МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Секція інформаційно-комунікаційних технологій

ВИПУСКНА РОБОТА

на тему:

«Графічний інтерфейс налаштування операторськоі мережі АТМ»

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

Керівник роботи **Великодний Д.В**

Студент гр. ІН-71 **Логутов С.В.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедри Довбиш А.С.

"_____" _____2021 г.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІН-71 спеціальності "Комп'ютерні науки" денної форми навчання Логутова Сергія Віталійовича.

Тема: «Графічний інтерфейс налаштування операторськоі мережі АТМ»

Затверджена наказом по СумДУ

№_____ от _____2021 г.

Зміст пояснювальної записки: 1) огляд існуючих рішень; 2) моделювання мережі з застосуванням технології АТМ в емуляторі GNS3 та роутерів Cisco; 3) інформаційне та програмне забезпечення графічного інтерфейсу налаштування мережі АТМ; 4) висновок.

Дата видачі завдання "_____"_____2020 г.

Керівник випускної роботи _____ Великодний Д.В.

Завдання прийняв до виконання _____ Логутов С.В.

РЕФЕРАТ

Записка: 66 стор., 16 рис.,1 додаток, 20 джерел.

Об’єкт дослідження — налаштування операторської мережі АТМ.

Мета роботи — розробка веб-додатку для налаштування операторської мережі АТМ.

Методи дослідження — метод аналітично-статистичний.

Результати — розроблено Графічний інтерфейс налаштування операторськоі мережі АТМ. Додаток створено на базі мови програмування JavaScript та фреймворків React, Express, Electron.

АТМ, JAVASCRIPT, REACT, EXPRESS, MYSQL, ELECTRON, РОУТЕР

ЗМІСТ

ВСТУП

Робота присвячена темі розробки графічного інтерфейсу налаштування операторської мережі ATM. Універсальна технологія передачі даних ATM - це, свого роду, остання спроба змістити Ethernet. Суть технології є досить простою, розробники вирішили поєднати і ущільнити всі види трафіку. Таким чином, по одному каналу передавалося відео, голос і дані. Весь потік безперервних даних розділяється на ділянки - інтервали. Ці інтервали короткі і їх легко комутувати.

Так як налаштування мережі ATM проводиться вручну і є досить тривалою і монотонною роботою було вирішено розробити веб-орієнтовану середу для генерації конфігурації для окремого роутера в мережі.

В рамках роботи потрібно провести аналіз предметної області, переглянути вже існуючи аналоги, оцінити переваги і недоліки, і розробити власний проект.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Технологія ATM

Технологія ATM (з англ. Asynchronous Transfer Mode – асинхронний спосіб передачі даних) є альтернативою Ethernet. Мережа ATM має класичну структуру великої територіальної мережі – кінцеві роутери з'єднуються індивідуальними каналами з комутаторами нижнього рівня, які в свою чергу з'єднуються з комутаторами вищих рівнів. Комутатори ATM, або ATM -світчі використовують адреси кінцевих вузлів розміром в 20 байтів для маршрутизації трафіку на основі техніки віртуальних каналів.

Пакети комутуються на основі VCI (з англ. Virtual Channel Identifier, – ідентифікатор віртуального каналу), який назначається з'єднанню при його встановленні та знищується при розриві з'єднання. Адреса кінцевого вузла ATM, на основі якого прокладається віртуальний канал має ієрархічну структуру, що нагадує номер в телефонній мережі, та використовує префікси, які відповідають кодам країн, міст, мереж провайдерів тощо. Це спрощує маршрутизацію запитів встановлення з'єднання. Віртуальні з'єднання можуть бути постійними (англ. Permanent Virtual Circuit, PVC) і комутованими (англ. Switched Virtual Circuit, SVC).

Для прискорення комутації в великих мережах використовується поняття віртуального шляху – Virtual Path, який об'єднує віртуальні канали, що мають спільний маршрут між початковим і кінцевим вузлами або спільну частину маршрута між деякими двома ATM -світчами.

VPI (з англ. Virtual Path Identifier – ідентифікатор віртуального шляху) є старшою частиною локальної адреси і являє собою спільний префікс для деякої кількості віртуальних каналів. Таким чином, ідея агрегування адрес в технології ATM застосована на двох рівнях: на рівні адрес кінцевих вузлів (працює на стадії

встановлення віртуального каналу) і на рівні номерів віртуальних каналів (працює при передачі даних по наявному віртуальному каналі) [1].

У технології ATM для перенесення даних використовуються комірки. Принципово комірка відрізняється від кадру тільки тим, що має, по-перше, фіксований, по-друге, невеликий розмір. Довжина комірки становить 53 байта, а поля даних - 48 байт (рис 1.1). Саме такі розміри дозволяє мережі ATM передавати чутливий до затримок аудіо- та відеотрафік з необхідним рівнем якості.

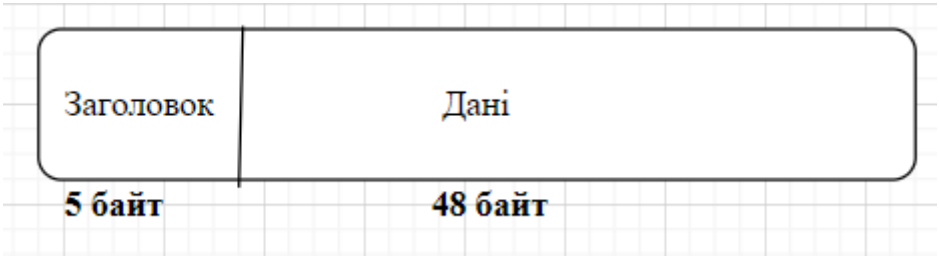

Рисунок 1.1 – Комірка ATM

Фізичний рівень

Фізичний рівень аналогічно фізичному рівню OSI визначає способи передачі в залежності від середовища. Стандарти ATM встановлюють, як біти будуть проходити через середу передачі і перетворюватись в комірку. На фізичному рівні ATM використовують цифрові канали передачі даних, з різними протоколами, а в якості ліній зв'язку використовуються: кабелі "кручена пара", екранована "кручена пара", оптоволоконний кабель.

Канальний рівень (рівень ATM + рівень адаптації)

Рівень ATM разом з рівнем адаптації приблизно еквівалентний другому рівню моделі OSI. Рівень ATM відповідає за передачу осередків через мережу ATM, використовуючи інформацію їх заголовків. Тема містить ідентифікатор віртуального каналу, який призначається з'єднанню при його встановленні і віддаляється при розриві з'єднання [2].

Переваги:

- однією з найважливіших переваг АТМ є забезпечення високої швидкості передачі інформації;
- АТМ усуває відмінності між локальними і глобальними мережами, перетворюючи їх в єдину інтегровану мережу;
- стандарти АТМ забезпечують передачу різнорідного трафіку (цифрових, голосових і мультимедійних даних) по одним і тим же системам і лініях зв'язку.

Недоліки:

- висока вартість обладнання, тому технології АТМ гальмується наявністю більш дешевих технологій;
- високі вимоги до якості ліній передачі даних [3].

1.2 Аналіз мов програмування

Мова програмування - це набір лексичних, синтаксичних і семантичних правил, які придумали люди, щоб створювати програми. Сучасні мови програмування поділяються на три типи:

- «Швидкі», які використовуються для оперативного створення додатків або їхніх прототипів.
- «Інфраструктурні», які допомагають оптимізувати або допрацьовувати окремі частини вже написаного додатки для того, щоб підвищити його продуктивність.
- Так звані системні мови програмування, використання яких дозволяє отримати в своє розпорядження повноцінний контроль над пам'яттю пристрою.

В 2020 році фахівці Інституту інженерів електротехніки та електроніки (Institute of Electrical and Electronics Engineers, IEEE) опублікували рейтинг найпопулярніших мов програмування. в першу п'ятірку за їхньою версією увійшли Python, Java, C, C ++ і JavaScript [4].

1. Python

Швидка і проста в розгортанні і використанні мова програмування Python. Це потужна скриптова мова з великою кількістю модулів і бібліотек.

Python є інтерпретованою мовою програмування. Містить невелику кількість ключових слів, гнучка і виразна.

Велика кількість стартапів використовують Python. Цією мовою написані такі популярні проекти, як YouTube, Instargam, Pinterest, SurveyMonkey.

Переваги

- Інтерпретатор Python є кросплатформеним.
- Мова має можливість розширюватися. Будь-який програміст має можливість вдосконалити мову. Вихідний код є відкритим і доступний до редагування або модифікацій.
- Наявність великої кількості модулів що підключаються до програми та забезпечують різні додаткові можливості. Такі модулі пишуться на C і на самому Python і можуть бути розроблені усіма досить кваліфікованими програмістами. Як приклад можна навести такі модулі:
  - Numerical Python – збільшує кількість математичних можливостей;
  - Tkinter – розробка додатків з застосуванням GUI (з англ. Graphical User Interface – графічний інтерфейс користувача) на основі Tk-інтерфейсу;
  - OpenGL - використання великої бібліотеки графічного моделювання дво- і тривимірних об'єктів Open Graphics Library фірми Silicon Graphics Inc.

Недоліки

- Щвидкість виконання Python-програми недостатньо висока. Це обумовлено її інтерпретованістю. Однак це з лишком окупається перевагами мови при написанні програм не дуже критичних до швидкості виконання [5].

Складність: низька.

2. *JavaScript*

JavaScript дуже багатий на фреймворки і зручні бібліотеки (Angular, React, jQuery, Vue) які зробили його ще популярнішим. Фактично JavaScript бере на

себе всі дії на боці клієнта, дозволяє управляти інтерфейсом і управляти серверної частиною проекту (Node.js) [6].

JavaScript це мова з великими перевагами, які роблять його найкращим вибором серед подібних йому, особливо в деяких варіантах застосування.

Переваги

- Не потрібен компілятор, тому що веб-браузер інтерпретує його в HTML;
- Його простіше вивчати, ніж інші мови програмування;
- Помилки простіше виявити, а значить і виправити;
- JS може прив'язуватися до спеціальних елементів сторінок або події на зразок натискання (click) або наведення миші (mouseover);
- JS працює в різних браузерах і на різних платформах;
- використовується для валідації вхідних даних і зниження необхідності ручної перевірки даних;
- робить сайт більш інтерактивним і привабливим для відвідувачів;
- швидше і легше, ніж інші мови програмування.

Недоліки

- Вразливий до експлойтів (шкідливий код, який використовує уразливості програмного продукту);
- Може бути використаний для запуску шкідливого коду на комп'ютері користувача;
- Інколи не підтримується деякими веб-браузерами;
- Фрагменти JS коду можуть бути громіздкими [7].

Складність: низька.

3. Java

Java був протестований програмістами в самих різних сферах: від кишенькових комп'ютерів до інтерактивного телебачення. Зараз він найбільш затребуваний в таких напрямках, як:

1) веб-розробка (масштабні бізнес-проекти);
2) програми для ПК (десктопний софт);
3) комп'ютерні ігри (наприклад, Minecraft);
4) додатки для мобільних пристроїв (ОС Android);
5) наукові дослідження і розробки;
6) промисловий програмінг[8].

Об'єктно-орієнтована – на Java все є об'єктом.

Переваги

Безпека – завдяки захищеній функції Java вона дозволяє розробляти системи без вірусів, без втручання. Методи автентифікації базуються на шифруванні з відкритим ключем.

Нейтральна до архітектури – Java-компілятор генерує нейтральний до архітектури формат об'єктного файлу, що робить скомпільований код виконуваним на багатьох процесорах на яких є система виконання Java.

