Міністерство освіти і науки України

Національний технічний університет

«Дніпровська політехніка»

Інститут електроенергетики

(інститут)

Факультет інформаційних технологій
(факультет)
Кафедраінформаційних технологійта комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеня бакалавра
(бакалавра, спеціаліста, магістра)

студента Конопкін Сергій Ігорович
(ПІБ)

академічної групи 123-20ск-1
(шифр)

спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)

за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)

на тему «Комп'ютерна система АТ «ДТЕК Дніпровські електромережі» з опрацюванням побудови та налаштування корпоративної мережі та з підтримкою мобільного застосунку перевірки поточних відключень»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	Цвіркун Л.І.			
розділів:				
розробка апаратної частини	доц. Ткаченко С.М.			
розробка корпоративної мережі	ас. Бешта Л.В.			
Рецензент				
Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро

2023

ЗАТВЕРДЖЕНО:

завідувач кафедри
інформаційних технологій
та комп'ютерних інженерії
(повна назва)
_____ Гнатушенко В.В.
(підпис) (прізвище, ініціали)
« » 2023 року

ЗАВДАННЯ

на кваліфікаційну роботу ступеня бакалавр

студента Конопкін С.І. **академічної групи** 123-20ск-1
(прізвище та ініціали) (шифр)

спеціальності 123 Комп'ютерна інженерія

за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)

на тему «Комп'ютерна система АТ «ДТЕК Дніпровські електромережі» з опрацюванням побудови та налаштування корпоративної мережі та з підтримкою мобільного застосунку перевірки поточних відключень» затверджену наказом ректора НТУ «Дніпровська політехніка» від 16.05.2023 № 350-с

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел конкретизується предмет та мету роботи та виконується постанова завдання	10.05.2023
Розробка апаратної частини	На основі аналізу підприємства формулюються технічні вимоги до комп'ютерної системи та розробляється апаратна частина системи	17.05.2023
Розробка корпоративної мережі	Виконується розрахунок налаштувань корпоративної мережі та перевірка роботи системи, розробляються методи та налаштування обладнання для захисту інформації в системі	24.05.2023
Розробка компонента системи	Виконується детальна розробка компонента системи	30.06.2023

Завдання видано _____ проф. Цвіркун Л.І.
(підпис керівника) (прізвище, ініціали)

Дата видачі 19.04.2023

Дата подання до екзаменаційної комісії 07.07.2023

Прийнято до виконання _____ Конопкін С.І.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 74 с., 17 рис., 8 табл., 6 джерел.

Об’єкт: комп’ютерна система: ДТЕК з детальним опрацюванням підприємства, а також з опрацюванням комп’ютерної мережи підприємства, та створення мобільного додатку по відключенню світла.

Мета роботи – створення комп’ютерної мережі для компанії ДТЕК та розробка мобільного додатку.

Спрямованого на передачу і обмін інформацією між різними відділеннями компанії та збір інформації від клієнтів. Використання такої системи має на меті поліпшення ефективності роботи персоналу шляхом підвищення швидкості та надійності обміну даними. Крім того, система розроблена з використанням відкритих стандартів, що дозволяє використовувати її з іншими системами та послугами забезпечення безпеки та аналітики.

Такий підхід сприяє покращенню комунікації між різними відділами компанії, спрощує збір і обробку інформації, а також сприяє збільшенню продуктивності та ефективності роботи персоналу.

Робота системи перевірена за допомогою моделі схеми корпоративної мережі із застосуванням програми Cisco Packet Tracer.

Результати перевірки у вигляді таблиць та графіків описані і наводяться у пояснювальній записці та додатках.

Локальна обчислювальна мережа відповідає всім заявленим вимогам. Вона легко адмініструється, а також при необхідності може бути розширена.

Зміст

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ І ТЕРМІНІВ

ПО – програмне забезпечення

ТОВ – товариство з обмеженою відповідальністю

СЕО – головний виконавчий директор

КС – комп’ютерна система

КМ – корпоративна мережа

ПК – персональний комп’ютер

СВУ – системи виробничого управління

СУП – системи управління персоналом

СФУ – системи фінансового управління

СЗБП – системи забезпечення безпеки працівників

СЕБ – системи енергозабезпечення

СІЗ – системи інформаційного забезпечення

МА – мережева архітектура

HTTP − (Hypertext Transfer Protocol). Це протокол передачі даних, який використовується для обміну інформацією в Інтернеті.

