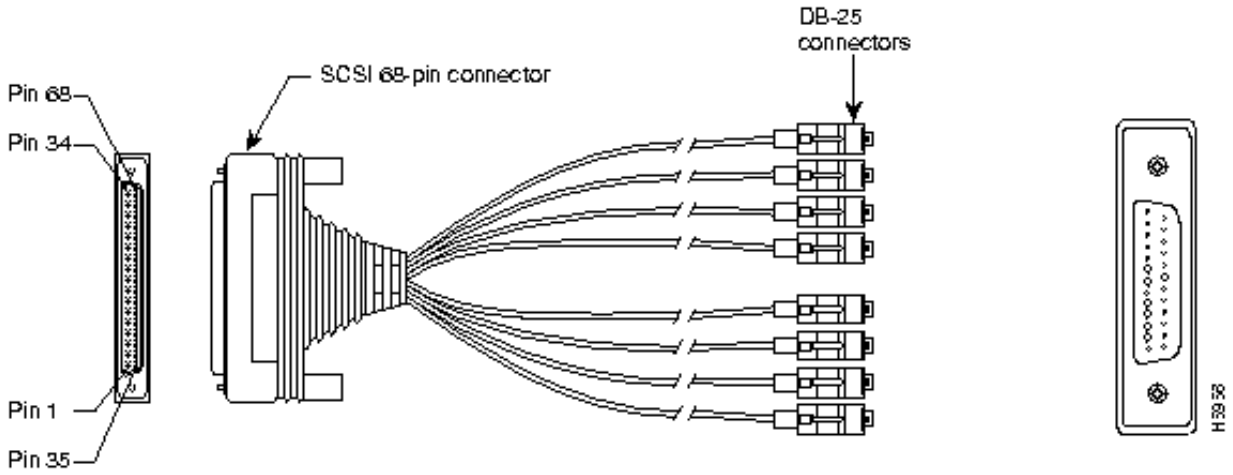

Рисунок 1.4 – Октальний кабель

Асинхронні порти від 68-контактного гнізда є пристроями кінцевого обладнання даних (DTE). Від DTE до пристроїв DTE потрібен згорнутий (нуль-модемний) кабель. Від DTE до пристроїв кінцевого обладнання ланцюга даних (DCE) потрібний прямий кабель. Оскільки кабель CAB-OCTAL-ASYNC сам згорнутий, можна підключити кожен кабель безпосередньо до консольних портів пристроїв з інтерфейсами RJ-45. Однак, якщо консольний порт пристрою, до якого ви підключаєтеся, є 25-контактним інтерфейсом (DCE), використовуйте адаптер RJ-45 на 25-контактний (номер продукту CAB-5MODCM=) з позначкою «Modem» (щоб перевернути "roll") для завершення з'єднання. Якщо потрібно збільшити радіус дії кабелю CAB-OCTAL-ASYNC, можна використати прямий кабель RJ-45 як подовжувач (рис 1.4) [12].

1.4. Постановка задачі

Проаналізувавши літературні дані, мету кваліфікаційної магістерської роботи можна сформулювати наступним чином: необхідно розробити веб-орієнтовану інформаційну систему, графічний інтерфейс якої буде дозволяти налаштовувати пристрої, що знаходяться у дистанційній телеком лабораторії, використовуючи SSH протокол. Студенти повинні мати змогу працювати одночасно та з декількома пристроями у режимі реального часу.

Для того щоб мати можливість правильного налаштування також знадобиться настільний додаток, встановлений на ПК, приєднаному для консольного сервера, що буде отримувати команди від клієнтської частини та виконувати налаштування приладів, використовуючи підключення за SSH протоколом.

Потрібно забезпечити оновлення даних та налаштувань для всіх користувачів одночасно, тому мі будемо використовувати технологію Web socket для миттєвого обміну повідомленнями і оновлення налаштувань при появі будь яких змін.

Інтерфейс має бути простим та інтуітивно зрозумілим, щоб студенти початкових курсів, які не мають навичок роботи з подібним програмним забезпеченням, могли з ним працювати.

Постановка задачі:

1. Конфігурація мережі на базі обладнання Cisco та симулятора Packet Tracer.
2. Розробка інтерфейсу налаштування приладів із використанням SSH протоколу.
3. Розробка адміністративної панелі для викладача.
4. Розробка настільного додатку для ПК, підключеного до консольного серверу.

2. МОДЕЛЮВАННЯ МЕРЕЖІ З КОНСОЛЬНИМ СЕРВЕРОМ ТА SSH ПРОТОКОЛОМ ЗА ДОПОМОГОЮ СИМУЛЯТОРА PACKET TRACER ТА ОБЛАДНАННЯ CISCO

2.1. Конфігурація мережі з використанням симулятора Packet Tracer.

Packet Tracer – це кросплатформний інструмент візуального моделювання, розроблений Cisco Systems, який дозволяє користувачам створювати мережеві топології та моделювати сучасні комп'ютерні мережі. Програмне забезпечення дозволяє користувачам моделювати конфігурацію маршрутизаторів та комутаторів Cisco за допомогою змодельованого інтерфейсу командного рядка. Packet Tracer використовує інтерфейс користувача з перетягуванням, який дозволяє користувачам додавати і видаляти змодельовані мережні пристрої на свій розсуд. Програмне забезпечення в першу чергу призначене для студентів, сертифікованих Cisco Network Associate Academy, як освітній інструмент, який допомагає вивчити фундаментальні концепції CCNA.

Packet Tracer також може працювати в Linux, Microsoft Windows та MacOS. Також доступні аналогічні програми для Android та iOS. Packet Tracer дозволяє користувачам створювати топології мережі, що моделюються, шляхом перетягування маршрутизаторів, комутаторів і різних інших типів мережевих пристроїв. Фізичне з'єднання між пристроями представлене кабелем. Packet Tracer підтримує безліч змодельованих протоколів рівня додатків, а також базову маршрутизацію з OSPF, EIGRP, BGP для обсягів, які потрібні поточною навчальною програмою CCNA.

Крім моделювання певних аспектів комп'ютерних мереж, Packet Tracer також можна використовувати для спільної роботи. Починаючи з Packet Tracer 5.0, Packet Tracer підтримує розраховану на багато користувачів систему, яка дозволяє кільком користувачам з'єднувати кілька топологій разом через комп'ютерну мережу. Packet Tracer також дозволяє вчителям створювати завдання для студентів. Packet Tracer часто використовується в навчальних

закладах як навчальний посібник. Cisco Systems стверджує, що Packet Tracer є корисним для мережевих експериментів.

Симулятор реалізує серії маршрутизаторів Cisco 800, 1800, 1900, 2600, 2800, 2900 і комутатори Cisco Catalyst 2950, 2960, 3560, а також міжмережевий екран ASA 5505. Крім того, існують сервери DHCP, HTTP, TFTP, FTP, DNS, AAA, SYSLOG, NTP та EMAIL, робочі станції, різні модулі для комп'ютерів та маршрутизаторів, IP-телефони, смартфони, концентратори, а також хмара, що імітує WAN. Ви можете підключати мережні пристрої за допомогою різних типів кабелів, наприклад, прямі та зворотні патч-корди, оптичні та коаксіальні кабелі, послідовні кабелі та телефонні пари. Також можна отримати доступ до реальної мережі і використовувати віртуальні машини VirtualBox.

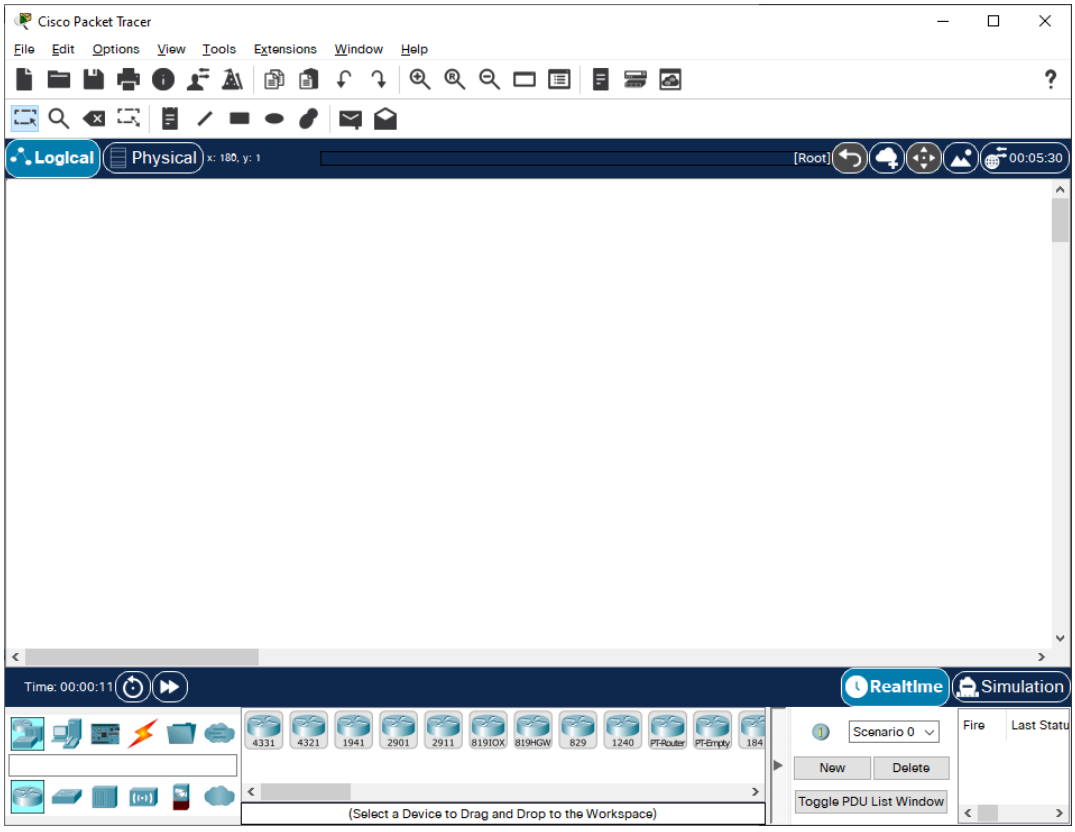

Рисунок 2.1 – Початок роботи з Packet tracer

Прилади для моделювання в Cisco Packet Tracer, які знадобляться для конфігурації мережі:

- кінцеві пристрої (робочі станції, ноутбуки, сервери);
- маршрутизатори;

- комутатори.

Для початку нам потрібно побудувати мережу, яка задовольнить наші умови:

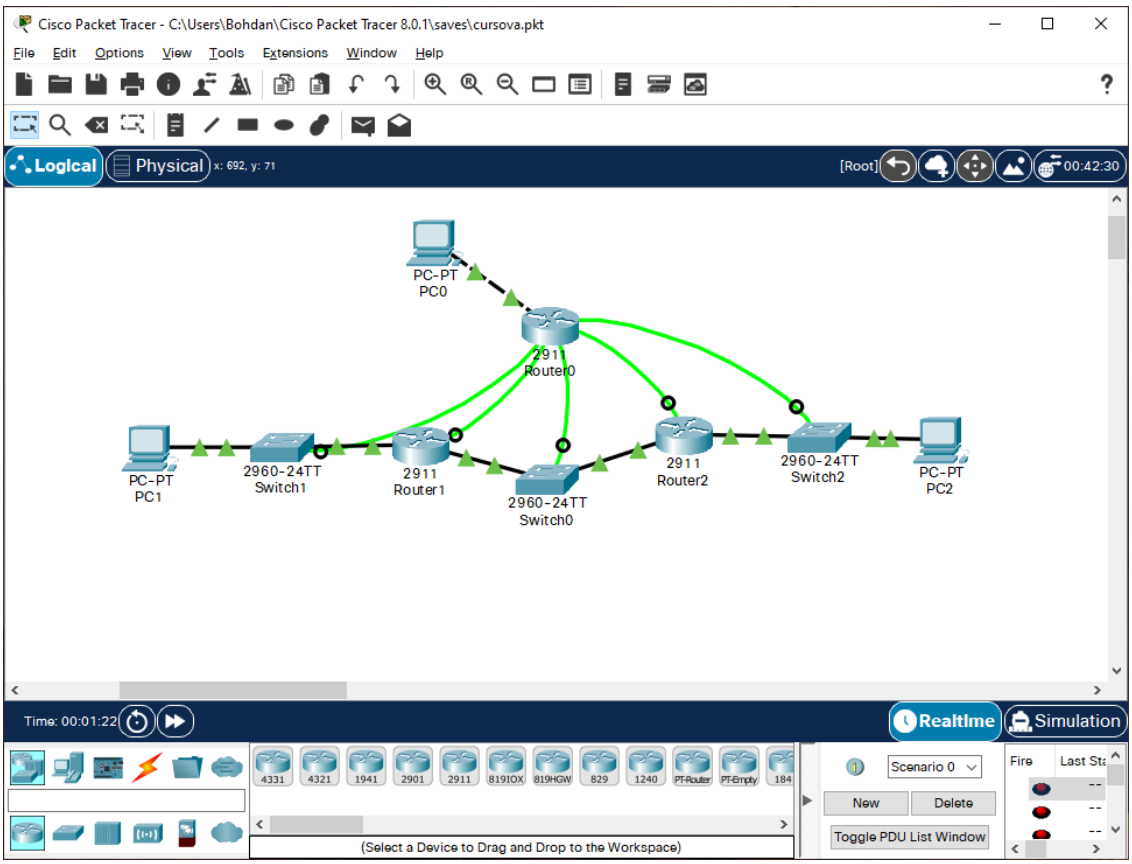

Рисунок 2.2 – Мережа із використанням консольного сервера

Router0 є нашим сервер роутером, за допомогою якого буде проходити налаштування всіх інших пристроїв, що підключені до нього через консольні порти. Ми використовуємо HWIC-8A модуль, що дає нам можливість мати до восьми підключених через консольний порт пристроїв.