Портативний – архітектурно нейтральний, не має аспектів специфікації, що залежать від реалізації. Це робить Java портативним.

Надійна – Java докладає зусиль для усунення ситуацій, схильних до помилок, роблячи основний упор на перевірку помилок під час компіляції та перевірку виконання.

Багатопотоковість – багатопотоковість мови Java надає можливість створювати програми, котрі зможуть виконати кілька поставлених задач одночасно.

Інтерпретованість – переклад байт-коду Java в машинні інструкції відбувається миттєво і ніде не зберігається. Процес розробки є більш швидким та аналітичним [9].

Недоліки

Головною проблемою Java є громіздкість. Це наслідок політики ООП (об`єктно-орієнтований підхід). Проте ООП необхідний для забезпечення безпеки проекту. В Java на відміну від C++ не має можливості створити функцію,

що не є методом якогось класу. В даному випадку громіздкість обумовлена неможливістю посилатися на функцію використовуючи покажчик, як в C++. В Java застосовуються інтерфейси, а вони досить багатослівні хоча і забезпечують безпеку програми [10].

Складність: помірна

На основі вище вказаних переваг мною було обрано мову JavaScript.

1.3 Вибір фреймворків та технологій для розробки веб-додатку

JavaScript-фреймворк – це фреймворк для додатків, написаний на мові програмування JavaScript. JavaScript-фреймворк відрізняється від Javascript бібліотеки потоком управління. Бібліотека містить функції для виклику батьківським кодом, а фреймворк спирається на структуру програми в цілому. Розробник не викликає код фреймворка, навпаки, фреймворк викликає і користується кодом розробника. Деякі Javascript-фреймворки використовують принцип MVC (model-view-controller), створений для відділення частин веб додатки, з метою поліпшення і розширюваності коду. Приклади Javascript-бібліотек: AngularJS, Ember.js, Vue.js [11]. Фреймворк схожий на інтерфейс прикладного програмування (API), хоча технічно фреймворк включає API. Як випливає з назви, фреймворк служить основою для програмування, тоді як API забезпечує доступ до елементів, що підтримуються фреймворком. Структура може також включати бібліотеки коду, компілятор та інші програми, що використовуються в процесі розробки програмного забезпечення.

1.3.1 React.js

Переваги ReactJS

1. Створення динамічних веб-додатків стає простішим

Створити динамічний веб-додаток спеціально із рядками HTML було складно, оскільки воно вимагає складного кодування, але React JS вирішив цю проблему та полегшив. Це забезпечує менше кодування та надає більше функціональних можливостей. Він використовує JSX (розширення JavaScript), який є певним синтаксисом, що дозволяє HTML-котировкам та синтаксису HTML-тегів відображати певні підкомпоненти. Він також підтримує побудову машиночитаних кодів.

2. Багаторазові компоненти

Веб-додаток ReactJS складається з декількох компонентів, і кожен компонент має свою логіку та елементи керування. Ці компоненти відповідають за виведення невеликого шматка HTML-коду, який можна багаторазово використовувати, і який можна використовувати повторно там, де вони вам потрібні. Код, що використовується багаторазово, полегшує розробку та обслуговування ваших програм. Ці компоненти можуть бути вкладені в інші компоненти, що дозволяє будувати складні програми з простих будівельних блоків. ReactJS використовує віртуальний механізм DOM для заповнення даних у HTML DOM. Віртуальний DOM працює швидко, оскільки змінює лише окремі елементи DOM, замість того, щоб кожен раз перезавантажувати повний DOM.

3. Підвищення продуктивності

ReactJS покращує продуктивність завдяки віртуальному DOM. DOM - це міжплатформенний API для програмування, який має справу з HTML, XML або XHTML. Більшість розробників зіткнулися з проблемою під час оновлення DOM, що уповільнило продуктивність програми. ReactJS вирішив цю проблему, представивши віртуальний DOM. Віртуальний DOM React повністю існує в пам'яті і є поданням DOM веб-браузера. Завдяки цьому, коли ми пишемо компонент React, ми не писали безпосередньо в DOM.

4. Підтримка зручних інструментів

React JS також завоював популярність завдяки наявності зручного набору інструментів. Ці інструменти роблять завдання розробників зрозумілими та простішими. Інструменти розробника React були розроблені як розширення для Chrome та Firefox і дозволяють перевіряти ієрархію компонентів React у віртуальній DOM. Це також дозволяє вибрати конкретні компоненти та вивчити та відредагувати їх поточний реквізит та стан.

5. Перевага наявності бібліотеки JavaScript

Сьогодні ReactJS вибирає більшість веб-розробників. Це тому, що він пропонує дуже багату бібліотеку JavaScript. Бібліотека JavaScript забезпечує більшу гнучкість для веб-розробників у виборі шляху, який вони хочуть.

6. Сфера тестування кодів

Програми ReactJS надзвичайно легко протестувати. Він пропонує область, де розробник може тестувати та налагоджувати свої коди за допомогою власних інструментів.

Недоліки ReactJS

1. Високі темпи розвитку

Високі темпи розвитку мають як переваги, так і недоліки. У разі недоліків, оскільки навколишнє середовище постійно змінюється так швидко, деякі розробники не почуваються комфортно переучувати нові способи регулярної роботи. Їм може бути важко прийняти всі ці зміни з усіма постійними оновленнями.

2. Скудна документація

Це ще один мінус, який є загальним для постійного оновлення технологій. Реагуйте на оновлення та прискорення технологій настільки швидко, що немає часу робити належну документацію. Щоб подолати це, розробники самостійно пишуть інструкції, розвиваючи нові випуски та інструменти у своїх поточних проектах.

3. JSX як бар'єр

ReactJS використовує JSX. Це розширення синтаксису, яке дозволяє змішувати HTML із JavaScript. Цей підхід має свої переваги, але деякі члени спільноти розробників розглядають JSX як бар'єр, особливо для нових розробників [12].

1.3.2 Electron.js

Electron – це фреймворк, який дозволяє створювати настільні програми за допомогою веб-технологій (HTML, CSS, JS).

Переваги

1. HTML, CSS, JS

Можливість створити десктопний додаток використовуючи HTML, CSS та JavaScript. Усі ці мови є легкими в навчанні та в роботі.

2. Електронні програми схожі на веб-програми

Додатки на Electron поводяться як веб-програми. Їх відрізняє те, що веб-програми можуть завантажувати файли лише у файлову систему комп'ютера, а Electron Apps може отримати доступ до файлової системи, а також читати та записувати дані.

3. Chromium

Electron використовує движок Chromium для візуалізації інтерфейсу користувача. Це означає, що ви можете отримати від цього кілька переваг, такі як Інструменти розробника (DevTools), доступ до сховища тощо.

Недоліки

1. Великий розмір

Програми Electron працюють на Chromium, це означає, що кожен додаток Electron постачається зі своєю власною версією Chromium. Гірше того, що Chromium складається з 20 мільйонів рядків коду, що майже відповідає цілій операційній системі. Тож це як встановлення операційної системи поверх іншої для запуску одного додатка. Наприклад, простому додатку "Hello World" на Electron знадобиться більше 100 Мб місця.

2. Захист коду

Файли не зашифровані, що означає, що кожен може отримати робочу копію коду.

3. Крос-платформні збірки

Якщо ваша програма має власні залежності, її можна скомпілювати лише на цільовій платформі. Наприклад, підпис коду macOS працює лише на MacOS [13].

1.3.3 Node.js і Express.js

Node.js не є фреймворком чи бібліотекою. Це середовище виконання, заснованим на механізмі JavaScript V8 Chrome. Це популярна платформа для створення повномасштабних веб-додатків Node.js, гібридних додатків, REST API, настільних програм та рішень IoT.

Express.js – це фреймворк Node.js, який по суті і є стандартним каркасом для Node.js. Він розроблений для створення веб-додатків та API.

Переваги

- Динамічний стек технологій

JavaScript є надзвичайно популярною мовою програмування. У той же час Node.js розглядається як незалежна технологія. Він охоплює всі переваги JavaScript, але на серверній частині, включаючи:

- Це широко прийнята мова
- Велика бібліотека та інструменти
- Ефективність завдяки використанню однієї мови
- Висока швидкість і продуктивність
- Високошвидкісна та подійна модель

Взявши асинхронну та неблокувальну природу від свого батьківського коду, JavaScript, Node.js може виконувати незначні завдання у фоновому режимі, не перешкоджаючи основному потоку. Ще однією причиною його швидкої обробки є двигун V8, який є найшвидшим двигуном JS на сьогодні.

Node.js покладається на модель на основі подій. Node.js передбачає оновлення даних у режимі реального часу, що є критичним для програм обміну повідомленнями, онлайн-ігор та відеочат. Використання однієї мови на передній та задній панелях - найкращий спосіб досягти швидкої синхронізації.

Також Node.js пропонує неблокуючі системи вводу-виводу, які дозволяють одночасно обробляти паралельні запити. Коли запити вишикуються, вони обробляються ефективно та без затримок. Навпаки асинхронній обробці запитів (що є основною перевагою Node.js), синхронна обробка обробляє запит по одному. Це означає, що запит не може бути оброблений, якщо в процесі є інший. Асинхронна обробка Node.js дозволяє обробляти запити без блокування потоку.

Недоліки

Зниження продуктивності під час складних обчислювальних завдань

Node.js є однопоточним та керується подіями, і тому він не підходить для важких обчислювальних завдань. Отримуючи масивне обчислювальне завдання, він використовує потужність центрального процесора для повної реалізації цього завдання, залишаючи інші завдання в черзі. Це означає уповільнення всього потоку подій, що перешкоджає інтерфейсу. Для вирішення цієї проблеми були введені "робочі потоки", але це рішення не є абсолютно ефективним для обробки обчислювальних завдань, пов'язаних з процесором.

Відсутність бібліотечної підтримки

Хоча бібліотека npm здається багатою, якість багатьох пакетів залишає бажати кращого. Багато з них не мають належної документації. Оскільки це система з відкритим кодом, професійний моніторинг тут мізерний, тому багато пакетів не відповідають стандартам кодування.

Нестача розробників

Незважаючи на те, що це частина концепції розробки повного стеку, не так багато розробників JavaScript готові вивчати NodeJS. Це тому, що це вимагає великих зусиль та попереднього досвіду в розробці програмного забезпечення [14].

1.4 Вибір системи управління базою даних

Основною метою веб-додатку є генерація набору команд для подальшого налаштування реальних роутерів або роутерів в емуляторі мереж GNS3. Щоб генерувати команди потрібен скрипт, який повинен десь зберігатись. Для цього потрібна база даних.

Що таке база даних?

База даних – це місце для зберігання даних. Використовуються в переважаючій більшості проектів. Управління базами відбувається за допомогою СУБД (система управління базами даних). СУБД діляться на SQL та NoSQL, реляційна та нереляційна СУБД відповдіно.

SQL або мова структурованих запитів використовується для роботи з даними, що зберігаються в базі даних. SQL залежить від реляційної

алгебри і кортежних реляційного числення [15].

За допомогою стандартних команд SQL, таких як "select", "insert", "create", "update", "delete" та "drop", можна виконати майже все, що потрібно зробити з базою даних [16].

Для бази даних зазвичай потрібна комплексна програма програмного забезпечення для баз даних, відома як система управління базами даних (СУБД). СУБД – це, інтерфейс між базою даних та її користувачами або програмами, які використовують бази даних в своїй архітектурі.