ВСТУП

В сучасному бізнес-середовищі належна організація внутрішньої комунікації є невід'ємною складовою успішної діяльності будь-якої організації. Особливо для великих підприємств з численним персоналом, налагодження та проектування локальних мереж стають важливим завданням. Цей процес включає в себе прокладання телефонних та комп'ютерних мереж, встановлення систем відеоспостереження та забезпечення підключення до глобальної мережі. Крім того, для запобігання відключенню систем у разі стрибків напруги використовуються блоки безперебійного живлення.

В результаті, правильно налагоджена внутрішня комунікація підприємства дозволяє забезпечити надійне зв'язок між співробітниками, ефективну обмін інформацією та підвищити продуктивність роботи. Крім того, вона гарантує постійний доступ до необхідних ресурсів та даних, а також забезпечує безперебійну роботу системи навіть у разі виникнення збоїв. Такі заходи впливають на загальну стійкість і надійність підприємства, зменшуючи можливі ризики втрати даних та перебоїв у роботі.

Мета цієї роботи є розробка проекту комп'ютерної мережі для ДТЕК « Донбаська паливно-енергетична компанія», а також розробка мобільного додатку по відключенню світла спрямованого на аварійне відключення світла. Цей додаток розроблений з метою забезпечення безпеки та ефективного керування освітленням в ситуаціях надзвичайних ситуацій.

1 Стан питання і постановка завдання

1.1 Загальна структура бази практики «ДТЭК»

ДТЕК - це одна з найбільших енергетичних компаній в Україні. Вона є частиною групи компаній SCM (System Capital Management) і спеціалізується на виробництві, передачі та постачанні електроенергії.

Основні напрямки діяльності ДТЕК включають:
Виробництво електроенергії: ДТЕК володіє та експлуатує різноманітні енергетичні активи, включаючи теплові та гідроелектростанції, вітрові ферми та сонячні електростанції. Вони виробляють електроенергію з використанням різних джерел, забезпечуючи стабільне енергопостачання в регіоні.

Передача та розподіл електроенергії: ДТЕК також володіє та управляє енергетичними мережами, які передають та розподіляють електроенергію. Це включає високовольтні та низьковольтні мережі, підстанції та лінії електропередачі.

Торгівля електроенергією: ДТЕК є одним з найбільших постачальників електроенергії на українському ринку. Вони забезпечують енергією як промислові, так і резиденціальні клієнти, та здійснюють торгівлю електроенергією на оптовому ринку.

Інновації та енергоефективність: ДТЕК активно впроваджує інноваційні технології та розвиває проекти з енергоефективності. Вони зосереджуються на зменшенні викидів шкідливих речовин, оптимізації енергоспоживання та використанні відновлюваних джерел енергії.

Соціальна відповідальність: ДТЕК віддає перевагу принципам сталого розвитку та соціальної відповідальності. Вони реалізують проекти зі спільнотами, спрямовані на покращення життя місцевих жителів, підтримку освіти та охорону навколишнього середовища. ДТЕК є одним з ключових

гравців на ринку енергетики в Україні та виконує важливу роль у забезпеченні енергетичної безпеки країни.

MODUS - це довгострокова DX програма компанії ДТЕК, що першою в українській енергетиці розпочала комплексну цифрову трансформацію. Вона охоплює всі основні виробничі та адміністративні процеси бізнесу. Команда MODUS - це менеджери, інженери, програмісти та дизайнери, що вирішують бізнес-завдання за допомогою інноваційних технологічних рішень.

Організаційно-управлінська структура ДТЕК MODUS, як лінійна система, включає багаторівневу ієрархію керівництва рисунок 1.1. У цій структурі, керівник виконує функцію одноосібного керівництва підлеглими керівниками, а нижче розташовані керівники підпорядковуються лише своєму безпосередньому керівникові.

Таким чином, інформація та рішення передаються від верхнього рівня до нижчих рівнів через ланцюг командування. Кожен керівник має свою підпорядковану команду та звітує перед безпосереднім керівником.