Після побудови мережі нам потрібно налаштувати SSH підключення між консольним сервером та PC0. Для цього ми виконуємо команди:

Router>en

Router#conf t

Задаємо ім’я роутеру, для того щоб після SSH підключення нам легше було відрізняти роутери між собою:

Router(config)#hostname SERVER

Конфигурація порта gi 0/0, до якого приєднаний PC0:

SERVER(config)#int fa 0/0

SERVER(config-if)#ip add 192.168.0.1 255.255.255.0

SERVER(config-if)#no sh

SERVER(config-if)#ex

Задаємо домен:

SERVER(config)#ip domain-name ciscolab.ua

Вмикаємо SSH підключення та генеруємо RSA ключі:

SERVER(config)#ip ssh version 2

SERVER(config)#crypto key generate rsa

Створюємо користувача admin с паролем cisco і максимальним рівнем привілегій 15.

SERVER(config)#username dima privilege 15 password cisco

```
Router>
Router>
Router>
Router>
Router>en
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#hostname SERVER
SERVER(config)#int gi 0/0
SERVER(config-if)#ip add 192.168.0.1 255.255.255.0
SERVER(config-if)#no sh
SERVER(config-if)#ex
SERVER(config)#ip domain-name ciscolab.ua
SERVER(config)#ip ssh version 2
SERVER(config)#crypto key generate rsa
% You already have RSA keys defined named SERVER.bohdan.ua .
% Do you really want to replace them? [yes/no]: 1024
% Please answer 'yes' or 'no'.
% Do you really want to replace them? [yes/no]: yes
The name for the keys will be: SERVER.ciscolab.ua
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 1024
% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]

SERVER(config)#username admin privilege 15 password cisco
*Mar 1 0:14:6.336: %SSH-5-ENABLED: SSH 2 has been enabled
SERVER(config)#ex
SERVER#
%SYS-5-CONFIG_I: Configured from console by console

SERVER#
SERVER#
```

Рисунок 2.3 – Налаштування серверного роутера

Далі нам потрібно знайти інтерфейси, до яких підключені пристрої, які ми будемо налаштовувати та включити для них transport input all. Для цього ми використовуємо команду sh line:

```
SERVER#
SERVER#sh line
   Tty Line Typ     Tx/Rx    A Roty AccO AccI   Uses   Noise  Overruns   Int
*    0    0 CTY              -    -    -    -      0       0     0/0       -
     1    1 AUX    9600/9600 -    -    -    -      0       0     0/0       -
 0/3/0   51 TTY    9600/9600 -    -    -    -      0       0     0/0       -
 0/3/1   52 TTY    9600/9600 -    -    -    -      0       0     0/0       -
 0/3/2   53 TTY    9600/9600 -    -    -    -      0       0     0/0       -
 0/3/3   54 TTY    9600/9600 -    -    -    -      0       0     0/0       -
 0/3/4   55 TTY    9600/9600 -    -    -    -      0       0     0/0       -
 0/3/5   56 TTY    9600/9600 -    -    -    -      0       0     0/0       -
 0/3/6   57 TTY    9600/9600 -    -    -    -      0       0     0/0       -
 0/3/7   58 TTY    9600/9600 -    -    -    -      0       0     0/0       -
   388  388 VTY              -    -    -    -      0       0     0/0       -
   389  389 VTY              -    -    -    -      0       0     0/0       -
   390  390 VTY              -    -    -    -      0       0     0/0       -
   391  391 VTY              -    -    -    -      0       0     0/0       -
   392  392 VTY              -    -    -    -      0       0     0/0       -
Line(s) not in async mode -or- with no hardware support:
3-50, 59-387
SERVER#
```

Рисунок 2.4 – Список інтерфейсів роутера

У цьому списку ми бачимо TTY інтерфейси, які нам потрібно налаштувати. Ми маємо однакові налаштування для кожного з них, тож використаємо такі команди:

SERVER#conf t
SERVER(config)#line 0/3/0 0/3/7
SERVER(config-line)#transport input all
SERVER(config-line)#login local

Тепер ми можемо перевірити наше SSH з’єднання між комп’ютером та роутером і налаштувати мережу. Для цього ми відкриваємо PC0, вкладку Desktop та обираємо Command Prompt. Для SSH з’єднання використовується команда: ssh – l username:line 192.168.0.1, де у нашому випадку username це admin, а line – це номер лінії, до якої приєднаний пристрій (рис 2.4). І після цього вводимо пароль до користувача admin (рис 2.5):

```
C:\>ssh -l admin:52 192.168.0.1

Password:


System Bootstrap, Version 15.1(4)M4, RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 2010 by cisco Systems, Inc.
Total memory size = 512 MB - On-board = 512 MB, DIMM0 = 0 MB
CISCO2911/K9 platform with 524288 Kbytes of main memory
Main memory is configured to 72/-1(On-board/DIMM0) bit mode with ECC disabled

Readonly ROMMON initialized

program load complete, entry point: 0x80803000, size: 0x1b340
program load complete, entry point: 0x80803000, size: 0x1b340

IOS Image Load Test
___________________
Digitally Signed Release Software
program load complete, entry point: 0x81000000, size: 0x3bcd3d8
Self decompressing the image :
######################### [OK]
Smart Init is enabled
smart init is sizing iomem
                  TYPE      MEMORY_REQ
     Onboard devices &
          buffer pools      0x022F6000
------------------------------------------------
                TOTAL:      0x022F6000
Rounded IOMEM up to: 36Mb.
Using 6 percent iomem. [36Mb/512Mb]

             Restricted Rights Legend

Use, duplication, or disclosure by the Government is
subject to restrictions as set forth in subparagraph
```

Рисунок 2.5 – SSH підключення через консольний роутер

Таким чином ми можемо підключитися і налаштувати кожен пристрій окремо. Після налаштування ми, за допомогою відправки повідомлення між двома ПК ми бачимо що мережа налаштована вірно (рисунок 2.6):

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC1	PC2	ICMP		0.000	N	0	(edit)	(delete)

Рисунок 2.6 – Перевірка правильності налаштування мережі

2.2. Конфігурація мережі з використанням обладнання Cisco.

Cisco Systems, Inc. (NASDAQ: CSCO) – американська транснаціональна корпорація. Вона є найбільшим виробником мережевого обладнання у світі, призначеного для обслуговування мереж віддаленого доступу, служб безпеки, мережі зберігання даних, маршрутизації та комутації, а також для потреб комерційного ринку IP-комунікацій та корпоративного ринку.

Cisco виробляє велику кількість різних пристроїв:

- Ethernet комутатори;
- Cisco Nexus — комутатори дата-центрів;
- маршрутизатори;
- продукти для IP-телефонії, такі як IP PBX, VoIP-шлюзи;
- пристрої мережної безпеки (міжмережні екрани, VPN, IDS тощо);
- Wi-Fi точки доступу;
- платформи оптичної комутації;
- ATM-комутатори;
- кабельні модеми;
- DSL-устаткування;
- універсальні шлюзи і шлюзи віддаленого доступу;
- комутатори мереж зберігання даних (SAN, Storage Area Network);
- програмне забезпечення управління мережею.

За допомогою роутерів Cisco 2911 та комутаторів було змодельовано реальну комп’ютерну мережу Ethernet, налаштовано SERVER роутер та SSH з’єднання із комп’ютером. Використовуючи ПК, підключений до SERVER роутеру за допомогою SSH протоколу були налаштовані усі інші пристрої.

Тестування правильності налаштування було перевірено за допомогою команди ping між ПК, підключеними до різних мереж. На даний експеримент було витрачено 5 годин.

2.3. Розробка інтерфейсу налаштування приладів із використанням SSH протоколу.

Розробка графічного інтерфейсу була виконана за допомогою мови програмування TypeScript та фреймворку Angular.

TypeScript – це мова програмування, представлена Microsoft в 2012 році є інструментом розробки веб-додатків, що розширює JavaScript. TypeScript зворотньо сумісний із JavaScript і скомпільований на JavaScript. Після компіляції програми TypeScript її можна запускати у будь-якому сучасному браузері або використовувати із серверною інфраструктурою Node.js. Код експериментального компілятора, який перекладає TypeScript JavaScript, знаходиться під ліцензією Apache. Розробка відбувається у громадському репозиторії через сервіс GitHub.

TypeScript відрізняється від JavaScript своєю здатністю явно надавати статичні типи, підтримувати використання повних класів (як у традиційних об'єктно-орієнтованих мовах) і модулів, що підключаються, призначених для прискорення розробки за рахунок спрощення читання і рефакторингу, а також для повторного використання коду і допомоги в пошуку помилок на етапі розробки та компіляції для прискорення виконання програми.

Angular (версія 2 і новіша) – це безкоштовна платформа для розробки веб-сайтів з відкритим вихідним кодом, написана на TypeScript. Angular – це повністю переписаний фреймворк тією самою командою, яка написала AngularJS.

Спочатку створювалася як інша версія AngularJS. Angular 2 був переписаний з нуля на TypeScript, має іншу архітектуру і несумісний із попередніми версіями AngularJS. Щоб уникнути плутанини, було вирішено розробити його як окремий фреймворк з нумерацією версій, починаючи з двох.

Платформа працює з HTML, який включає додаткові атрибути, описані в директивах, і пов'язує введення або виведення області сторінки із шаблоном, який є звичайною змінною JavaScript. Значення цих змінних встановлюються вручну або витягуються зі статичних або динамічних даних JSON. Angular слідує шаблону проектування MVC і заохочує вільне узгодження між поданням,

даними та логікою компонентів (рисунок 2.7). Використовуючи залежність, Angular надає класичні серверні служби, такі як контролери клієнта, що залежать від представлення:

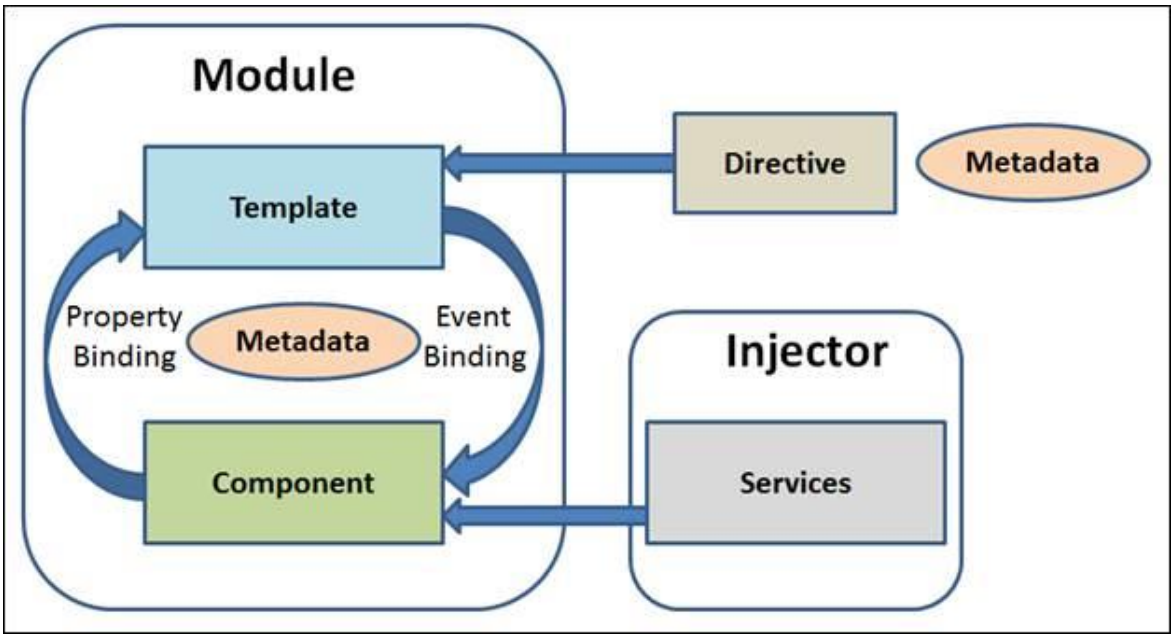

Рисунок 2.7 – Архітектура Angular

Ми також використовуватимемо технологію **Web-Socket** для оновлення даних у реальному часі. Веб-сокети – це передова технологія, яка створює інтерактивне з'єднання між клієнтом (браузером) та сервером для обміну повідомленнями у реальному часі. Веб-сокети, на відміну від HTTP, допускають двосторонній потік даних, що робить цю технологію унікальною.