До популярних на сьогодні СУБД відносяться: MySQL, PostgreSQL Microsoft SQL Server та Oracle Database.

MySQL є одним з найпопулярніших. Велика кількість розробників покладаються саме на цю СУБД, тому що вона:

1) Гнучка та проста у використанні

Є можливість змінити вихідний код (MySQL написана на C та C++), Процес встановлення відносно простий і не триває більше 30 хвилин.

2) Має простий синтаксис та легку складність

Структура та стиль MySQL дуже прості. Розробники навіть вважають MySQL базою даних з людиноподібною мовою. Крім того, MySQL простий у використанні. Наприклад, більшість завдань можна виконати прямо в командному рядку, зменшуючи кроки розробки [17].

3) Сумісна з хмарами

MySQL підтримується найпопулярнішими хмарними провайдерами. Він доступний на таких провідних платформах, як Amazon, Microsoft та інші. Це робить MySQL ще більш привабливим і надає компаніям, що використовують його, можливість для зростання.

4) Захищена

Завдяки системі привілеїв доступу та керуванню обліковими записами користувачів MySQL встановлює високий рівень безпеки. Доступні перевірка на основі хоста та шифрування пароля.

1.5 Постановка задачі

Провівши аналіз літератури, метою роботи можна вважати: веб-додаток, який дозволить автоматично налаштувати інтерфейси роутерів з застосуванням технології ATM. Він повинен забезпечити зручне копіювання згенерованих команд на реальне обладнання Cisco або на обладнання Cisco в емуляторі мереж GNS3.

Інтерфейс додатку має бути інтуїтивно зрозумілим та таким, що допоможе навіть не досвідченим користувачам налаштувати інтерфейси роутерів Cisco.

Для створення графічного інтерфейсу необхідно створити веб-додаток, де можна буде ввести вхідні дані та отримати перелік команд для налаштування, які можна скопіювати та внести в симулятор GNS3. Як результат роутер і мережа будуть зконфігуровані.

Постановка задачі:

1. Конфігурація мережі ATM на базі роутерів Cisco та симулятора GNS3.

2. Розробка графічного інтерфейсу налаштування мережі ATM.

3. Тестування розробленого веб-додатку в емуляторі GNS3 та на реальному обладнанні Cisco.

1 МОДЕЛЮВАННЯ МЕРЕЖІ З ЗАСТОСУВАННЯМ ТЕХНОЛОГІЇ АТМ В ЕМУЛЯТОРІ GNS3 ТА РОУТЕРАХ CISCO

1.1 Конфігурація мережі ATM в мережевому емуляторі GNS3

Існує безліч симуляторів і емуляторів, але найбільш популярні Cisco Packet Tracer, Boson NetSim, GNS3, VIRL, EVE-NG. Багато з цих інструментів також можуть бути використані для тестування мережевих технологій і для розгортання мереж в реальному світі.

Симулятор – це частина програмного забезпечення, яка, імітує топологію мережі, що складається з одного або декількох мережевих пристроїв. Модельовані мережеві пристрої не є реальними пристроями і не можуть передавати «живий» мережевий трафік.

Мережеві пристрої в симуляторі обмежені командами і функціями, запрограмованими в них. З цієї причини багато додаткові можливості, які присутні на реальних мережевих пристроях, відсутні в імітованих аналогах.

Ключова перевага симуляторів полягає в тому, що вони, як правило, не дуже вимогливі до ресурсів - програмне забезпечення симулятора може працювати практично на будь-якому сучасному комп'ютері.

Емулятор – це частина програмного забезпечення, яка працює і з'єднує пристрої віртуальної мережі разом. Емулятори віртуалізують реальні мережеві пристрої, які пропонують більш «просунутий» набір функцій в порівнянні з пристроями, представленими в симуляторах. Поведінка, що демонструється віртуальними мережевими пристроями, в більшій мірі відображає поведінку реальних фізичних пристроїв в реальному світі.

Графічний мережевий емулятор (GNS3) - це безкоштовний інтерфейс клієнт-сервер з відкритим вихідним кодом для емуляції і віртуалізації мережі. Це платформа на основі Python, яка в основному використовує програмне

забезпечення під назвою Dynamips для емуляції програмного і апаратного забезпечення Cisco.

GNS3 підтримує великий обсяг віртуальних мережевих пристроїв від різних постачальників мережевого обладнання за рахунок використання пристроїв, які є простими в імпорті шаблонами.

Оскільки GNS3 є клієнт-серверним додатком, рекомендується розгортати віртуальну машину GNS3 VM (Virtual Machine) в якості сервера. Потім можна встановити клієнтську програму GNS3 на локальному комп'ютері і підключитися до сервера віртуальної машини GNS3. Після установки можна створювати мережеві топології за допомогою клієнтської частини ПО, які будуть виконуватися на сервері.

GNS3 має ряд переваг в якості безкоштовного емулятора мережі з відкритим вихідним кодом. GNS3 має співтовариство розробників і користувачів, головна перевага якого - позитивний зворотний зв'язок, створеної групою однодумців, які хочуть допомогти іншим вчитися, працювати. Емулятор має хорошу документацію з ілюстраціями для початківців користувачів або при необхідності керівництва по розширеній конфігурації. У GNS3 кожне віртуальне мережеве пристрій можна запускати і зупиняти незалежно від інших віртуальних пристроїв.

Емулятор не тільки підтримує Ethernet-з'єднання між мережевими пристроями, але і дозволяє встановлювати послідовні з'єднання між пристроями, що підтримують відповідні модулі.

Програмне забезпечення GNS3 не має вбудованих образів мережевих операційних систем. Тому для емуляції будь-яких маршрутизаторів або комутаторів Cisco необхідно спочатку мати існуючий образ програмного забезпечення Cisco IOS, сумісний з GNS3.

Головним недоліком GNS3 є факт необхідності створювати свої власні програмні образи мережевих пристроїв для емуляції. Це не вина GNS3. Інтегрування образів програмного забезпечення Cisco IOS в GNS3 було б

незаконним. Наявність цих образів є важливим фактором, і це необхідно враховувати перед розгортанням GNS3 для особистого або комерційного використання [18].

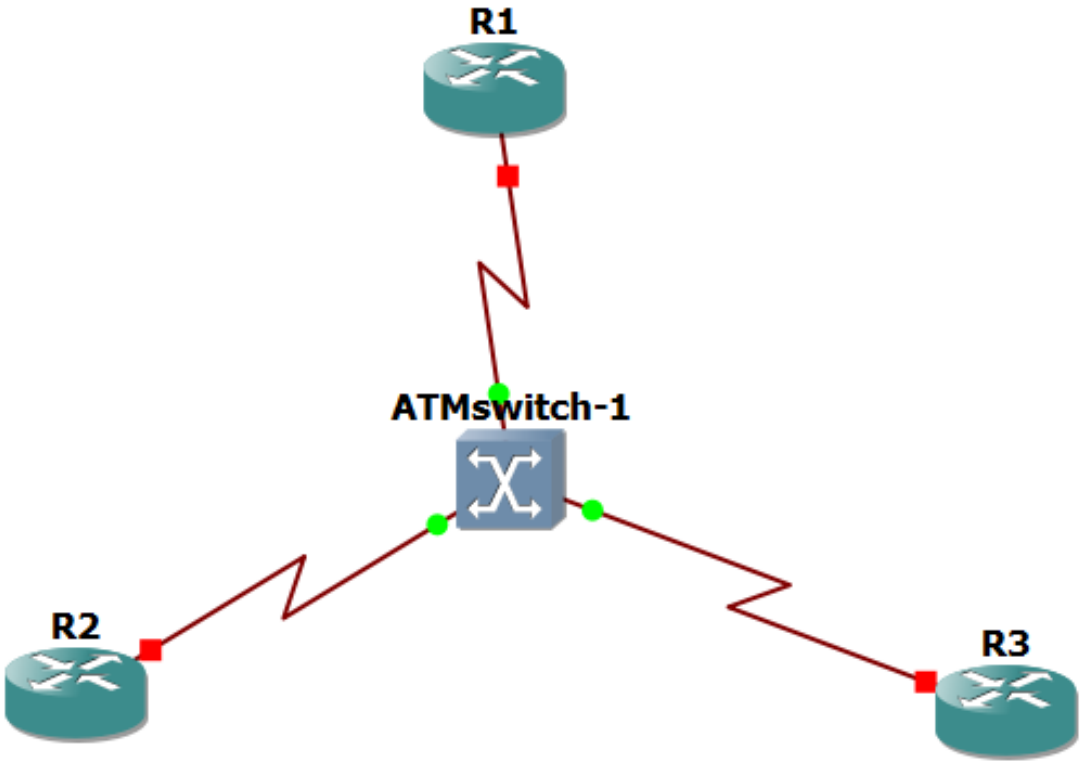

Рисунок 2.1 – Мережа ATM в емуляторі GNS3

Спочатку була змодельована мережа ATM в емуляторі GNS3. Потім налаштований ATM-світч (рис. 2.2). Далі були налаштовані інтерфейси роутерів. Для моделювання були використані роутери Cisco 7200 зі слотом PA-A1 (рис. 2.3).