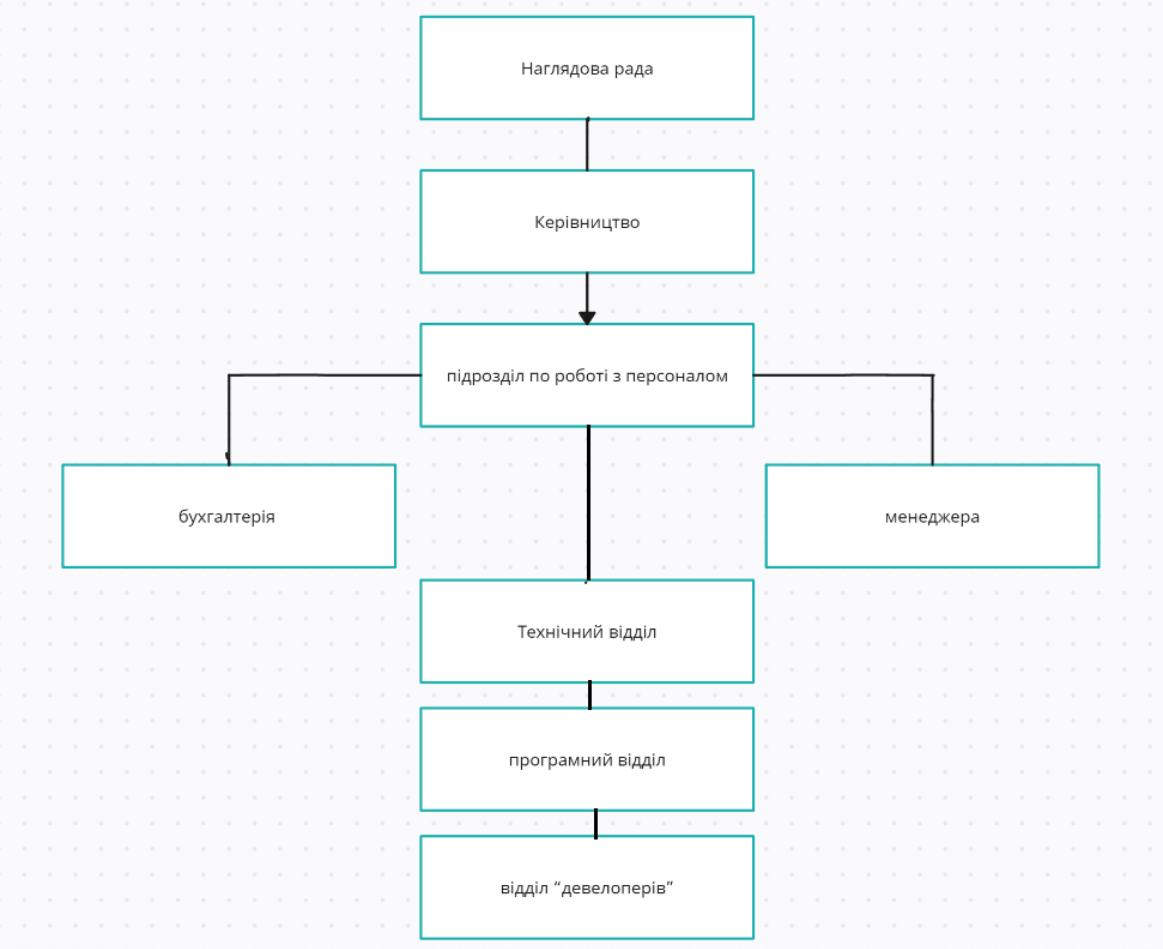

Рисунок 1.1 − Схема Організаційно-управлінська структура

ДТЭК «Modus» складається з таких підрозділів: наглядова рада, керівництво, бухгалтерія, менеджера, підрозділ по роботі з персоналом, персонал по роботі з клієнтами, та програмне відділення та відділення «девелоперів».

Керівництво повністю керує процесом роботи компанії, основними переговорами з клієнтами. Менеджер більш детально слідкує за робочим процесом та працює з клієнтами на початковому етапі.

Бухгалтерія веде аналіз та статистику доходів на витрат компанії, відповідальний за заробітну платню.

Технічний відділ слідкує за працездатністю мережі та вирішенням проблем з мережею.

Персонал по роботі з клієнтами займається пошуком нового персоналу, бо організація швидко розвивається.

Програмне відділення та відділення «девелоперів», які займаються програмним забезпеченням, а також слідкують за стабільністю роботи програм та серверів. Як раз все відділення IT знаходиться на базі «MODUS».

1.2 Системи управління персоналом

В ДТЕК, як і в багатьох великих компаніях, використовуються системи управління персоналом для ефективного керування робочою силою та забезпечення оптимального функціонування організації. Ці системи допомагають автоматизувати та оптимізувати різні аспекти управління персоналом, такі як найм, підбір, розвиток, заробітна плата, оцінка праці, управління компетенціями та багато інших.

Найм та підбір персоналу: системи управління персоналом допомагають управляти процесом найму та підбору кандидатів на вакансії. Вони можуть включати модулі для розміщення вакансій, збору резюме, оцінки кандидатів та проведення співбесід.