Веб-сокети також можуть шифрувати передані дані; для цього використовується надбудова протоколу – WSS. Спеціальні доповнення до протоколів передачі даних кодують інформацію на стороні відправника та декодують на стороні одержувача, залишаючи її зашифрованою для будь-яких посередників. Це забезпечує безпечний рівень транспорту.

2.4. Розробка адміністративної панелі для викладача.

Для розробки адміністративної панелі було використано мову програмування Python та фреймворк Django. База даних – PostgreSQL.

Python - це мова програмування загального призначення високого рівня, орієнтована підвищення продуктивності праці розробників і спрощення коду. Базовий синтаксис Python мінімалістичний, але в той же час стандартна бібліотека містить багато корисних функцій.

Python підтримує безліч парадигм програмування, включаючи структуроване, об'єктно-орієнтоване, функціональне, імперативне та аспектно-орієнтоване. Він також включає такі функції, як підтримка багатопоточності, автоматичне керування пам'яттю, динамічна типізація, обробка винятків, високорівневі структури даних. Підтримується розділення програм на модулі, які можна об'єднувати в пакети.

Реалізація тестів Python - це інтерпретатор CPython, який підтримує найчастіше використовувані платформи. Він розповсюджується під безкоштовною ліцензією Python Software Foundation, що дозволяє використовувати його без обмежень у будь-яких програмах, включаючи пропрієтарні. Існує реалізація інтерпретатора для JVM із можливістю компіляції, CLR, LLVM та інших незалежних реалізацій. У проекті PyPy використовується JIT-компіляція, що значно збільшує швидкість виконання програм Python.

Django - це безкоштовне середовище веб-застосунків Python, засноване на шаблоні проектування MVC. Проект підтримується програмою Django Software Foundation.

Веб-сайт Django складається з однієї або кількох програм, які ми рекомендуємо ізолювати. Це одна з основних архітектурних відмінностей цього фреймворку багатьох інших (наприклад, Ruby on Rails). Фреймворк заснований на принципі DRY (do not repeat), тобто максимально повторне використання кодової бази. Крім того, на відміну від інших фреймворків, обробники URL-адрес Django можна явно налаштувати за допомогою регулярних виразів.

Для роботи з базою даних Django використовує свій ORM, в якому модель даних описується класами Python, а з них генерується схема бази даних. Веб-фреймворк Django використовується великими і відомими веб-сайтами, такими як Instagram, Disqus, Mozilla, Washington Times, Pinterest, YouTube, Google і т.д.

та побудови діаграм, FreeNAS, безкоштовна реалізація системи зберігання та обміну файлами.

Архітектура Django схожа на модель-вистава-контролер (MVC). Класичний контролер MVC - це практично той самий рівень, який Django називає уявленням, а логіка уявлення реалізована у Django з використанням рівня шаблону. Ось чому архітектуру Django часто називають моделлю-шаблоном-виставою (MTV).

Початковий дизайн Django як блогу і ресурсу новин зробив великий вплив на його архітектуру: він надає ряд інструментів, які допоможуть вам швидко створювати новинні сайти. Наприклад, розробнику не потрібно створювати контролери та сторінки для адміністративної області веб-сайту, тому що Django має вбудоване керування контентом, яке можна вбудувати у будь-який із ваших веб-сайтів. Помічник Django може керувати кількома сайтами одночасно на тому ж самому сервері.

Модуль адміністрування дозволяє керувати будь-яким об'єктом контенту сайту та фіксує всі вжиті дії, щоб уникнути непередбачених ситуацій та надає інтерфейс для адміністрування користувачів та груп (з призначенням дозволів для окремих об'єктів).

PostgreSQL – це безкоштовна система управління об'єктно-реляційними базами даних (СУБД). Він існує в реалізаціях багатьох UNIX-подібних платформ, включаючи AIX, різні системи BSD, HP-UX, IRIX, Linux, macOS, Solaris / OpenSolaris, Tru64, QNX, а також Microsoft Windows.

Плюсами PostgreSQL є:

- високопродуктивні і надійні механізми транзакцій і реплікації;
- розширювана система вбудованих мов програмування: в стандартному постачанні підтримуються PL / pgSQL, PL / Perl, PL / Python і PL / Tcl; додатково можна використовувати PL / Java, PL / PHP, PL / Py, PL / R, PL / Ruby, PL / Scheme, PL / sh і PL / V8, а також є підтримка завантаження модулів розширення на мові C;
- наслідування;

- можливість індексування геометричних об'єктів і наявність базується на ній розширення PostGIS;
- вбудована підтримка слабоструктурованих даних в форматі JSON з можливістю їх індексації;
- розширюваність (можливість створювати нові типи даних, типи індексів, мови програмування, модулі розширення, підключати будь-які зовнішні джерела даних).

2.5. Розробка настільного додатку для ПК, підключеного до консольного серверу

Для безпечної взаємодії з SERVER роутером ми будемо використовувати настільний додаток, що буде отримувати команди із конфігурацією від нашого сервера на виконувати їх на обладнанні, використовуючи SSH протокол. Також буде вестися лог команд, що допоможе вчителю відслідковувати стан виконання роботи. Для розробки додатку буде використано ElectronJS та NodeJS.

Electron (раніше відомий як атомна оболонка) – це фреймворк, розроблений GitHub. Він дозволяє створювати власні графічні програми для комп'ютерних операційних систем з використанням Інтернет-технологій. Фреймворк включає Node.js для роботи з серверною частиною та бібліотеку рендерингу Chromium.

На базі Electron побудований не тільки текстовий редактор для програмістів Atom, але й таке програмне забезпечення для програмістів, як Visual Studio Code, Light Table (з версії 0.8), Ionic Lab, Avocode, консоль Mancy REPL для Node, js та Meteor js, Mongotron - менеджер графічного інтерфейсу для MongoDB. Крім того, на основі цього фреймворку написані клієнт чату Slack, Skype, Discord, настільний клієнт WordPress та багато інших.

Node або Node.js - це програмна платформа, заснована на двигуні V8 (перекладає JavaScript в машинний код), який перетворює JavaScript з вузькоспеціалізованої мови на мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виводу через свій API (написаний на C++), підключаючи інші зовнішні бібліотеки, написані різними

мовами, забезпечуючи їх виклики з коду JavaScript. Node.js в основному використовується на сервері, виступаючи як веб-сервер, але його можна розробляти в Node.js та віконних додатках робочого столу (використовуючи NW.js, AppJS або Electron для Linux, Windows та MacOS) і навіть програмних мікроконтролерів. (наприклад, тессель та спруїно). В основі Node.js лежить подієво-орієнтоване та асинхронне (або реактивне) програмування з неблокуючим введенням/виведенням.

Взаємодія між процесами (англ. Inter-Process Communication, скорочено англ. IPC) – це набір засобів, який дозволяє обмінюватися повідомленнями між процесами. Для взаємодії процесів, які виконуються на одному комп'ютері (під керуванням однієї операційної системи) використовують (взаємодія забезпечується ядром операційної системи, в якій виконуються процеси):

- сигнали – асинхронні повідомлення, які сповіщають про подію (надзвичайну ситуацію), на яку процес має відреагувати виконанням наперед визначеною функцією або командою, тощо;
- неіменовані й іменовані канали для передачі даних у вигляді синхронних (очікуваних) повідомлень; відправка повідомлення відбувається аналогічно запису в файл, отримання – як читання даних з файлу, якщо канал порожній – процес, який чекає на дані, призупиняється до тих пір, поки дані не надійдуть;
- черги повідомлень – пакети даних, що передаються між процесами та доводять до відома отримувача інформацію про надходження пакету;
- сегменти подільної пам'яті – засіб, що дозволяє кільком процесам сумісно використовувати (поділяти) фрагмент оперативної пам'яті з метою обміну даними; відправлення даних відбувається шляхом запису в пам'ять, отримання – читанням з пам'яті.

3. ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ГРАФІЧНОГО ІНТЕРФЕЙСУ ВІДДАЛЕНОЇ ТЕЛЕКОМ-ЛАБОРАТОРІЇ

3.1. Проектування бази даних.

У проекті буде використано PostgreSQL. ER-діаграма бази даних виглядає наступним чином (рис. 3.1.):

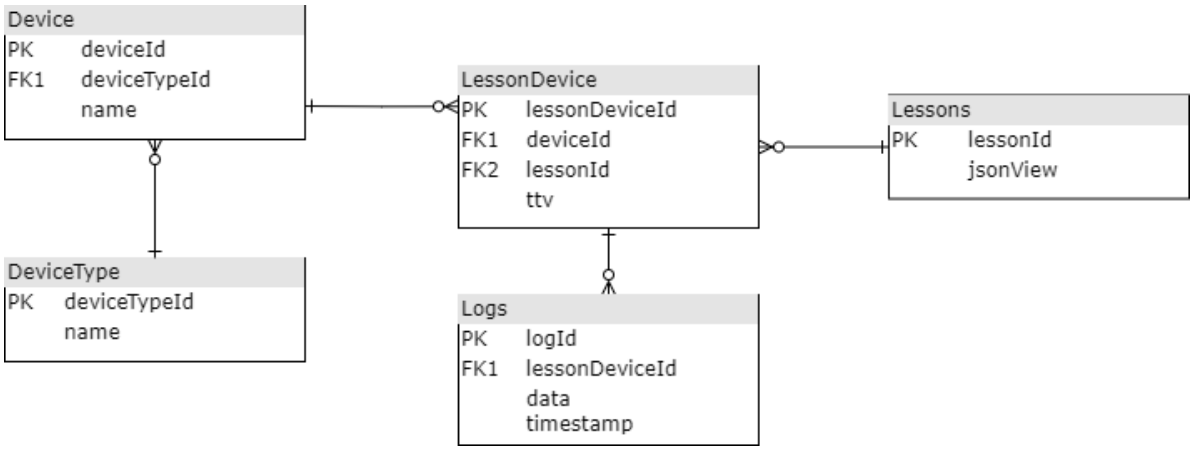

Рисунок 3.1 – База даних

Через те що було використано Python + Django, то база даних створена у вигляді ORM.

Для підвищення відмовостійкості та безпечності ми використовуємо таблицю Logs. Таким чином вчитель у будь-який час зможе відновити лабораторну роботу, використовуючи збережені дані та відслідкувати зміни, зроблені учнями.

3.2. Панель адміністрування.

У проекті використовується панель адміністрування Django. Сторінка авторизації панелі адміністрування виглядає наступним чином (рис. 3.2):

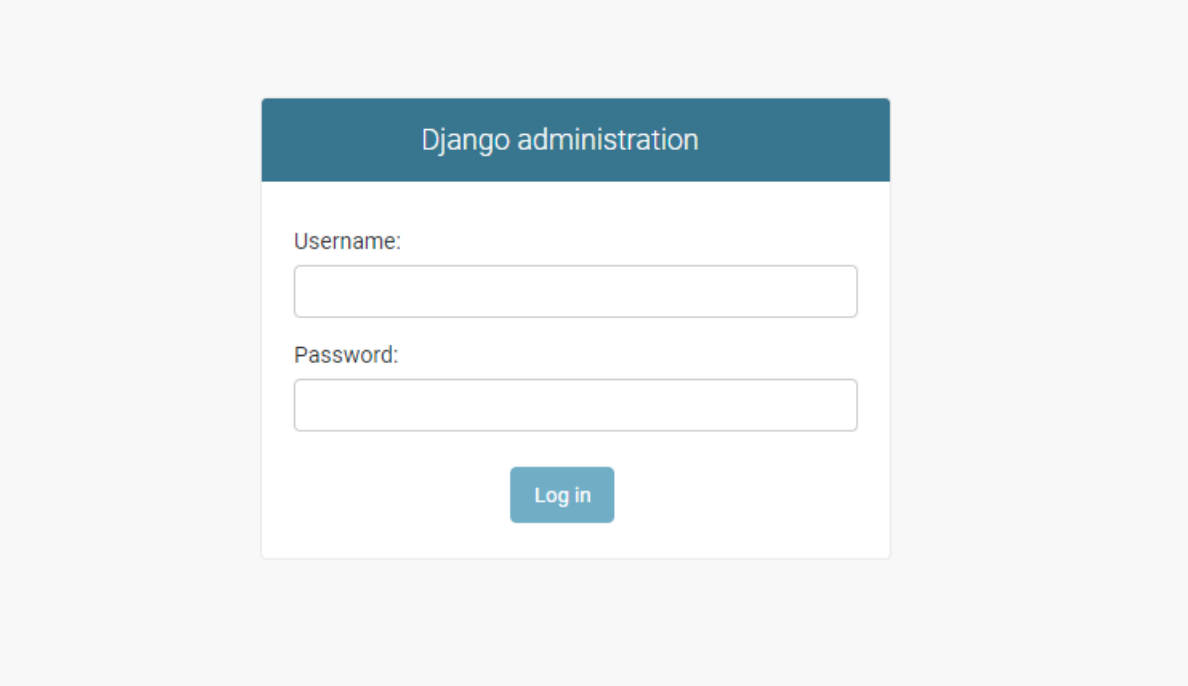

Рисунок 3.2 – Сторінка авторизації

Після авторизації користувач потрапляє на головну сторінку (рис. 3.3), де він може додавати або редагувати уроки чи пристрої:

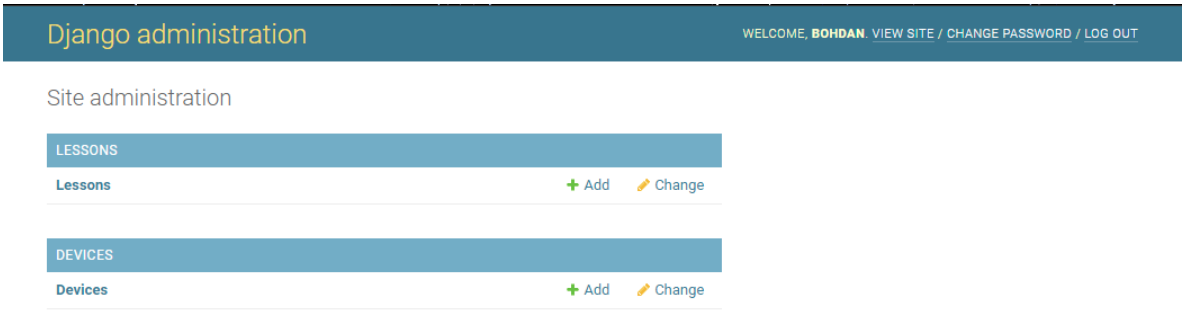

Рисунок 3.3 – Головна сторінка

Сторінка редагування пристроїв виглядає наступним чином (рис. 3.4):

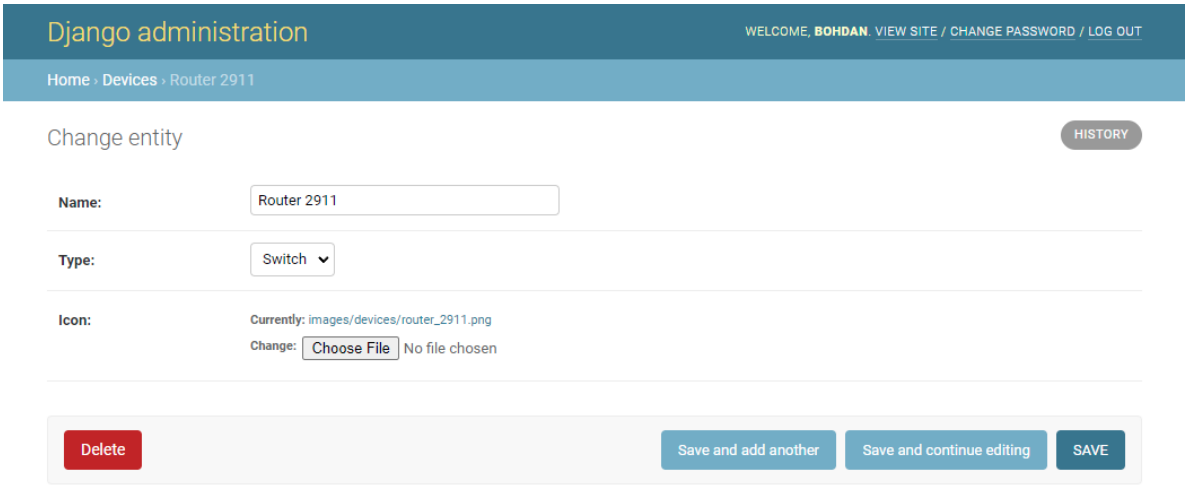

Рисунок 3.4 – Сторінка редагування пристроїв

Адміністратор може додати необхідну кількість пристроїв, для використання їх у конструкторі мереж. Для пристроя обирається тип, ім’я та картинка. Коли було додано всі потрібні пристрої – ми можемо створювати уроки. Сторінка редагування уроку виглядає наступним чином (рис. 3.5):

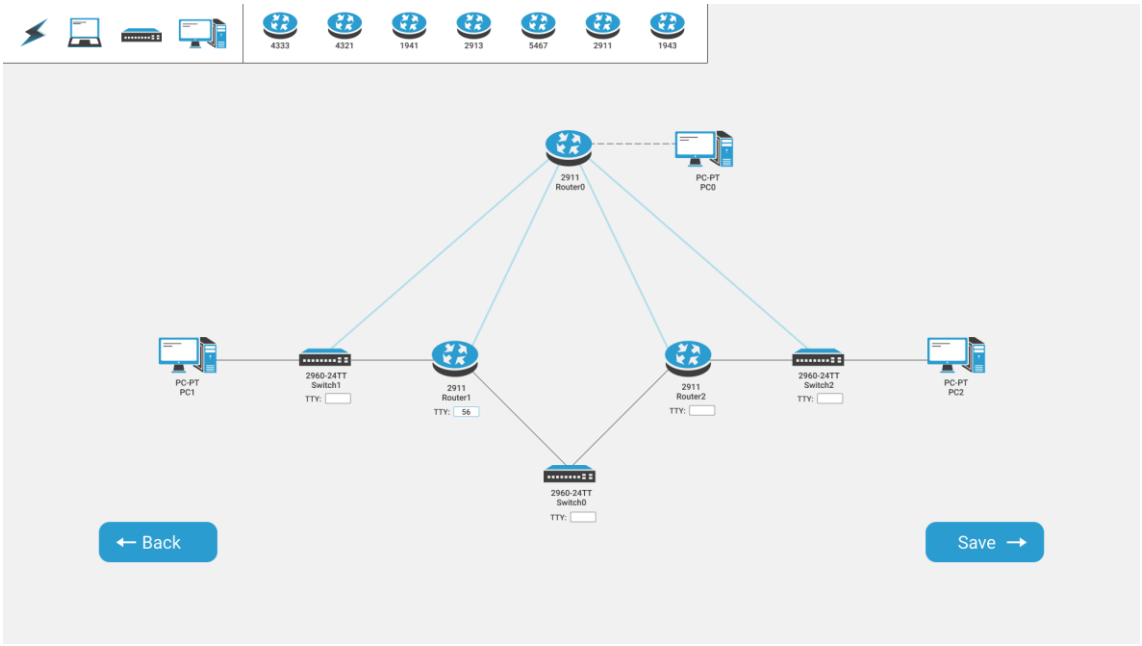

Рисунок 3.5 – Сторінка редагування уроку

Після того як вчитель зібрав мережу у лабораторії він має відтворити її у імітаторі та вказати номери TTY для пристроїв.

3.3. Графічний інтерфейс користувача

Після того як вчитель зібрав мережу у лабораторії він має відтворити її у імітаторі та вказати номери TTY для пристроїв. Коли лабораторна робота готова – вчитель може надати студентам короткий код, за яким студенти можуть під’єднатися до виконання роботи (рис. 3.6):

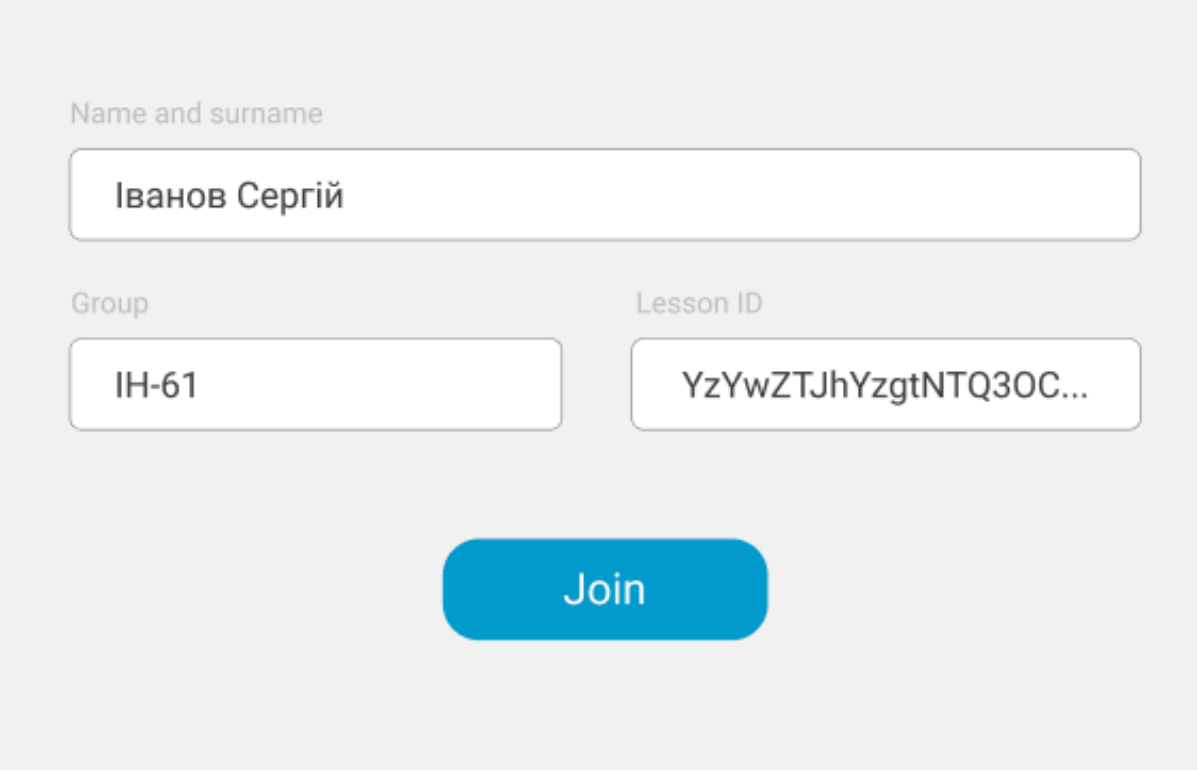

Рисунок 3.6 – Форма приєднання до лабораторної роботи

Коли студент приєднується до лабораторної роботи програма будує мережу (на основі мережі, створеної вчителем) та встановлює зв’язки між віртуальними та реальними пристроями. Також, завдяки лог записам, студент бачить команди, які були виконані на роутерах раніше, тому він може припинити або відновити роботу у будь-який час.

Тож коли студент вводить якусь команду – вона відправляється на сервер, який, з використанням SSH протоколу перенаправляє її до пристрою, який

відповідає заданому TTY. Таким чином ми можемо повністю налаштувати мережу (рис. 3.7).

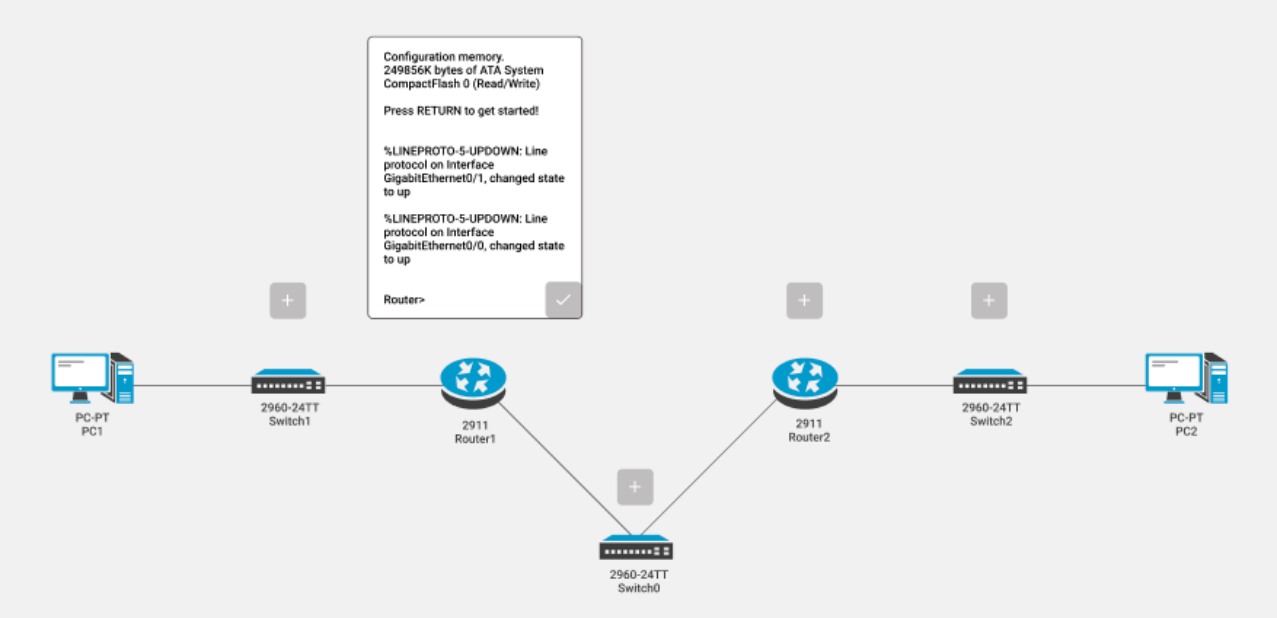

Рисунок 3.7 – Веб інтерфейс налаштування мережі

Завдяки технології Web-socket дані у вікні налаштувань оновлюються у реальному часі, тому студенти бачать результати роботи своїх колег та можуть їм допомагати. Під час виконання роботи вчитель може слідкувати за її станом за допомогою логу уроку, що зберігається у log файл (рис 3.9):