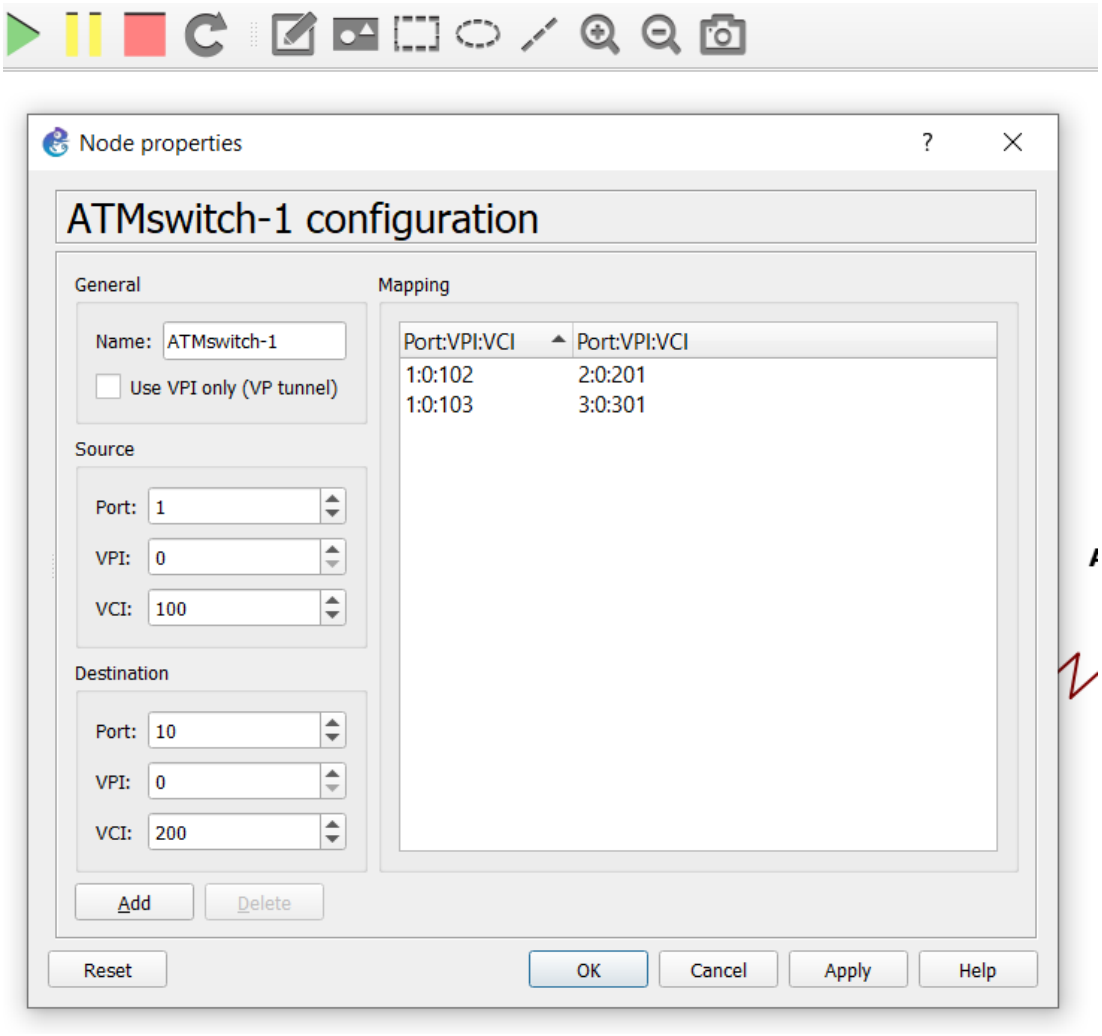

Рисунок 2.2 – Конфігурація комутатора ATM

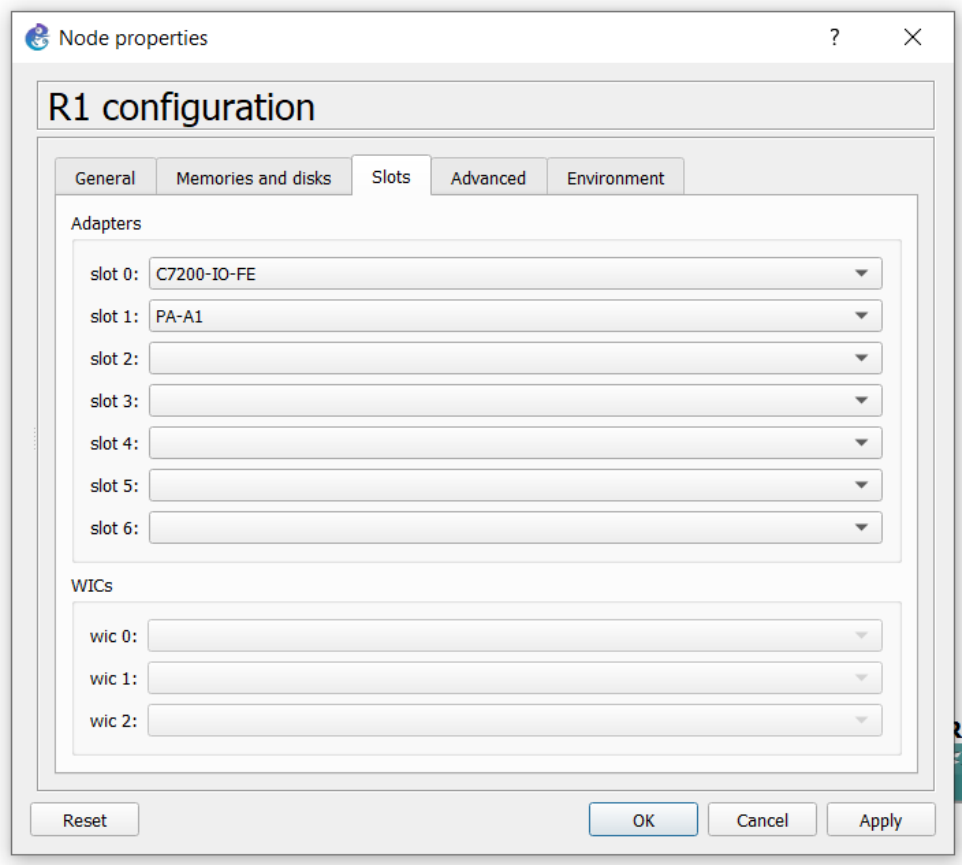

Рисунок 2.3 – Конфігурація роутера

Роутер R1 був підключений до інших шляхом точка-багатоточка, або point-to-multipoint. Всі інші роутери підключені до R1 шляхом точка-точка, або point-to-point.

Конфігурація роутеру R1 підключеного шляхом точка-багатоточка:

conf t

int atm1/0

no sh

int atm1/0.1 multipoint

ip add 192.168.1.1 255.255.255.0

pvc 0/102

encap aal5snap

protocol ip 192.168.1.2 broad

pvc 0/103

```
encap aal5snap
protocol ip 192.168.1.3 broad
pvc 0/104
encap aal5snap
protocol ip 192.168.1.4 broad
end
```

Конфігурація роутеру R2 підключеного шляхом точка-точка:

```
conf t
int atm1/0
no sh
int atm1/0.1 point-to-point
ip add 192.168.1.2 255.255.255.0
pvc 0/201
encap aal5snap
end
```

Команда encapsulation aal5snap вказує, що для даного PVC буде використовуватися рівень адаптації5 (AAL5), а в осередку ATM будуть міститися пакети IP (Subnetwork Access Protocol - SNAP).

Рисунок 2.4 – Маршрути на роутері

1.2 Конфігурація мережі ATM на роутерах Cisco

Cisco Systems, Inc. — американська транснаціональна корпорація, яка є найбільшим у світі виробником мережевого обладнання, наприклад:

- Ethernet комутатори
- Cisco Nexus — комутатори дата-центрів
- Маршрутизатори
- Продукти для IP-телефонії, такі як IP PBX, VoIP-шлюзи
- ATM -комутатори
- Кабельні модеми
- DSL-устаткування
- Універсальні шлюзи і шлюзи віддаленого доступу

- Комутатори мереж зберігання даних (SAN, Storage Area Network)
- Програмне забезпечення управління мережею

В основному компанія працює в таких напрямках:

- Магістральна маршрутизація;
- Рішення для спільної роботи;
- Віртуалізація центрів обробки даних та хмарні обчислення;
- Відеотехнології;
- Комутація [19].

Для моделювання мережі були застосовані роутери Cisco 7200 з додатковим слотом PA-A1 та комутатор ATM. Були налаштовані інтерфейси роутерів та конфігурація портів комутатора. Проаналізовано маршрутизацію.

Для прискорення моделювання та подальшого налаштування мережі було вирішено розробити веб-додаток, який генерує перелік команд для кожного окремого роутера в мережі в залежності в їхньої кількості.

1.3 Розробка веб-додатку на мові програмування JavaScript

Додаток був розроблений за допомогою мови JavaScript та його фреймворків.

JavaScript - це текстова мова програмування, яка застосовується як на стороні клієнта, так і на стороні сервера. Це надає можливість зробити веб-сторінки інтерактивними. Якщо HTML і CSS - це мови, які керують структурою та стилем веб-сторінок, JavaScript надає веб-сторінкам інтерактивні елементи. Застосування до проектів JavaScript покращує взаємодію з веб-сторінкою, перетворюючи її зі статичної сторінки в інтерактивну. Підсумовуючи, JavaScript додає поведінку веб-сторінок.

Основний напрямок застосування JavaScript це створення веб-додатків, однак його успішно використовують для розробки програмного забезпечення за межами веб.

Основні моменти використання JavaScript:

1)Додавання інтерактивності до сторінки для взаємодії користувача з сторінками. JS працює в тандемі із HTML та CSS, а тому дозволяє створювати нові елементи на сторінці та змінювати їхні стилі.

2) Створення веб- і мобільних додатків. JavaScript має велику кількість різноманітних фреймворків і бібліотек для різних цілей, які розробники можуть використовувати для реалізації своїх проектів. React, React Native, Angular та Vue входят в число найпопулярніших.

3) Створення веб-серверів та розробка серверних додатків. Окрім створення клієнтської частини проекту JavaScript може бути корисним і для розробки серверної частини. Робиться це за допомогою фреймворку Node.js. Таким чином веб-додаток може бути написаний на одній мові JavaScript [20].

2 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ГРАФІЧНОГО ІНТЕРФЕЙСУ НАЛАШТУВАННЯ МЕРЕЖІ ATM

2.1 Розробка графічного інтерфейсу налаштування мережі ATM

Мережа ATM була побудована в емуляторі мереж GNS3, де були налаштовані роутери та ATM -світч. Після налаштувань можна зробити висновок, що великим недоліком GNS3 є відсутність графічного інтерфейсу для конфігурації роутерів, як, наприклад, в Cisco Packet Tracer. Налаштування роутерів через командну строку є достатньо довгою та монотонною роботою, тому розробка графічного інтерфейсу є актуальною задачею.

Проект був розроблений з застосуваннм мови JavaScript. Інтерфейс додатку є інтуїтивно зрозумілим та зручним (рис. 3.1, рис. 3.2).

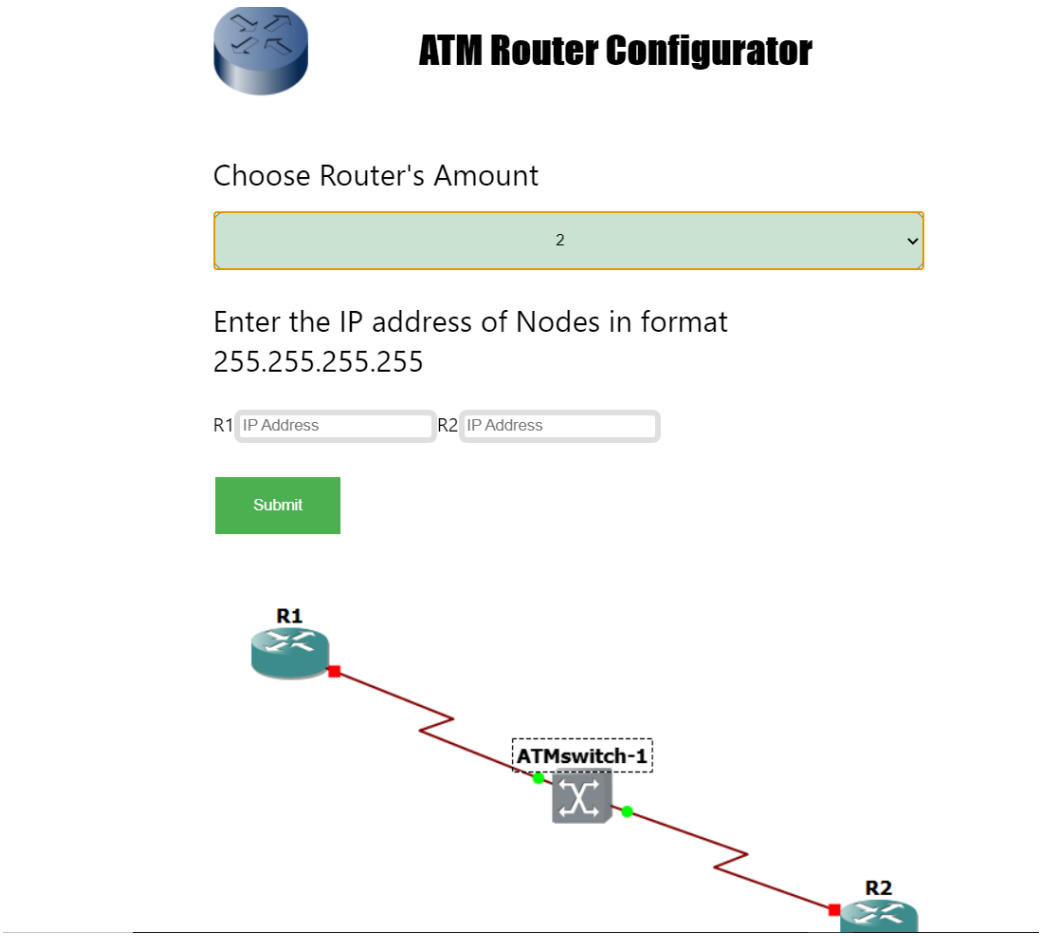

Рисунок 3.1 – Інтерфейс веб-додатку (частина 1)

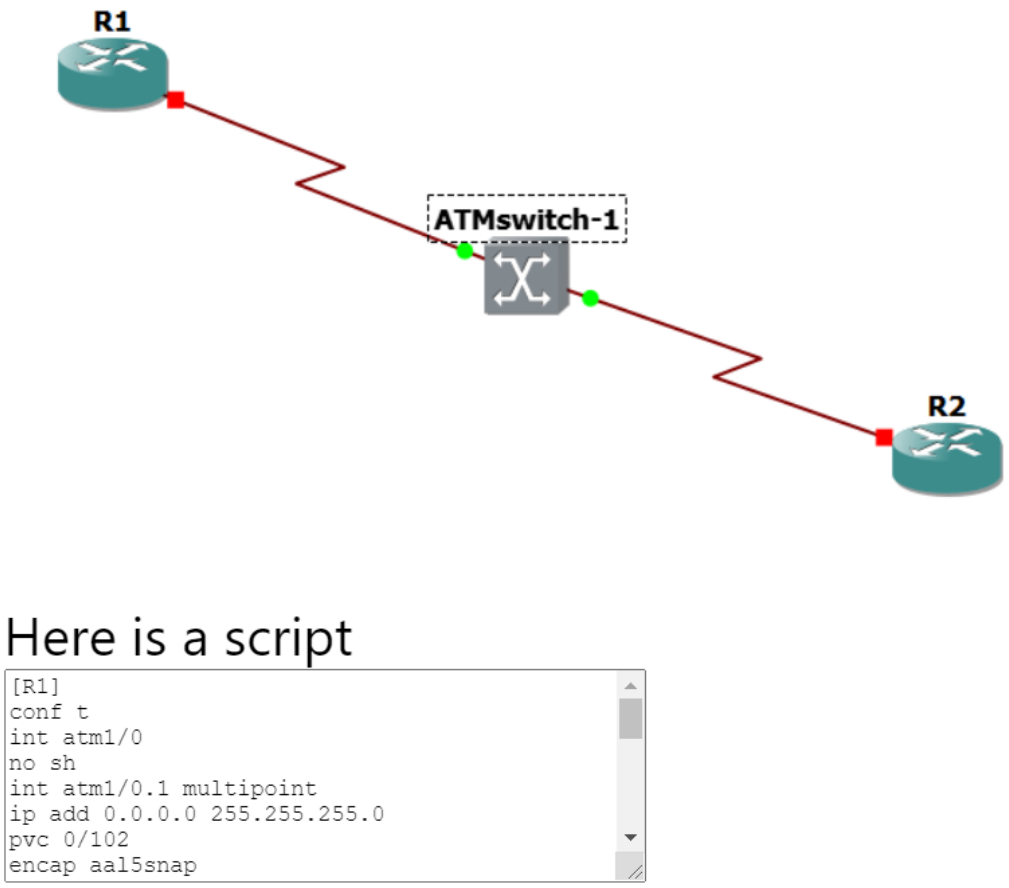

Рисунок 3.2 – Інтерфейс веб-додатку (частина 2)

Після відкриття Electron-додатку можна побачити:

- мережу ATM, що складається з двох роутерів та ATM -світча;
- випадаючий список, в якому можна обрати кількість роутерів;
- текстові поля для введення IP-адрес роутерів;
- кнопку «Submit»;
- текстову зону зі згенерованим переліком команд для налаштування роутерів.

Текстові поля для введення IP-адрес провалідовані під формат 255.255.255.255, при введенні даних, що не відповідають даному формату, поле змінить колір на червоний, а після натискання кнопки «Submit» з’явиться попередження про невірно введені дані (рис 3.3).

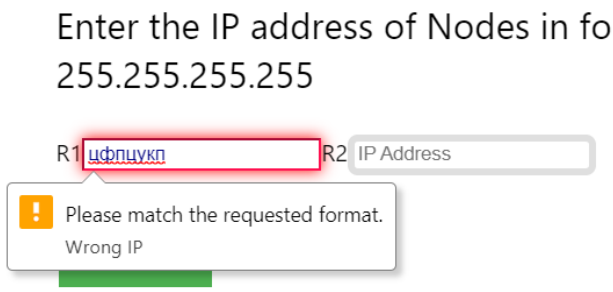

Рисунок 3.3 – Демонстрація роботи валідації текстового поля

Для успішної генерації переліку команд налаштування роутерів, користувач має обрати в випадаючому списку кількість вузлів в мережі (від 2 до 4), потім ввести коректні IP-адреси для кожного роутера, натиснути кнопку «Submit» та звернутися до текстової зони під заголовком «Here is a script». На рисунках нижче (рис 3.4, рис. 3.5) зображено приклад налаштування мережі ATM, що складається з трьох роутерів, де кожен з них має власну IP-адресу:

- R1 – 192.168.1.1
- R2 – 192.168.1.2
- R3 – 192.168.1.3

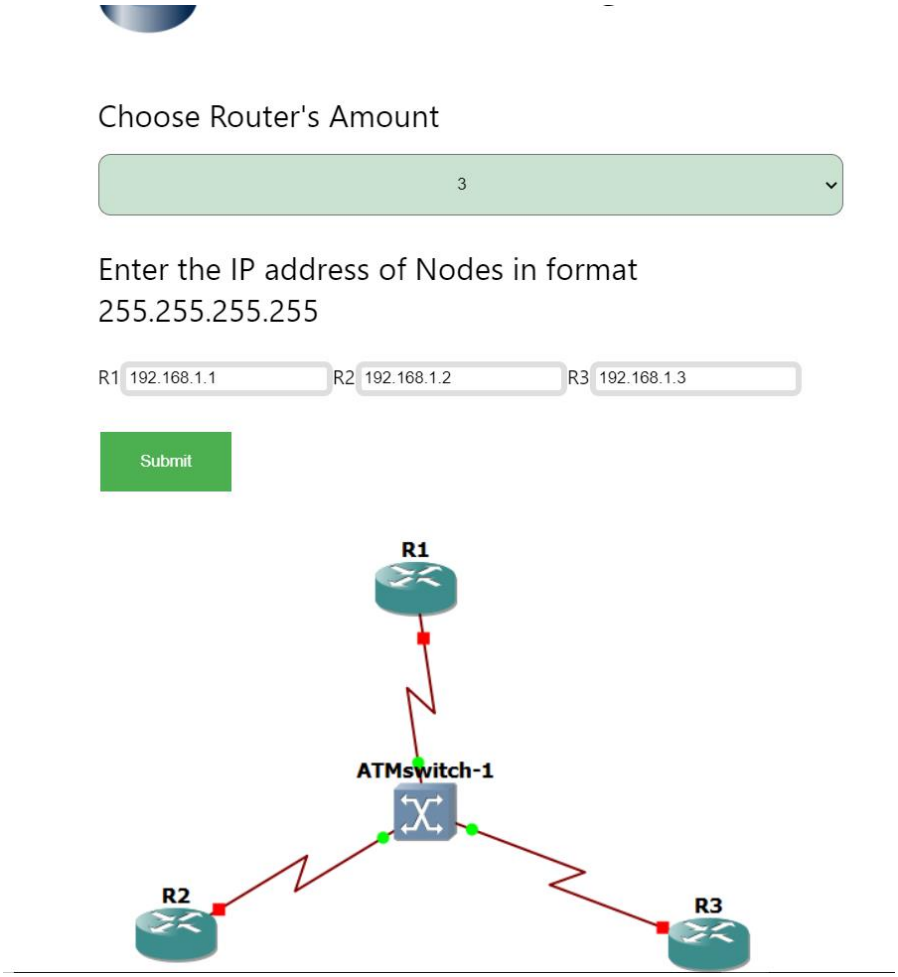

Рисунок 3.4 – Приклад генерації переліку команд в мережі ATM, що складається з трьох роутерів (частина 1)

Here is a script

```
[R1]
conf t
int atm1/0
no sh
int atm1/0.1 multipoint
ip add 192.168.1.1 255.255.255.0
pvc 0/102
encap aal5snap
protocol ip 192.168.1.2 broad
pvc 0/103
encap aal5snap
protocol ip 192.168.1.3 broad
pvc 0/104
encap aal5snap
protocol ip  broad
end

[R2]
conf t
int atm1/0
no sh
int atm1/0.1 point-to-point
ip add 192.168.1.2 255.255.255.0
pvc 0/201
encap aal5snap
end

 [R3]
conf t
int atm1/0
no sh
int atm1/0.1 point-to-point
ip add 192.168.1.3 255.255.255.0
pvc 0/301
encap aal5snap
end
```

Рисунок 3.5 – Приклад генерації переліку команд в мережі ATM, що складається з трьох роутерів (частина 2)

2.2 Тестування веб-додатку в емуляторі GNS3 та на реальному обладнанні Cisco

Для того щоб переконатися в працездатності розробленого веб-додатку потрібно провести його тестування в емуляторі мереж GNS3 та на реальних пристроях Cisco.

Перевіримо правильність згенерованих додатком команд на прикладі мережі ATM з трьома вузлами. В розробленому веб-додатку обираєму в випадаючому списку 3, що відповідає мережі ATM з трьома роутерами, вводимо IP-адреси. Для R1 – 192.168.100.1, для R2 – 192.168.100.2 і для R3 – 192.168.100.3 (рис. 3.6).