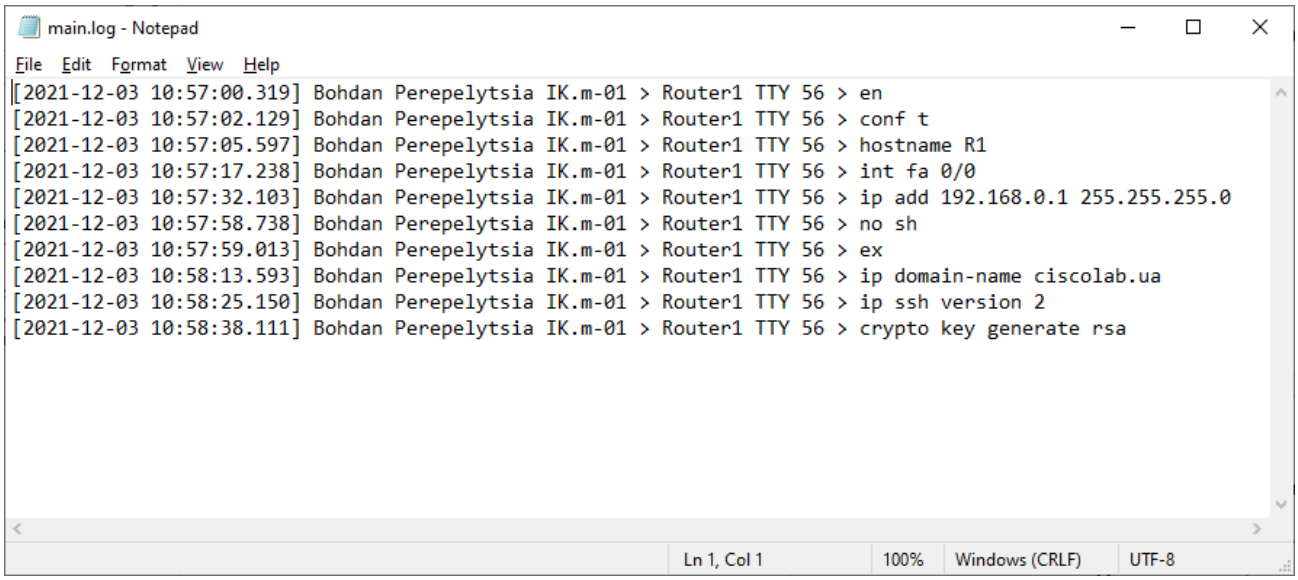

Рисунок 3.9 – Файл логів уроку

3.4. Тестування web-орієнтованої інформаційної системи у симуляторі PacketTracer та на реальному обладнанні Cisco

Тестування проводилося у середовищі PacketTracer, де була змодельована захищена мережа між двома віддаленими офісами з підтримкою QoS VoIP зв'язку та блокуванням доступу співробітників до соціальних мереж.

У середовищі GNS3 зібрана мережа на основі образів:

– Cisco ASA, IOS версії 9.7

– Cisco 3640, IOS версії 12

– Windows 7 VM.

Схема тестової мережі наведена на рисунку 3.10. Два ПК сполучені між собою через мережу Інтернет. Всі пристрої з'єднані із серверним роутером за допомогою октального кабеля, що допомагає налаштовувати їх, використовуючи SSH підключення між серверним роутером і ПК. Після конфігурації ми отримали повністю функціонуючу мережу, яка налаштована віддалено та без фізичного доступу до пристроїв.

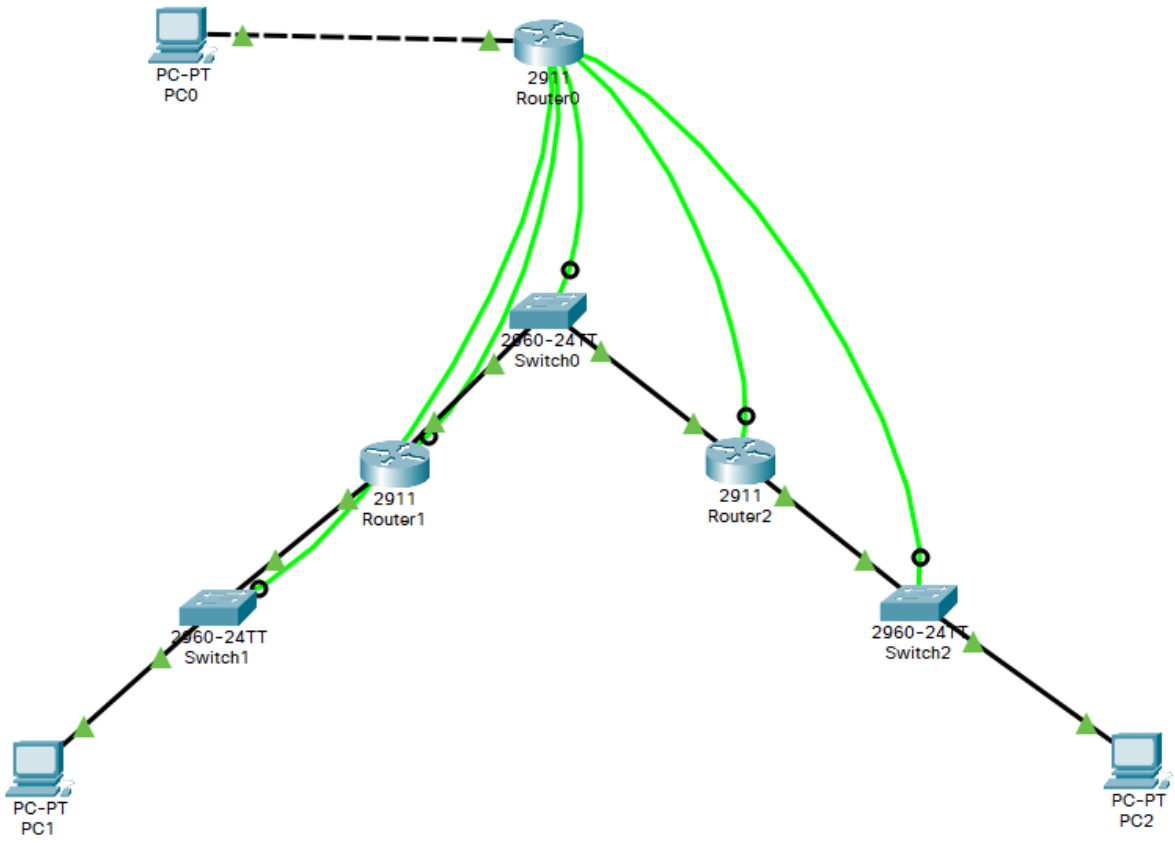

Рисунок 3.10 – Схема тестової мережі

Для перевірки правильності налаштування мережі ми використаємо команду ping на комп’ютерах (рис. 3.11, рис. 1.12).

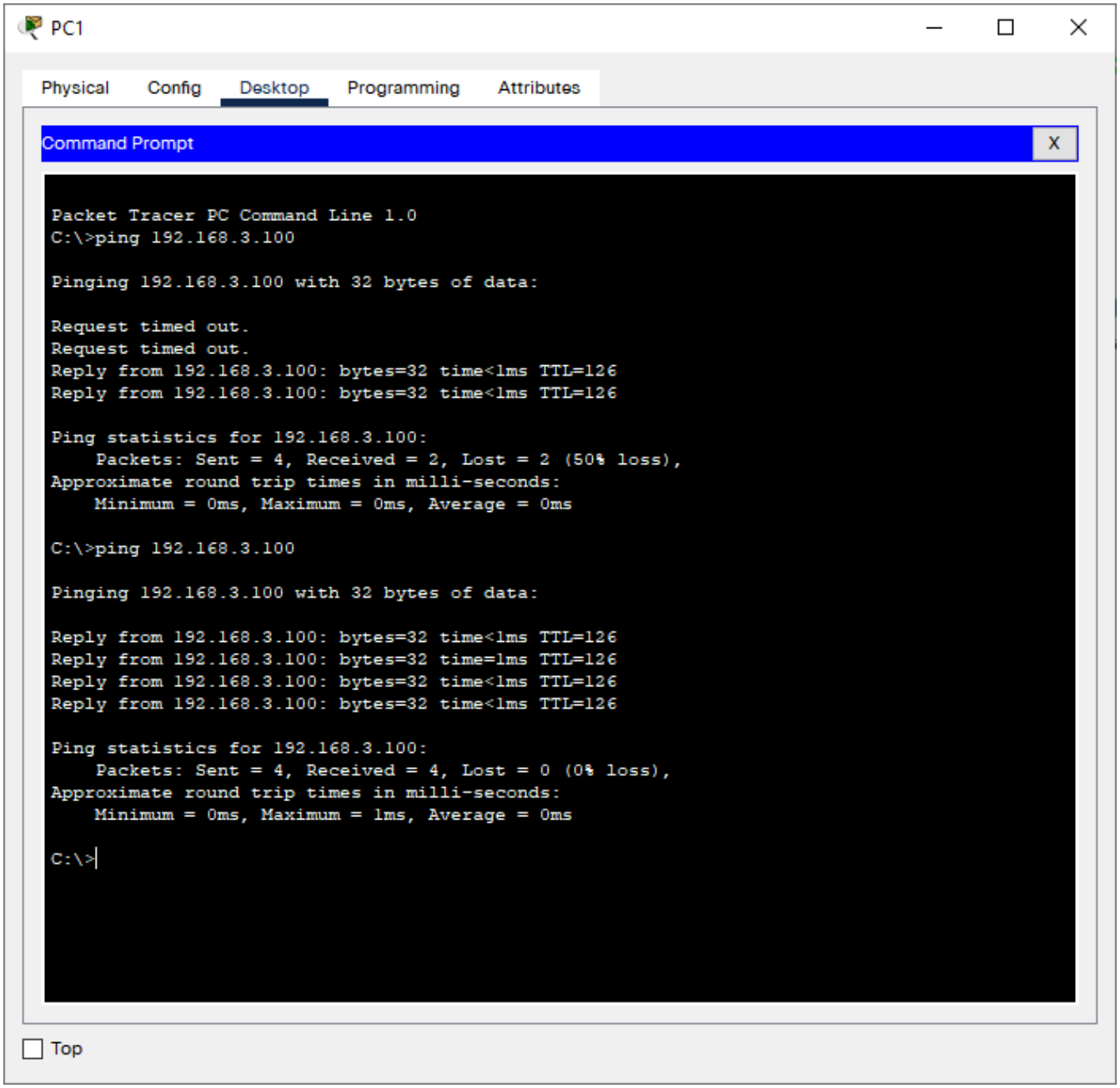

Рисунок 3.11 – Перевірка налаштування на PC1

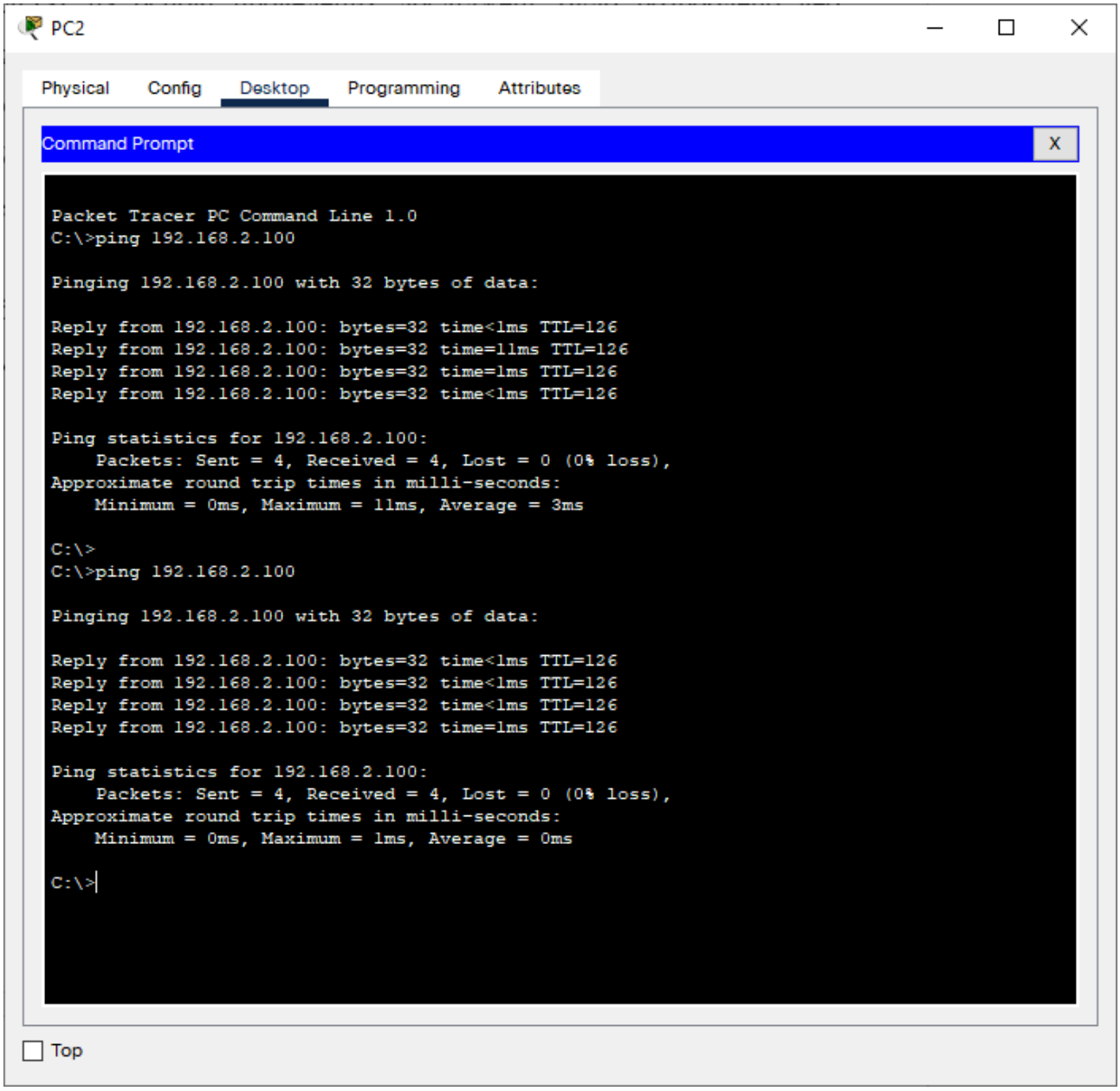

Рисунок 3.12 – Перевірка налаштування на PC2

Як ми бачимо з наведених рисунків – з’єднання встановлене, тож мережа налаштована вірно.

Правильність налаштування мережі у нашому додатку ми можемо перевірити за допомогою команди ping між роутерами (рис. 3.13):

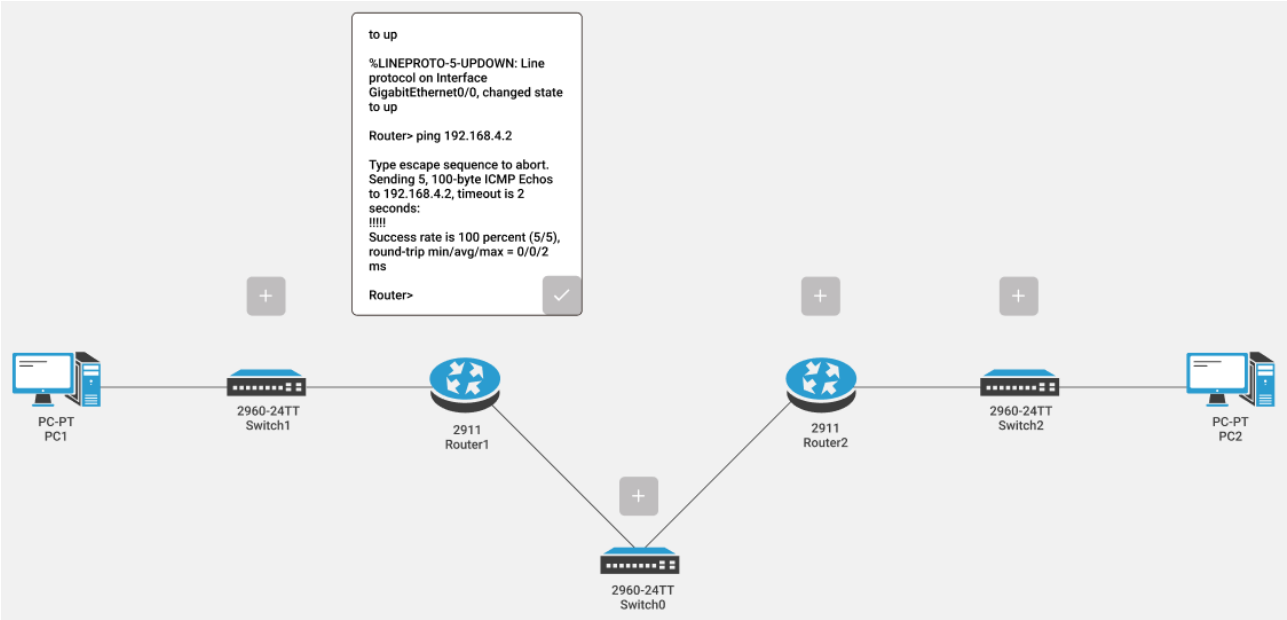

Рисунок 3.13 – Перевірка налаштування мережі у додатку

Як ми бачимо з’єднання пройшло успішно, тож мережа між роутерами успішно налаштована. Таким чином можна протестувати з’єднання між усіма роутерами і переконатися що все працює нормально.

ВИСНОВКИ

У ході виконання кваліфікаційної магістерської роботи було проведено критичний аналіз літературних джерел, пов’язаних з протоколами віддаленого доступу, виходячи з яких можна побачити що SSH є найбільш поширеним та безпечним через його особливий механізм зберігання ключів доступу. Також з’ясувалося що октальний кабель є дуже гарним інструментом у випадку необхідності асинхронно налаштувати декілька пристроїв.

Було здійснено побудову тестової мережі, що включає в себе серверний роутер, що поєднаний із трьома іншими роутерами і двома свічами октальним кабелем. Було налаштовано SSH з’єднання між роутером і ПК та сконфігуровано інші пристрої, використовуючи це з’єднання. Після конфігурації ми отримали повністю функціонуючу мережу, яка налаштована віддалено та без фізичного доступу до пристроїв.

Як результат на основі проведених досліджень було розроблено веб додаток, який дозволяє вчителю будувати телекомунікаційні мережі у віддаленій телеком-лабораторії та надавати студентам можливість налаштовувати і тестувати їх без безпосередньої присутності, що покращує навчальний досвід студентів. Система надає можливість передивлятися історію змін на кожному з пристроїв та відслідкувати типові помилки чи неточності студентів, що полегшує роботу вчителя. Систему реалізовано у формі web-додатку з використанням мови програмування JavaScript та технології Web-socket.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Сети и системы передачи информации: телекоммуникационные сети: учебник и практикум для академического бакалавриата / К. Е. Самуйлов [и др.]; под редакцией К. Е. Самуйлова, И. А. Шалимова, Д. С. Кулябова. — Москва: Издательство Юрайт, 2016. — 363 с.
2. Зиангирова, Л. Ф. Инфокоммуникационные системы и сети : учебное пособие для СПО / Л. Ф. Зиангирова. — Саратов : Профобразование, Ай Пи Ар Медиа, 2019. — 128 с. — ISBN 978-5-4488-0302-4, 978-5-4497-0183-1.
3. Галас, В. П. Вычислительные системы, сети и телекоммуникации. Часть 2. Сети и телекоммуникации : электронный учебник / В. П. Галас. — Владимир : Владимирский государственный университет им. А.Г. и Н.Г. Столетовых, 2016. — 311 с. — ISBN 2227-8397.
4. Альбекова, З. М. Инфокоммуникационные системы и сети : учебное пособие (лабораторный практикум) / З. М. Альбекова. — Ставрополь : Северо-Кавказский федеральный университет, 2019. — 112 с. — ISBN 2227-8397.
5. Ковган, Н. М. Компьютерные сети : учебное пособие / Н. М. Ковган. — Минск : Республиканский институт профессионального образования (РИПО), 2019. — 179 с. — ISBN 978-985-503-947-2.
6. Проскуряков, А. В. Компьютерные сети. Основы построения компьютерных сетей и телекоммуникаций : учебное пособие / А. В. Проскуряков. — Ростов-на-Дону, Таганрог : Издательство Южного федерального университета, 2018. — 201 с. — ISBN 978-5-9275-2792-2.
7. Васин, Н. Н. Сети и системы передачи информации : методические указания по курсовому проектированию / Н. Н. Васин, М. В. Кузнецов, И. В. Ротенштейн. — Самара : Поволжский государственный университет телекоммуникаций и информатики, 2016. — 58 с. — ISBN 2227-8397.
8. Prof Adithya Srivastava. SSH Protocol Architecture / Prof Adithya Srivastava. — Independently published, 2018. — 39 p. — ISBN 978-1982922429.

9. Michael W Lucas. SSH Mastery: OpenSSH, PuTTY, Tunnels and Keys (IT Mastery) / Michael W Lucas. — Tilted Windmill Press; 2nd ed. Edition, 2018. — 242 p. — ISBN 978-1642350029.
10. Walter Goralski. The Illustrated Network: How TCP/IP Works in a Modern Network / Walter Goralski. — Morgan Kaufmann; 2nd edition, 2017. — 936p. — ISBN 978-0128110270.
11. Dr. Hedaya Mahmood Alasooly. Quick Guide for Obtaining Free Remote Desktop Protocol (RDP) / Dr. Hedaya Mahmood Alasooly, 2020. — 50 p. — ISBN 979-8563605145.
12. Смычёк, М. А. Технологические сети и системы связи : учебное пособие / М. А. Смычёк. — 2-е изд. — Москва, Вологда : Инфра-Инженерия, 2019. — 400 с. — ISBN 978-5-9729-0338-2.

ДОДАТКИ

Додаток А

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Mag Web UI</title>
    <base href="/" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="icon" type="image/x-icon" href="assets/favicon.ico" />
  </head>
  <body class="mat-app-background" oncontextmenu="return false;">
    <app-root></app-root>
  </body>
</html>

<div>
  <router-outlet></router-outlet>
</div>
<div class="content">
  <div class="default-header"><h1>Servers</h1></div>
  <div class="default-content">
    <app-server-discovery></app-server-discovery>

    <div class="mat-elevation-z8">
      <mat-table #table [dataSource]="dataSource">
        <ng-container matColumnDef="id">
          <mat-header-cell *matHeaderCellDef> ID </mat-header-cell>
          <mat-cell *matCellDef="let row"> {{ row.id }} </mat-cell>
        </ng-container>

        <ng-container matColumnDef="name">
          <mat-header-cell *matHeaderCellDef> Name </mat-header-cell>
          <mat-cell *matCellDef="let row">
            <a
              *ngIf="getServerStatus(row)  ===  'running'  ||  row.location  ===
'remote' || row.location === 'bundled'"
              [routerLink]="['/server', row.id, 'lessons']"
              class="table-link"
              >{{ row.name }}</a
```

```
            >
            <span
              *ngIf="getServerStatus(row)  !=  'running'  &&  row.location  !==
'remote' && row.location !== 'bundled'"
              >{{ row.name }}</span
            >
          </mat-cell>
        </ng-container>

        <ng-container matColumnDef="location">
          <mat-header-cell *matHeaderCellDef> Location </mat-header-cell>
          <mat-cell *matCellDef="let row"> {{ row.location }} </mat-cell>
        </ng-container>

        <ng-container matColumnDef="ip">
          <mat-header-cell *matHeaderCellDef> Host </mat-header-cell>
          <mat-cell *matCellDef="let row"> {{ row.host }} </mat-cell>
        </ng-container>

        <ng-container matColumnDef="port">
          <mat-header-cell *matHeaderCellDef> Port </mat-header-cell>
          <mat-cell *matCellDef="let row"> {{ row.port }} </mat-cell>
        </ng-container>

        <ng-container matColumnDef="actions">
          <mat-header-cell *matHeaderCellDef> Actions </mat-header-cell>
          <mat-cell *matCellDef="let row" style="text-align: right">
            <button
              mat-icon-button
              matTooltip="Go to lessons"
              matTooltipClass="custom-tooltip"
              (click)="openLessons(row)"
              *ngIf="getServerStatus(row)  ===  'running'  ||  row.location  ===
'remote' || row.location === 'bundled'"
            >
              <mat-icon aria-label="Go to lessons">arrow_forward</mat-icon>
            </button>

            <button
              mat-icon-button
              matTooltip="Start server"
```

```
              matTooltipClass="custom-tooltip"
              (click)="startServer(row)"
              *ngIf="row.location   ===   'local'   &&   getServerStatus(row)   ===
'stopped'"
            >
              <mat-icon aria-label="Start server">play_arrow</mat-icon>
            </button>

            <button
              mat-icon-button
              matTooltip="Stop server"
              matTooltipClass="custom-tooltip"
              (click)="stopServer(row)"
              *ngIf="row.location   ===   'local'   &&   getServerStatus(row)   ===
'running'"
            >
              <mat-icon aria-label="Stop server">stop</mat-icon>
            </button>

            <mat-spinner
              [diameter]="24"
              *ngIf="row.location   ===   'local'   &&   getServerStatus(row)   ===
'starting'"
            ></mat-spinner>

            <button
              mat-icon-button
              matTooltip="Remove server"
              matTooltipClass="custom-tooltip"
              (click)="deleteServer(row)"
            >
              <mat-icon aria-label="Remove server">delete</mat-icon>
            </button>
          </mat-cell>
        </ng-container>

        <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
        <mat-row *matRowDef="let row; columns: displayedColumns"></mat-row>
      </mat-table>
    </div>
```