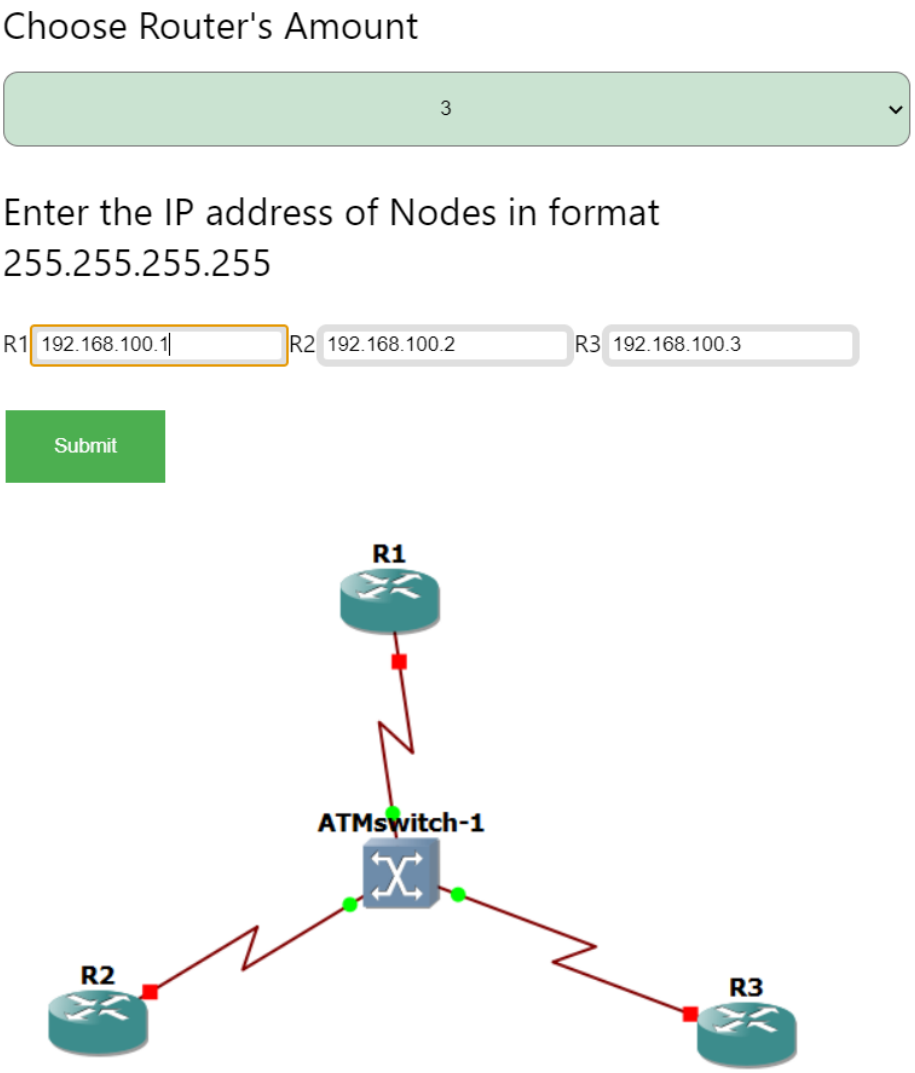

Рисунок 3.6 – Обрана кількість роутерів в мережі та заповнені текстові поля для введення IP

Потім натискаємо кнопку «Submit» і переходимо до текстової зони зі згенерованим переліком команд (рис. 3.7).

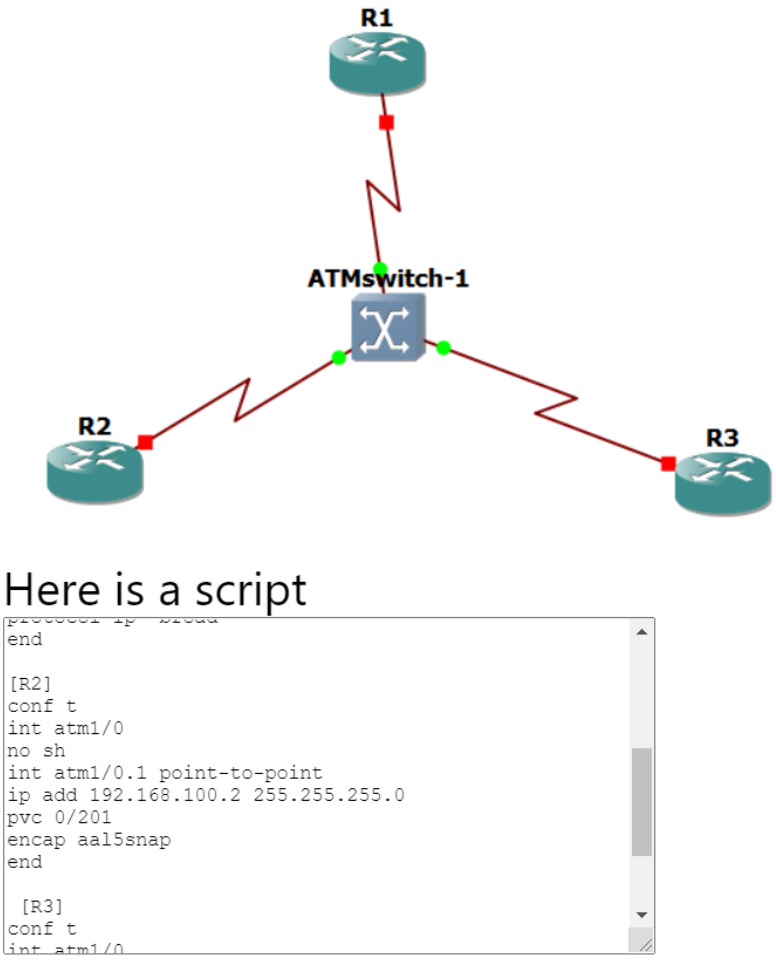

Рисунок 3.7 – Згенерований перелік команд для налаштування роутерів в мережі ATM

Копіюємо скрипт для кожного роутера окремо (назва роутеру, для якого призначені команди написані над скриптом) комбінацією клавіш Ctrl+C (Windows) або Command+C (macOS) та вставляємо в CLI (з англ. Command Line Interface – Інтерфейс командної строки) відповідного роутеру(рис. 3.8).

Рисунок 3.8 – CLI роутера R1

Аналогічно було налаштовано роутери R2 та R3. Команда «show run» дозволить переконатись в тому, що команди були виконані (рис. 3.9).

Рисунок 3.9 – Результат виконання команди «show run» в CLI роутера R1

Згідно з рисунку 3.9 (рис. 3.9), налаштування роутера було виконане успішно. Для перегляду маршрутів буде доречно використати команду «show ip route» (рис. 3.10).

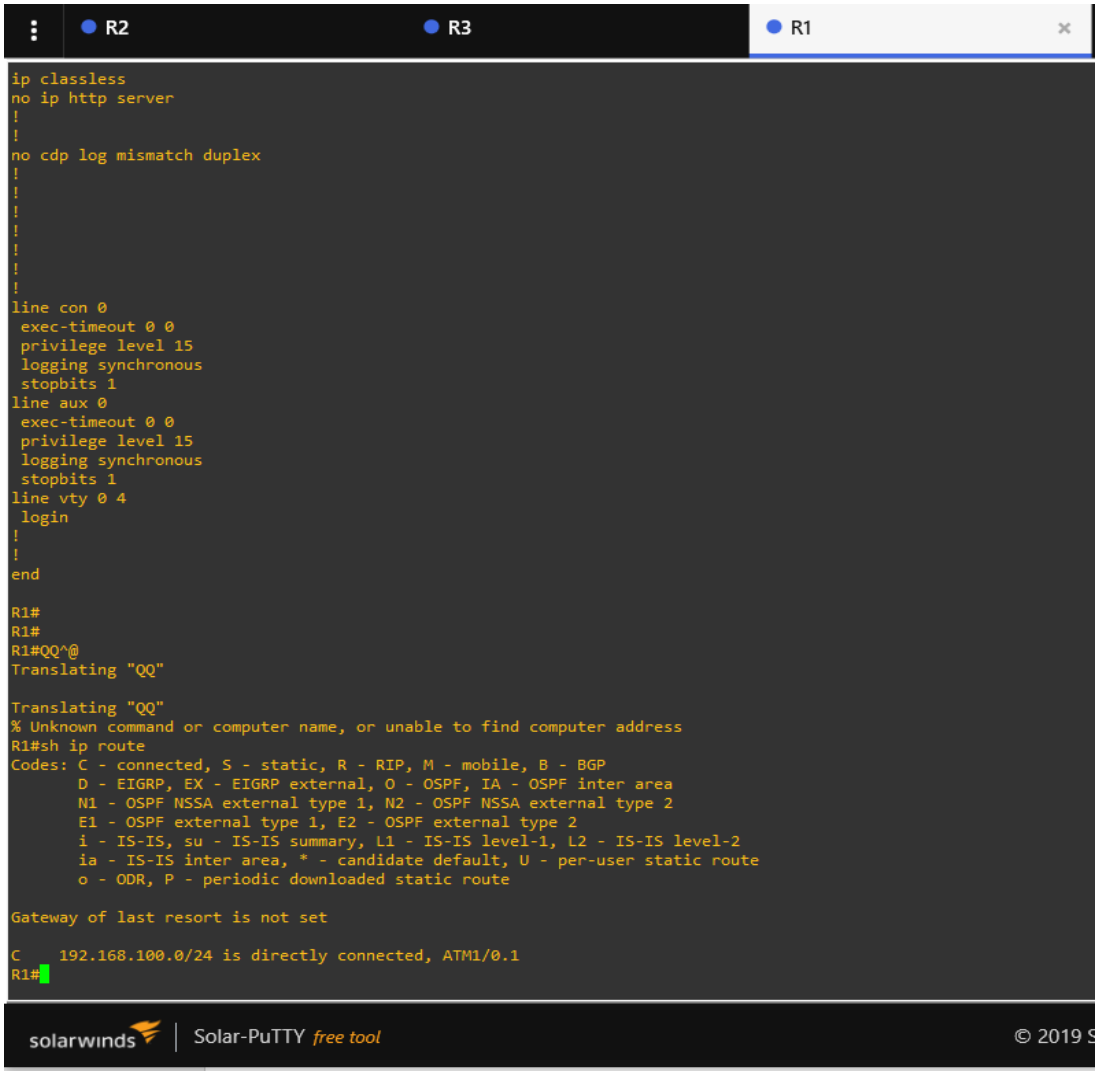

Рисунок 3.10 – Таблиця маршрутизації роутера R1 з застосуванням технології ATM

Тепер потрібно переконатись, що мережа ATM працює і роутери R1, R2 та R3 «знайомі» один з одним. Щоб перевірити цілісність і якість з’єднання в мережі використаємо команду «ping». Відправимо ICMP-повідомлення з роутера R1 на ATM -інтерфейси роутерів R2 та R3 (рис. 3.11).