```
    <div class="buttons-bar">
      <button     *ngIf="isElectronApp"     mat-raised-button     class="button"
(click)="startLocalServer()">
        Start local server
      </button>
      <button       mat-raised-button       class="button"       color="primary"
(click)="createModal()">Add server</button>
    </div>
  </div>
</div>

<span>{{ confirmationMessage }}</span>
<div mat-dialog-actions *ngIf="!isOpen">
  <button       mat-button       class="cancelButton"       (click)="onNoClick()"
color="accent">No</button>
  <button  mat-button  class="confirmButton"  (click)="onYesClick()"  tabindex="2"
mat-raised-button color="primary">
    Yes
  </button>
</div>
<div mat-dialog-actions *ngIf="isOpen"><button mat-button (click)="onNoClick()"
color="accent">Ok</button></div>

<header>
  <mat-toolbar color="primary">
    <button mat-icon-button><mat-icon svgIcon="gns3"></mat-icon></button>

    <button mat-button routerLink="/servers">Servers</button>

    <button  *ngIf="!recentlyOpenedLessonId  &&  serverIdLessonList"  mat-button
(click)="listLessons()">
      Lessons
    </button>

    <button *ngIf="recentlyOpenedLessonId && recentlyOpenedServerId" mat-button
(click)="backToLessons ()">
      Back to lessons
    </button>

    <span class="fill-space"></span>
```

```
    <button mat-button [matMenuTriggerFor]="menu">
      <mat-icon>more_vert</mat-icon>
    </button>

    <mat-menu #menu="matMenu">
      <button mat-menu-item routerLink="/settings">
        <mat-icon>settings</mat-icon>
        <span>Settings</span>
      </button>
      <button           mat-menu-item           routerLink="/installed-software"
[disabled]="!isInstalledSoftwareAvailable">
        <mat-icon>cloud_download</mat-icon>
        <span>Installed software</span>
      </button>
      <button mat-menu-item routerLink="/help">
        <mat-icon>help</mat-icon>
        <span>Help</span>
      </button>
    </mat-menu>
  </mat-toolbar>
</header>

<main class="mat-app-background"><router-outlet></router-outlet></main>

<app-progress></app-progress>

<svg   id="map"   #svg   class="map"   preserveAspectRatio="none"   movingCanvas
zoomingCanvas>
  <filter    id="grayscale"><feColorMatrix    id="feGrayscale"    type="saturate"
values="0" /></filter>
  <defs>
    <pattern
      attr.x="{{ drawingGridX }}"
      attr.y="{{ drawingGridY }}"
      id="gridDrawing"
      attr.width="{{ project.drawing_grid_size }}"
      attr.height="{{ project.drawing_grid_size }}"
      patternUnits="userSpaceOnUse"
    >
      <path
```

```
        attr.d="M   {{   project.drawing_grid_size   }}   0   L   0   0   0   {{
project.drawing_grid_size }}"
        fill="none"
        stroke="silver"
        attr.stroke-width="{{ gridVisibility }}"
      />
    </pattern>
  </defs>

  <defs>
    <pattern
      attr.x="{{ nodeGridX }}"
      attr.y="{{ nodeGridY }}"
      id="gridNode"
      attr.width="{{ project.grid_size }}"
      attr.height="{{ project.grid_size }}"
      patternUnits="userSpaceOnUse"
    >
      <path
        attr.d="M {{ project.grid_size }} 0 L 0 0 0 {{ project.grid_size }}"
        fill="none"
        stroke="DarkSlateGray"
        attr.stroke-width="{{ gridVisibility }}"
      />
    </pattern>
  </defs>

  <rect width="100%" height="100%" fill="url(#gridDrawing)" />
  <rect width="100%" height="100%" fill="url(#gridNode)" />
</svg>

<app-drawing-adding [svg]="svg"></app-drawing-adding>
<app-drawing-resizing></app-drawing-resizing>
<app-selection-control></app-selection-control>
<app-selection-select></app-selection-select>
<app-text-editor #textEditor [server]="server" [svg]="svg"></app-text-editor>
<app-draggable-selection [svg]="svg"></app-draggable-selection>
```

Додаток Б

```
body {
  background-color: #e8ecef;
}

img.logo-header {
  width: 50px;
}

a.table-link {
  color: #0097a7;
}

.snackabar-success {
  background: #0097a7 !important;
  color: white !important;
}

.snackbar-warning {
  background: rgb(197, 199, 64) !important;
  color: white !important;
}

.snackbar-error {
  background: #b00020 !important;
  color: white !important;
}

.mat-dialog-actions {
  margin-bottom: -12px !important;
}

@-moz-document url-prefix() {
  .temporaryElement {
    line-height: 1.4em;
  }
}

.full-width-field {
  width: 100%;
}
```

```
app-root {
  width: 100%;
}

mat-menu-panel {
  min-height: 0px;
}

.custom-tooltip {
  background-color: grey;
  color: #ffffff;
}

mat-menu-panel {
  min-height: 0px;
}

.dark {
  background: #263238 !important;
}

.light {
  background: white !important;
}

html,
body {
  height: 100%;
}

app-root,
app-default-layout {
  height: 100%;
}

app-default-layout {
  display: flex;
  flex-direction: column;
  height: 100vh;
}
```

```
.content {
  /*flex: 1 0 auto;*/
}

.footer {
  /*flex-shrink: 0;*/
  padding: 20px;
  margin: auto 0 0 0;
  /*background-color: #0097a7;*/
}

.default-content {
  margin: 0 auto;
  max-width: 940px;
  padding-top: 20px;
  padding-bottom: 20px;
}

header {
  box-shadow: 0 3px 5px -1px rgba(0, 0, 0, 0.2), 0 6px 10px 0 rgba(0, 0, 0, 0.14),
0 1px 18px 0 rgba(0, 0, 0, 0.12);
  z-index: 10;
}

/*main {*/
/*height: 100%;*/
/*}*/

.default-header h1 {
  font-weight: 300;
  margin: 0;
  font-size: 20px;
  padding: 28px 8px;
}

.default-header {
  margin: 0 auto;
  max-width: 940px;
  /*background-color: #0097a7;*/
}
```

```
.buttons-bar {
  padding-top: 10px;
  text-align: right;
}

.fill-space {
  flex: 1 1 auto;
}

.container > * {
  width: 100%;
}

.container {
  padding: 0%;
}

.mat-dialog-content > * {
  width: 100%;
}

.wrapper {
  height: 600px;
}

app-root,
app-project-map,
.project-map,
app-map {
  width: auto;
}

g.node:hover {
  background-color: #0097a7;
}

.lessons-map {
  background-color: #e8ecef;
}
```

```
#lessons-titlebar {
  position: fixed;
  top: 0px;
  left: 0px;
  right: 0px;
  height: 60px;
  padding: 0px 20px;
  background-color: #20313b;
  display: flex;
  justify-content: space-between;
  align-items: center;
  box-shadow: 3px 3px 10px rgba(0, 0, 0, 0.2);
  z-index: 2;

  .button {
    position: relative;
    top: -2px;

    mat-icon {
      width: 30px !important;
      height: 30px !important;
    }
  }

  .primary-controls {
    border-right: 1px solid rgba(255, 255, 255, 0.3);
    padding-right: 15px;
    margin-right: 15px;

    &.lightTheme {
      border-right: 1px solid black;
    }
  }

  .menu-button-group {
    display: flex;
    align-items: center;
    height: 24px;
  }

  .menu-button {
```

```
    display: flex;
    justify-content: center;
    align-items: center;
    height: 36px;
    width: 36px;
    margin: 0px 8px;
    border-radius: 18px;
    background: none;
    font-size: 20px;
  }

  .add-menu-button mat-icon {
    font-size: 28px !important;
  }

  .selected {
    background: rgba(0, 151, 167, 0.1);

    mat-icon {
      color: #0097a7 !important;
    }
  }

  .project-titlebar-controls {
    display: flex;
    align-items: center;
  }

  &.lightTheme {
    background-color: white !important;

    .selected mat-icon {
      color: #0097a7 !important;
    }
  }
}

#project-toolbar {
  position: fixed;
  top: 60px;
  left: 0px;
```

```
width: 50px;
margin: 20px;
background-color: #20313b;
border-radius: 6px;
box-shadow: 1px 1px 10px rgba(0, 0, 0, 0.2);
z-index: 2;

mat-icon {
  font-size: 20px;
}

.menu-button {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 36px;
  width: 36px;
  border-radius: 18px;
  background: none;
  margin: 2px 0px;
}

.zoom-button {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 36px;
  width: 36px;
  border-radius: 18px;
  background: none;
  margin: 2px 0px;
  color: white;

  mat-icon {
    font-size: 24px !important;
  }
}

.reset-zoom-button {
  display: flex;
  justify-content: center;
```

```
  align-items: center;
  height: 36px;
  width: 36px;
  border-radius: 18px;
  background: none;
  margin: 2px 0px;
  color: white;

  mat-icon {
    font-size: 18px !important;
  }
}

.selection-button {
  margin-bottom: 0px !important;
}

.snapshot-button mat-icon {
  font-size: 16px;
}

.section {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  border-top: 1px solid rgba(255, 255, 255, 0.3);
  padding: 5px 0px;

  &:first-child {
    border: none;
  }
}

&.lightTheme {
  background-color: rgba(244, 248, 252, 0.95) !important;

  .zoom-button {
    opacity: 0.7;
    color: black;
  }
```

```
    .reset-zoom-button {
      opacity: 0.7;
      color: black;
    }

    .section {
      border-top: 1px solid rgba(0, 0, 0, 0.1);

      &:first-child {
        border: none;
      }
    }
  }
}

img {
  -webkit-filter: invert(1);
  filter: invert(1);
}

.lightTheme {
  color: black !important;
}

.darkTheme {
  color: white !important;
}

#show-menu-wrapper {
  position: fixed;
  background: transparent;
  top: 0px;
  left: 92px;
  background: #263238;
  height: 72px;
  padding-top: 16px;

  .arrow-button {
    outline: 0 !important;
  }
```

```
}

.shadowed {
  box-shadow: 0 4px 8px -4px rgba(0, 0, 0, 0.2), 0 6px 20px -20px rgba(0, 0, 0,
0.19);
}

.non-visible {
  display: none;
}

#menu-wrapper {
  position: fixed;
  background: transparent;
  top: 0px;
  left: 92px;
  right: 0px;
  background: #263238;
  height: 72px;
  padding-top: 16px;
  padding-bottom: 16px;
  transition: 35s;
  width: 0;
  overflow: hidden;
  transition: 0.15s;
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
  display: flex;

  .menu-button {
    outline: 0 !important;
    transition: 0.5s;
    margin-bottom: 16px;
    width: 40px;
    margin-right: 12px !important;
    margin-left: 12px !important;
    background: transparent;
    padding: 0;
    border: none;
    background-color: transparent;
  }
```

```
  .arrow-button {
    outline: 0 !important;
    transition: 0.5s;
    margin-bottom: 16px;
  }
}

.extended {
  width: 830px !important;
  height: 100%;
  overflow: hidden;
}

mat-divider.divider {
  height: 40px;
  margin-left: 1px;
  margin-right: 7px;
  width: 10px;
  color: gray;
}

@-moz-document url-prefix() {
  /** fixes gray background of drawing menu on Firefox **/
  .mat-drawer-content {
    display: inline !important;
  }
}

.shadow {
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
}

.mat-drawer-backdrop.mat-drawer-shown {
  background-color: transparent;
}

.project-toolbar .mat-toolbar-multiple-rows {
  width: auto !important;
}

.loading-spinner {
```

```
  position: absolute;
  top: 50%;
  width: 100px;
  margin-left: -50px;
  margin-top: -50px;
  left: 50%;
}

line.selected {
  stroke: #0097a7 !important;
}

svg.map image:hover,
svg.map image.chosen,
g.selected {
  -webkit-filter: grayscale(100%);
  -moz-filter: grayscale(100%);
  -ms-filter: grayscale(100%);
  -o-filter: grayscale(100%);
  filter: grayscale(100%);
  filter: gray;
  filter: url('#grayscale'); /* Chrome doesn't support CSS filters on SVG */
}

path.selected {
  stroke: darkred;
}

.selected > .interface_label_border {
  stroke: black;
  fill: none;
}

.selection-line-tool .selection {
  fill: #7ccbe1;
  stroke: #66aec2;
  fill-opacity: 0.3;
  stroke-opacity: 0.7;
  stroke-width: 1;
  stroke-dasharray: 5, 5;
}
```

```
g.node text,
.noselect {
  -webkit-touch-callout: none;
  -webkit-user-select: none;
  -khtml-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
}

.lessons -toolbar button {
  outline: 0;
  border: none;
  -moz-outline-style: none;
}

.options-item {
  padding-left: 15px;
  padding-right: 15px;
}

.context-menu-items .mat-menu-item {
  line-height: 24px !important;
  height: 24px !important;
  font-size: 13px !important;
  padding: 0 6px;
  outline: none !important;
}