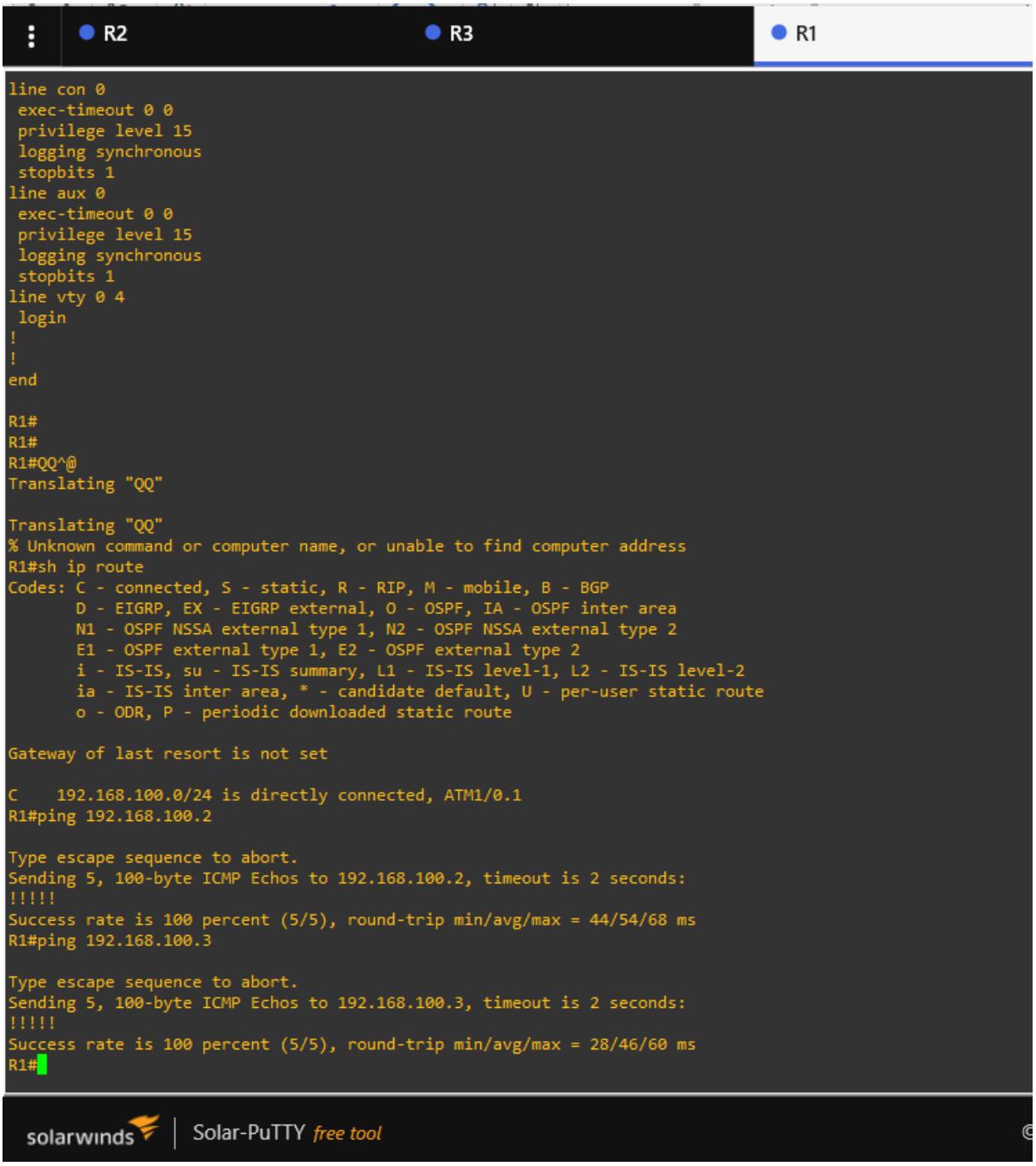

Рисунок 3.11 – Перевірка якості та цілісності з'єднання за допомогою команди «ping» в CLI роутера R1

Згідно результату виконання команди «ping» перевірка працездатності мережі була пройдена успішно (рис. 3.11).

ВИСНОВОК

Було переглянуто кілька популярних симуляторів та емуляторів мереж: GNS3 та Cisco Packet Tracer. GNS3 виявився занадто вимогливим до ресурсів комп'ютера, проте цей емулятор має можливість змоделювати ATM мережу. В Cisco Packet Tracer такої можливості не виявилося.

В процесі роботи були детально порівняні технології Ethernet і ATM, переглянуті їхні переваги і недоліки. Ethernet основано на кадрах, ATM – на комірках. Ethernet більше підходить для локальних мереж, а ATM для глобальних.

В рамках роботи був розроблений веб-додаток. Зручно і швидко налаштувати мережу ATM дозволяє графічний інтерфейс додатку. Для успішної генерації переліку команд достатньо обрати кількість роутерів та ввести бажані IP-адреси в відповідні текстові поля. Потім вже для налаштування реального обладнання або змодельованої в емуляторі мережі скопіювати згенерований скрипт та вставити його в CLI відповідного роутера.

Були проаналізовані найпопулярніші мови програмування для веб-розробки. Серед них була обрана мова програмування JavaScript. JavaScript позиціонує себе як оптимальна мова для роботи з інтерфейсами для браузерів і веб-сторінок. Завдяки багатому набору бібліотек він також забезпечив нові можливості для візуалізації.

Серед багаточисельних фреймворків мови JavaScript, було вирішено використати React.js для розробки користувацького інтерфейсу і Node.js+Express.js для серверної частини додатку. Використання React.js та Node.js дозволяє створити проект з клієнт-серверною архітектурою за допомогою концепції MVC (Model-View-Controller). Використання фреймворку Electron.js дозволило зробити веб-додаток настільним.

Для зберігання та читання скрипту майбутніх команд для конфігурації роутерів використана одна з найпопулярніших реляційних СУБД MySQL.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Основные принципы технологии АТМ [Електронний ресурс] – 2020. – Режим доступу: http://kunegin.com/ref3/atm6/01.htm – Назва з екрану.
2. Буассо М. Введение в технологию АТМ / М. Буассо, М. Деманж, Ж. Мунье., 1997. – 64 с.
3. Глобальные сети с коммутацией пакетов.Технология АТМ - технология передачи ячеек или технология трансляции ячеек [Електронний ресурс] – 2020. – Режим доступу: https://www.lessons-tva.info/edu/telecom-glob/m2t2_3glob.html – Назва з екрану.
4. Pick your battles: Choosing and Learning the “right” Programming Language [Електронний ресурс] – 2020. – Режим доступу: https://hackernoon.com/pick-your-battles-choosing-and-learning-the-right-programming-language-731973698385 – Назва з екрану.
5. Краткий обзор языка Python [Електронний ресурс] – 2020. – Режим доступу: https://www.helloworld.ru/texts/comp/lang/python/python2/index.htm#2 – Назва з екрану.
6. The 9 Best Programming Languages to Learn in 2021 [Електронний ресурс] – 2020. – Режим доступу: https://www.fullstackacademy.com/blog/nine-best-programming-languages-to-learn – Назва з екрану.
7. Что Такое JavaScript?[Електронний ресурс] – 2020. – Режим доступу:https://www.hostinger.ru/rukovodstva/chto-takoe-javascript/ – Назва з екрану.
8. Преимущества языка программирования Java [Електронний ресурс] – 2020. – Режим доступу: https://vc.ru/dev/144873-preimushchestva-yazyka-programmirovaniya-java – Назва з екрану.

9. Java - Overview [Електронний ресурс] – 2020. – Режим доступу:https://www.tutorialspoint.com/java/java_overview.htm – Назва з екрану.

10. Java достоинства и недостатки [Електронний ресурс] – 2020. – Режим доступу: https://4systems.ru/inf/java-dostoinstva-i-nedostatki/– Назва з екрану.

11. Difference Between Library and Framework [Електронний ресурс] – 2020. – Режим доступу: https://www.c-sharpcorner.com/uploadfile/a85b23/framework-vs-library/ – Назва з екрану.

12. Pros and Cons of ReactJS [Електронний ресурс] – 2020. Режим доступу: https://www.javatpoint.com/pros-and-cons-of-react – Назва з екрану.

13. Electron | Pros And Cons [Електронний ресурс] – 2018. Режим доступу: https://medium.com/@nalegaveshardul40/electron-pros-and-cons-8f58fd6313d5 – Назва з екрану.

14. Review of Node.JS: Pros and Cons [Електронний ресурс] – 2020. Режим доступу: https://ncube.com/blog/review-of-node-js-pros-and-cons – Назва з екрану.

15. What Is a Database? [Електронний ресурс] – 2018. Режим доступу: https://www.oracle.com/database/what-is-database/ – Назва з екрану.

16. What is SQL? [Електронний ресурс] – 2021. Режим доступу: http://www.sqlcourse.com/intro.html – Назва з екрану.

17. Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch and others [Електронний ресурс] – 2019. Режим доступу: https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/ – Назва з екрану

18. СЕТЕВЫЕ СИМУЛЯТОРЫ И ЭМУЛЯТОРЫ ОБОРУДОВАНИЯ CISCO [Електронний ресурс] – 2021. Режим доступу: https://top-technologies.ru/ru/article/view?id=38134 – Назва з екрану.

19. Cisco [Електронний ресурс] – 2021. Режим доступу: https://habr.com/ru/company/cisco/profile/ – Назва з екрану.

20. What is JavaScript Used For? [Електронний ресурс] – 2021. Режим доступу: https://www.hackreactor.com/blog/what-is-javascript-used-for – Назва з екрану.

ДОДАТОК А. КОД РЕАЛІЗАЦІЇ

App.js

```
import React from 'react'
import AmountPicker from './RouterList'

function App() {
  return (
    <div className="wrapper">
      <img  id = "routerImg" src = "./Images/Router.png" alt =
'Router'></img>
      <h1 id ="heading">ATM Router Configurator</h1>
      <br></br>
      <p>Choose Router's Amount</p>
    <AmountPicker/>
    {/* <IpTextField/> */}
    {/* <Scripts/> */}
    </div>
  );

}



export default App;
```

RouterList.js

```
import React, { useState, useEffect, } from "react";
import Axios from 'axios'



function AmountPicker() {
  const [amountState, setAmountState] = useState('2');
  //const [number, setNumber] = useState('')

  const [r1ip, setr1Ip] = useState('')
  const [r2ip, setr2Ip] = useState('')
  const [r3ip, setr3Ip] = useState('')
  const [r4ip, setr4Ip] = useState('')
  const[script, setScript] = useState([])



  useEffect(() => {

Axios.get('http://localhost:3001/api/get').then((response) => {
      setScript(response.data)
    })
  }, [])

  const submitIp = () => {
    Axios.post("http://localhost:3001/api/insert", {
      r1ip: r1ip,
      r2ip: r2ip,
      r3ip: r3ip,
      r4ip: r4ip,

    })
```

```
  setScript([
    ...script,
    { r1ip: r1ip,
      r2ip: r2ip,
      r3ip: r3ip,
      r4ip: r4ip}
  ])

  }




  const styles = {
    select: {
      display :'block',
      width:'-webkit-fill-available',
      padding:'5',
      height:'50px',
      backgroundColor: '#cae3d1',
      textAlignLast: 'center',
      fontFamily:'"Impact" sans-serif',
      marginTop:'-10px',
      borderRadius:'10px'
    }
}
if(parseInt(amountState) === 1){

  return (
    <div className="container p-5">
      <select style = {styles.select}
        className="custom-select"
```

```
        value={amountState}
        onChange={(e) => {
          const selectedAmount = e.target.value;
          setAmountState(selectedAmount);
        }}
      >

        <option value="2" defaultValue >2</option>
        <option value="3">3</option>
        <option value="4">4</option>
      </select>

      <div className = 'txtArea'>
    <label id = 'scrLabel'>Here is a script</label>
      <textarea id = 'script' rows = {8} cols= {45} readOnly

    ></textarea>

      {/* {script.map((val) => {
        return <h1>Script: {val.newconfig}</h1>
      })} */}


    </div>

    </div>)}


else if(parseInt(amountState) === 2){

 return (
    <div className="container p-5">
      <select style = {styles.select}
        className="custom-select"
```

```
            value={amountState}
            onChange={(e) => {
              const selectedAmount = e.target.value;
              setAmountState(selectedAmount);
            }}
          >

            <option value="2">2</option>
            <option value="3">3</option>
            <option value="4">4</option>
          </select>
          <form className = 'node'>
          <p>Enter the IP address of Nodes in format
255.255.255.255</p>
          <label>R1</label>
          <input type="text" className ="to_validate"
placeholder="IP Address"
        pattern="(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-
5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])"
        onChange = {(e) => {
                    setr1Ip(e.target.value)
                  }}
         title="Wrong IP"  />
     <label>R2</label>
     <input type="text" className ="to_validate"  placeholder="IP
Address"
        pattern="(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-
5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])"
        onChange = {(e) => {
                    setr2Ip(e.target.value)
                  }}
         title="Wrong IP"  />
            <p><button id = 'submitBtn' onClick =
{submitIp}>Submit</button></p>
```

```
            </form>
            <img id = "#networks" src ="./Images/Router2.png" alt =
"Two Routers"></img>

            <div className = 'txtArea'>
          <label id = 'scrLabel'>Here is a script</label>
            <textarea id = 'script' rows = {8} cols= {45} readOnly
            value = {script.map((val) => {
              return [val.r1,val.r2].join('\n\n')
            }

            )}></textarea>


            {/* {script.map((val) => {
              return <h1>Script: {val.newconfig}</h1>
            })} */}


          </div>

          </div>

        )
      } else if(parseInt(amountState) === 3){
        return (<div className="container p-5">

        <select style = {styles.select}
          className="custom-select"
          value={amountState}
          onChange={(e) => {
            const selectedAmount = e.target.value;
            setAmountState(selectedAmount);
          }}
```

```
        >

              <option value="2" defaultValue>2</option>
              <option value="3">3</option>
              <option value="4">4</option>
        </select>

        <form className = 'node'>
        <p>Enter the IP address of Nodes in format
255.255.255.255</p>
        <label>R1</label>
        <input  type="text" className ="to_validate"  placeholder="IP
Address"
          pattern="(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-
5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])"
          onChange = {(e) => {
                      setr1Ip(e.target.value)
                    }}
           title="Wrong IP"  />
     <label>R2</label>
     <input  type="text"  className  ="to_validate"   placeholder="IP
Address"
          pattern="(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-
5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])"
          onChange = {(e) => {
                      setr2Ip(e.target.value)
                    }}
           title="Wrong IP"  />
     <label>R3</label>
     <input  type="text"  className  ="to_validate"   placeholder="IP
Address"
          pattern="(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-
5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])"
          onChange = {(e) => {
```

```
                    setr3Ip(e.target.value)
                  }}
          title="Wrong IP"  />

    <p><button    id    =    'submitBtn'          onClick    =
{submitIp}>Submit</button></p>
      </form>
      <img src ="./Images/Router3.png" alt = "Three Routers"></img>

      <div className = 'txtArea'>
        <label id = 'scrLabel'>Here is a script</label>
        <textarea id = 'script' rows = {8} cols= {45} readOnly
          value = {script.map((val) => {
            return [val.r1, val.r2, val.r3].join('\n\n')
          }

          )}></textarea>
        </div>

    </div>

      )
    } else if(parseInt(amountState) === 4){
      return (<div className="container p-5" >

      <select style = {styles.select}
        className="custom-select"
        value={amountState}
        onChange={(e) => {
          const selectedAmount = e.target.value;
          setAmountState(selectedAmount);
        }}
      >
```

```
            <option value="2" defaultValue>2</option>
            <option value="3">3</option>
            <option value="4">4</option>
        </select>


              <div className = "form" >
        <p>Enter the IP address of Network in format
255.255.255.255</p>

        <label>R1</label>
        <input         type="text"    className    ="to_validate"
placeholder="IP Address"
        pattern="(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-
5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])"
        onChange = {(e) => {
                    setr1Ip(e.target.value)
                  }}
         title="Wrong IP"  />
     <label>R2</label>
     <input type="text" className ="to_validate"  placeholder="IP
Address"
        pattern="(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-
5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])"
        onChange = {(e) => {
                    setr2Ip(e.target.value)
                  }}
         title="Wrong IP"  />
     <label>R3</label>
     <input type="text" className ="to_validate"  placeholder="IP
Address"
        pattern="(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-
5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])"
        onChange = {(e) => {
```

```
                    setr3Ip(e.target.value)
                  }}
          title="Wrong IP"  />
     <br/><label>R4</label>
     <input  type="text"  className ="to_validate"   placeholder="IP
Address"
         pattern="(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-
5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])"
         onChange = {(e) => {
                    setr4Ip(e.target.value)
                  }}
          title="Wrong IP"  />



            <p><button   id   =   'submitBtn'       onClick       =
{submitIp}>Submit</button></p>

         </div>

       <img src ="./Images/Router4.png" alt = "Four Routers"></img>

       <div className = 'txtArea'>
         <label id = 'scrLabel'>Here is a script</label>
         <textarea id = 'script' rows = {8} cols= {45} readOnly
           value = {script.map((val) => {
             return [val.r1, val.r2, val.r3, val.r4].join('\n\n')
           }

           )}></textarea>
         </div>


     </div>
```

```
    )
  }

  }

  export default AmountPicker;
```

index.js

```
  const express = require('express')
  const bodyParser = require('body-parser')
  const cors = require('cors')
  const app = express()
  const mysql = require('mysql')


  const db = mysql.createPool({
    host:"localhost",
    user:"root",
    password:"password",
    database:"configuratordb",
    //insecureAuth : true
  })

  app.use(cors())
  app.use(express.json())
  app.use(bodyParser.urlencoded({ extended: true }));
  app.use(express.urlencoded({ extended: true }))


  app.get("/api/get", (req,res) => {
    const sqlSelect =
    ` select * from (select replace(replace(replace(replace(r1,
'192.168.1.1', (select r1ip from configuratordb.scripts_ip
```

```
        where id = (select max(id) from configuratordb.scripts_ip
))), '192.168.1.2',((select r2ip from configuratordb.scripts_ip
        where id = (select max(id) from configuratordb.scripts_ip
)))),'192.168.1.3', ((select r3ip from configuratordb.scripts_ip
        where id = (select max(id) from configuratordb.scripts_ip
)))),'192.168.1.4', ((select r4ip from configuratordb.scripts_ip
        where id = (select max(id) from configuratordb.scripts_ip
))))
         AS r1 from configuratordb.scripts where id = 1) as r1,

    (select   replace(r2,   '192.168.1.2',   (select   r2ip   from
configuratordb.scripts_ip
        where id = (select max(id) from configuratordb.scripts_ip
)))
         AS r2 from configuratordb.scripts where id = 1) as r2,

    (select   replace(r3,   '192.168.1.3',   (select   r3ip   from
configuratordb.scripts_ip
        where id = (select max(id) from configuratordb.scripts_ip
)))
         AS r3 from configuratordb.scripts where id = 1) as r3,

    (select   replace(r4,   '192.168.1.4',   (select   r4ip   from
configuratordb.scripts_ip
        where id = (select max(id) from configuratordb.scripts_ip
)))
         AS r4 from configuratordb.scripts where id = 1) as r4;`
        db.query(sqlSelect, (err, result) => {
          res.send(result)
           })
    });

     app.post("/api/insert", (req, res) => {
```

```
    const r1ip = req.body.r1ip
    const r2ip = req.body.r2ip
    const r3ip = req.body.r3ip
    const r4ip = req.body.r4ip

    const sqlInsert = `INSERT INTO scripts_ip (r1ip, r2ip, r3ip,
r4ip) VALUES (?,?,?,?)`
    db.query(sqlInsert, [r1ip, r2ip, r3ip, r4ip], (err, result)
=> {
      console.log(result);
     });
  });

  app.listen(3001, () => {
    console.log('running on port 3001');
  })
```

package.json

```
  {
    "name": "atm-app",
    "version": "0.1.0",
    "private": true,
    "dependencies": {
      "@testing-library/jest-dom": "^5.12.0",
      "@testing-library/react": "^11.2.7",
      "@testing-library/user-event": "^12.8.3",
      "axios": "^0.21.1",
      "bootstrap": "^5.0.1",
      "concurrently": "^6.2.0",
      "cross-env": "^7.0.3",
      "electron-builder": "^22.11.7",
      "electron-is-dev": "^2.0.0",
      "electron-squirrel-startup": "^1.0.0",
      "express": "^4.17.1",
```

```
    "mysql": "^2.18.1",
    "mysql2": "^2.2.5",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-scripts": "4.0.3",
    "react-select": "^4.3.1",
    "wait-on": "^5.3.0",
    "web-vitals": "^1.1.2"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "pack": "electron-packager .",
    "test": "react-scripts test",
    "eject": "react-scripts eject",
    "electron-dev": "concurrently \"SET BROWSER=none&&npm run
start\" \"wait-on http://localhost:3000 && electron .\"",
    "package": "electron-forge package",
    "make": "electron-forge make"
  },
  "homepage": "./",
  "main": "public/main.js",
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
```

```
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "description": "This project was bootstrapped with [Create
React App](https://github.com/facebook/create-react-app).",
  "devDependencies": {
    "@electron-forge/cli": "^6.0.0-beta.57",
    "@electron-forge/maker-deb": "^6.0.0-beta.57",
    "@electron-forge/maker-rpm": "^6.0.0-beta.57",
    "@electron-forge/maker-squirrel": "^6.0.0-beta.57",
    "@electron-forge/maker-zip": "^6.0.0-beta.57",
    "electron": "^13.1.2"
  },
  "proxy": "http://localhost:3001",
  "author": "Serhii Lohutov",
  "license": "ISC",
  "config": {
    "forge": {
      "packagerConfig": {},
      "makers": [
        {
          "name": "@electron-forge/maker-squirrel",
          "config": {
            "name": "atm_app"
          }
        },
        {
          "name": "@electron-forge/maker-zip",
          "platforms": [
            "darwin"
          ]
```

```
          },
          {
            "name": "@electron-forge/maker-deb",
            "config": {}
          },
          {
            "name": "@electron-forge/maker-rpm",
            "config": {}
          }
        ]
      }
    }
  }
```

index.css

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI',
'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica
Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas,
'Courier New',
    monospace;
}

.wrapper{
```

```
  padding-top: 1rem;
  margin: 0 auto;
  width: 600px;
}

#heading{
  text-align: center;
   font-family: Impact, Haettenschweiler, 'Arial Narrow Bold',
sans-serif;
  padding-left: 20px;
  float: auto;
        background-size: 20px 20px;
        background-position: 0px 11px;


  top: 18px;
  left: 10px;
}

p{
  font-size: 25px;
}

#routerImg{
  /* display:inline-block;
  width: 50px;
  height: 50px;
  padding-right: 50px;
  padding-bottom: 20px;
   */
   float: left;
  width: 80px;
  height: 80px;
  background: #555;
```

```
}

#selName{
  padding-top: 50px;
}

#amount{
  padding-top: 0px;
}

/* .addRouterBtn{
  background-color: #4CAF50;
    border: none;
    color: white;
    padding: 15px 32px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
}
 */

 #networks{
   padding-top: 200px;
 }

 #submitBtn{
  background-color: #4CAF50;
  border: none;
  color: white;
  padding: 16px 32px;
  text-decoration: none;
  margin: 4px 2px;
  cursor: pointer;
```

```
 }

#scrLabel {
  padding-right: 100px;
  padding-bottom: 100px;
  font-size: xx-large;
}

 .txtArea{
   outline: none;
   resize: auto ;
 }

 .to_validate:valid {
  color: black;
  border: 5px solid #dadadada;
  border-radius: 7px;

}
.to_validate:invalid {
  color: navy;
  outline: none;
  border-color: #ff1050;
  box-shadow: 0 0 10px #ff0000;
}
```

main.js

```
  const electron = require('electron');
  const app = electron.app;
  const BrowserWindow = electron.BrowserWindow;

  const path = require('path');
  const url = require('url');
  const isDev = require('electron-is-dev');
```

```
let mainWindow;

function createWindow() {
  mainWindow = new BrowserWindow({width: 900, minWidth: 900,
                                  height: 680, minHeight: 680,
                                 icon:"atm-app\
public\favicon.ico",
                                 webPreferences: {
                                  webSecurity: false}});
  mainWindow.loadURL(isDev    ?    'http://localhost:3000'    :
`file://${path.join(__dirname, '../build/index.html')}`);
  mainWindow.on('closed', () => mainWindow = null);
}

app.on('ready', createWindow);

app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') {
    app.quit();
  }
});

app.on('activate', () => {
  if (mainWindow === null) {
    createWindow();
  }
});
```