.context-menu-items .mat-menu-item .mat-icon {
  margin-right: 3px;
}

.context-menu-items .mat-menu-item:focus {
  background: none;
}

.visible {
  display: none;
}
```

```
mat-menu-panel {
  min-height: 0px;
}

.unmarked {
  color: white !important;
}

.unmarkedLight {
  color: black !important;
}

.marked {
  color: #0097a7 !important;
}
@media screen and (max-width: 700px) {
  .consoleWrapper {
    visibility: hidden;
  }
}

.consoleWrapper {
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
  position: fixed;
  bottom: 40px;
  left: 80px;
  height: 180px;
  width: 600px;
  background: #000000 !important;
  color: white;
  overflow: hidden;
  font-size: 12px;
  border-radius: 8px;
}

.lightTheme {
  background: white !important;
  color: black;
}
```

```
.filterButton {
  background: transparent;
  color: white;
  border: none;
  margin-top: 0px;
  outline: none;
  color: #dbd5d5;
  font-weight: bold;
  padding: 0px;
}

.consoleFiltering {
  display: flex;
}

.consoleHeader {
  width: 100%;
  font-size: 12px;
  overflow: hidden;
  display: flex;
  padding: 2px;
  justify-content: space-between;
  background: #263238 !important;
}

.lightThemeConsoleHeader {
  background: white !important;
  color: black !important;
}

:host ::ng-deep .mat-tab-label {
  height: 3rem !important;
  min-width: 8rem !important;
}

:host ::ng-deep .mat-tab-label.mat-tab-label-active {
  border-bottom: 2px solid #0097a7;
  box-sizing: border-box;
  font-weight: 1200;
}
```

```
:host ::ng-deep .mat-ink-bar {
  display: none !important;
}

.tabs {
  width: 80%;
}

.console {
  width: 596px;
  height: 120px;
  overflow-y: scroll;
  padding: 2px;
  color: #dbd5d5;
  scrollbar-color: darkgrey #263238;
  scrollbar-width: thin;
}

.xterm-console {
  background: black;
}

.consoleInput {
  width: 100%;
  height: 30px;
  padding: 2px;
  display: flex;
}

.commandLine {
  background-color: transparent;
  color: white;
  border: none;
}

.inputIcon {
  margin-top: 2px;
}

mat-icon {
  font-size: 20px;
```

```
  width: 20px;
  height: 20px;
}

input:focus {
  outline: none;
}

::-webkit-scrollbar {
  width: 0.5em;
}

::-webkit-scrollbar-track {
  -webkit-box-shadow: inset 0 0 6px rgba(0, 0, 0, 0.3);
}

::-webkit-scrollbar-thumb {
  background-color: darkgrey;
  outline: 1px solid #263238;
}

.closeButton {
  cursor: pointer;
}
```

Додаток В

```
import { Component, EventEmitter, Input, OnInit, Output } from '@angular/core';
import { FormControl } from '@angular/forms';
import { ResizeEvent } from 'angular-resizable-element';
import { Node } from '../../../cartography/models/node';
import { Project } from '../../../models/project';
import { Server } from '../../../models/server';
import { MapSettingsService } from '../../../services/mapsettings.service';
import { NodeConsoleService } from '../../../services/nodeConsole.service';
import { ThemeService } from '../../../services/theme.service';

@Component({
  selector: 'app-console-wrapper',
  templateUrl: './console-wrapper.component.html',
  styleUrls: ['./console-wrapper.component.scss'],
})
```

```
export class ConsoleWrapperComponent implements OnInit {
  @Input() server: Server;
  @Input() project: Project;
  @Output() closeConsole = new EventEmitter<boolean>();

  filters: string[] = ['all', 'errors', 'warnings', 'info', 'map updates', 'server
requests'];
  selectedFilter: string = 'all';

  public style: object = {};
  public styleInside: object = { height: `120px` };
  public isDraggingEnabled: boolean = false;
  public isLightThemeEnabled: boolean = false;
  public isMinimized: boolean = false;

  public resizedWidth: number = 720;
  public resizedHeight: number = 480;

  constructor(
    private consoleService: NodeConsoleService,
    private themeService: ThemeService,
    private mapSettingsService: MapSettingsService
  ) {}

  nodes: Node[] = [];
  selected = new FormControl(0);

  ngOnInit() {
    this.themeService.getActualTheme() === 'light'
      ? (this.isLightThemeEnabled = true)
      : (this.isLightThemeEnabled = false);
    this.style = { bottom: '20px', left: '80px', width: '720px', height: '460px'
};

    this.consoleService.nodeConsoleTrigger.subscribe((node) => {
      this.addTab(node, true);
    });

    this.consoleService.closeNodeConsoleTrigger.subscribe((node) => {
      let index = this.nodes.findIndex((n) => n.node_id === node.node_id);
      this.removeTab(index);
```

```
    });
  }

  minimize(value: boolean) {
    this.isMinimized = value;
    if (!value) {
      this.style    =    {    bottom:    '20px',    left:    '80px',    width:
`${this.resizedWidth}px`, height: `${this.resizedHeight}px` };
    } else {
      this.style    =    {    bottom:    '20px',    left:    '20px',    width:
`${this.resizedWidth}px`, height: '56px' };
    }
  }

  addTab(node: Node, selectAfterAdding: boolean) {
    this.minimize(false);
    this.nodes.push(node);

    if (selectAfterAdding) {
      this.selected.setValue(this.nodes.length);
    }

    this.consoleService.openConsoles++;
  }

  removeTab(index: number) {
    this.nodes.splice(index, 1);
    this.consoleService.openConsoles--;
  }

  toggleDragging(value: boolean) {
    this.isDraggingEnabled = value;
  }

  dragWidget(event) {
    let x: number = Number(event.movementX);
    let y: number = Number(event.movementY);

    let width: number = Number(this.style['width'].split('px')[0]);
    let height: number = Number(this.style['height'].split('px')[0]);
    let left: number = Number(this.style['left'].split('px')[0]) + x;
```

```
  if (this.style['top']) {
    let top: number = Number(this.style['top'].split('px')[0]) + y;
    this.style = {
      position: 'fixed',
      left: `${left}px`,
      top: `${top}px`,
      width: `${width}px`,
      height: `${height}px`,
    };
  } else {
    let bottom: number = Number(this.style['bottom'].split('px')[0]) - y;
    this.style = {
      position: 'fixed',
      left: `${left}px`,
      bottom: `${bottom}px`,
      width: `${width}px`,
      height: `${height}px`,
    };
  }
}

validate(event: ResizeEvent): boolean {
  if (
    event.rectangle.width &&
    event.rectangle.height &&
    (event.rectangle.width < 500 || event.rectangle.height < 100)
  ) {
    return false;
  }
  return true;
}

onResizeEnd(event: ResizeEvent): void {
  this.style = {
    position: 'fixed',
    left: `${event.rectangle.left}px`,
    top: `${event.rectangle.top}px`,
    width: `${event.rectangle.width}px`,
    height: `${event.rectangle.height}px`,
  };
```

```
    this.styleInside = {
      height: `${event.rectangle.height - 60}px`,
      width: `${event.rectangle.width}px`,
    };

    this.consoleService.consoleResized.next({
      width: event.rectangle.width,
      height: event.rectangle.height - 53,
    });

    this.resizedWidth = event.rectangle.width;
    this.resizedHeight = event.rectangle.height;
  }

  close() {
    this.closeConsole.emit(false);
  }

  enableScroll(e) {
    this.mapSettingsService.isScrollDisabled.next(false);
  }

  disableScroll(e) {
    this.mapSettingsService.isScrollDisabled.next(true);
  }
}
```

es (78 sloc) 3.12 KB

RawBlame

```
import { ChangeDetectionStrategy, Component, Input } from '@angular/core';
import { MapSettingsService } from '../../../services/mapsettings.service';
import { ElectronService } from 'ngx-electron';
import { NodesDataSource } from '../../../cartography/datasources/nodes-datasource';
import { Project } from '../../../models/project';
import { Server } from '../../../models/server';
```

```
import { NodeService } from '../../../services/node.service';
import { ServerService } from '../../../services/server.service';
import { SettingsService } from '../../../services/settings.service';
import { ToasterService } from '../../../services/toaster.service';
import { NodeConsoleService } from '../../../services/nodeConsole.service';


@Component({
  selector: 'app-nodes-menu',
  templateUrl: './nodes-menu.component.html',
  styleUrls: ['./nodes-menu.component.scss'],
  changeDetection: ChangeDetectionStrategy.OnPush,
})
export class NodesMenuComponent {
  @Input('project') project: Project;
  @Input('server') server: Server;


  constructor(
    private nodeService: NodeService,
    private nodeConsoleService: NodeConsoleService,
    private nodesDataSource: NodesDataSource,
    private toasterService: ToasterService,
    private serverService: ServerService,
    private settingsService: SettingsService,
    private mapSettingsService: MapSettingsService,
    private electronService: ElectronService
  ) {}
```

```
  async startConsoleForAllNodes() {
    if (this.electronService.isElectronApp) {
      let consoleCommand = this.settingsService.getConsoleSettings()
        ? this.settingsService.getConsoleSettings()
        : this.nodeService.getDefaultCommand();


      let nodes = this.nodesDataSource.getItems();
      for (var node of nodes) {
        const request = {
          command: consoleCommand,
          type: node.console_type,
          host: node.console_host,
          port: node.console,
          name: node.name,
          project_id: node.project_id,
          node_id: node.node_id,
          server_url: this.serverService.getServerUrl(this.server),
        };
        await this.electronService.remote.require('./console-
executor.js').openConsole(request);
      }
    } else {
      if (this.mapSettingsService.openConsolesInWidget) {

this.nodeConsoleService.openConsolesForAllNodesInWidget(this.nodesDataSource.getItems()
);
      } else {

this.nodeConsoleService.openConsolesForAllNodesInNewTabs(this.nodesDataSource.getItems(
));
      }
```

```
  }
}


startNodes() {
  this.nodeService.startAll(this.server, this.project).subscribe(() => {
    this.toasterService.success('All nodes successfully started');
  });
}


stopNodes() {
  this.nodeService.stopAll(this.server, this.project).subscribe(() => {
    this.toasterService.success('All nodes successfully stopped');
  });
}


suspendNodes() {
  this.nodeService.suspendAll(this.server, this.project).subscribe(() => {
    this.toasterService.success('All nodes successfully suspended');
  });
}


reloadNodes() {
  this.nodeService.reloadAll(this.server, this.project).subscribe(() => {
    this.toasterService.success('All nodes successfully reloaded');
  });
}
```

```
      }
import { Component, Input, OnDestroy, OnInit, ViewChild } from '@angular/core';
import { Subscription } from 'rxjs';
import { MapNodeToNodeConverter } from '../../../cartography/converters/map/map-node-to-node-converter';
import { NodeToMapNodeConverter } from '../../../cartography/converters/map/node-to-map-node-converter';
import { PortToMapPortConverter } from '../../../cartography/converters/map/port-to-map-port-converter';
import { MapLinkCreated } from '../../../cartography/events/links';
import { LinksEventSource } from '../../../cartography/events/links-event-source';
import { NodesEventSource } from '../../../cartography/events/nodes-event-source';
import { MapNode } from '../../../cartography/models/map/map-node';
import { MapPort } from '../../../cartography/models/map/map-port';
import { DrawingLineWidget } from '../../../cartography/widgets/drawing-line';
import { NodeSelectInterfaceComponent } from '../../../components/project-map/node-select-interface/node-select-interface.component';
import { Link } from '../../../models/link';

@Component({
  selector: 'app-draw-link-tool',
  templateUrl: './draw-link-tool.component.html',
  styleUrls: ['./draw-link-tool.component.scss'],
})
export class DrawLinkToolComponent implements OnInit, OnDestroy {
  @Input() links: Link[];
  @ViewChild(NodeSelectInterfaceComponent) nodeSelectInterfaceMenu: NodeSelectInterfaceComponent;

  private nodeClicked$: Subscription;

  constructor(
    private drawingLineTool: DrawingLineWidget,
    private nodesEventSource: NodesEventSource,
    private linksEventSource: LinksEventSource,
    private mapNodeToNode: MapNodeToNodeConverter,
    private nodeToMapNode: NodeToMapNodeConverter,
    private portToMapPort: PortToMapPortConverter
  ) {}
```

```
  ngOnInit() {
    this.nodeClicked$ = this.nodesEventSource.clicked.subscribe((clickedEvent) =>
{
      let node = this.mapNodeToNode.convert(clickedEvent.datum);
      this.nodeSelectInterfaceMenu.open(node, clickedEvent.y, clickedEvent.x);
    });
  }

  ngOnDestroy() {
    if (this.drawingLineTool.isDrawing()) {
      this.drawingLineTool.stop();
    }
    this.nodeClicked$.unsubscribe();
  }

  public onChooseInterface(event) {
    const node: MapNode = this.nodeToMapNode.convert(event.node);
    const port: MapPort = this.portToMapPort.convert(event.port);
    if (this.drawingLineTool.isDrawing()) {
      const data = this.drawingLineTool.stop();
      this.linksEventSource.created.emit(new        MapLinkCreated(data['node'],
data['port'], node, port));
    } else {
      this.drawingLineTool.start(node.x + node.width / 2, node.y + node.height /
2, {
        node: node,
        port: port,
      });
    }
  }
}
```