Розвиток та навчання: системи управління персоналом надають можливості для планування, відстеження та оцінки процесу навчання та розвитку працівників. Вони можуть включати модулі для проведення тренінгів, оцінки компетенцій, розробки індивідуальних планів розвитку тощо.

Управління компетенціями: системи управління персоналом дозволяють визначати, відстежувати та оцінювати компетенції працівників. Вони допомагають ідентифікувати прогалини в компетенціях, планувати розвиток та забезпечувати належний розподіл знань та навичок усередині організації.

` Управління заробітною платою та винагородами: системи управління персоналом допомагають встановлювати, керувати та відстежувати заробітну плату та винагороди працівників. Вони можуть включати модулі для обрахунку заробітної плати, адміністрування бонусів, контролю витрат на персонал тощо.

Управління продуктивністю та оцінка праці: системи управління персоналом надають інструменти для оцінки та управління продуктивністю працівників. Вони можуть включати модулі для встановлення цілей, проведення оцінок, надання зворотного зв'язку та розробки планів для поліпшення продуктивності.

1.3 Загальна структура «MODUS»

В основному робочі місця оснащені моноблоками, для легкості переміщення в ситуації переїзду. Моноблоки використовуються на перших поверхах банку де працюють з клієнтами, на всіх інших поверхах використовують ПК, які оснащені сучасними технологіями, для швидкої і якісної роботи.

Відділення «девелоперів» працює з серверами офісу. Це відділення займається не тільки розробкою програмного забезпечення для ДТЕК, а також слідкує за їх роботою і за роботою сервера також.

Сервер - виділений або спеціалізований комп'ютер для виконання сервісного програмного забезпечення. Він оптимізований для роботи з іншими комп'ютерами. Клієнтами сервера можуть бути комп'ютери, факси, принтери. Для взаємодії з клієнтом сервер виділяє необхідні ресурси між процесами взаємодії і очікує запити на відкриття з'єднання. Залежно від типу такого ресурсу, сервер може обслуговувати процеси в межах однієї комп'ютерної системи або процеси на інших машинах через канали передачі даних або мережеві з'єднання.

Компанії використовують сервера для загального доступу всіх співробітників до певної інформації і для загального користування доступними ресурсами.

Весь обмін інформацією йде виключно через центральний комп'ютер або агрегат, на який таким способом покладається дуже велике навантаження, тому нічим іншим, крім мережі, він займатися не може. Як правило, саме

центральний комп'ютер або агрегат є найпотужнішим в мережевому відношенні, і саме на нього покладаються всі функції по управлінню мережею і передачі даних. Весь обмін інформацією йде виключно через центральний комп'ютер або агрегат, на який таким способом покладається дуже велике навантаження, тому нічим іншим, крім мережі, він займатися не може. Як правило, саме центральний комп'ютер або агрегат є найпотужнішим в мережевому відношенні, і саме на нього покладаються всі функції по управлінню мережею і передачі даних.

На апаратному рівні локальна обчислювальна мережа представляє сукупність комп'ютерів та інших засобів обчислювальної техніки, об'єднаних за допомогою кабелів та мережевих адаптерів Комп'ютер, підключений до обчислювальної мережі, називається робочою станцією чи сервером, залежно від виконуваних ним функцій

На підприємстві використовуються мережева архітектура. Мережева архітектура – це комбінація стандартів, топології і протоколів, які утворюють працездатну мережу. МА характеризує загальну структуру мережі, тобто всі компоненти завдяки яким мережа функціонує як апаратні засоби так і системи ПЗ. Кожна МА має свої характеристики, параметри продуктивності, апаратні та програмні засоби. Це дає проектувальнику полегшенні можливості створення мережі, яка має задовольняти вимогам функціонування. Тобто проектувальник обґрунтовує вибір МА і не проводить внутрішнього аналізу та розрахунку мережі.

До основних переваг віодносяться всі обчислення які виконуються на сервері, вимоги до комп'ютерів, на яких встановлено клієнт, знижуються
Всі дані зберігаються на сервері, який, як правило, захищений набагато краще за більшість клієнтів. На сервері простіше забезпечити контроль повноважень, щоб дозволяти доступ до даних лише клієнтам із відповідними правами доступу, дозволяє розвантажити мережі через те, що між сервером і клієнтом передаються невеликі порції даних.

До основних недоліків відноситься підтримка роботи даної системи яка потребує окремого спеціаліста – системного адміністратора. Також Непрацездатність сервера може зробити непрацездатною всю обчислювальну мережу.

В компанії «ДТЕК» існують шість основних відділів рисунок 1.1 – 1.2:

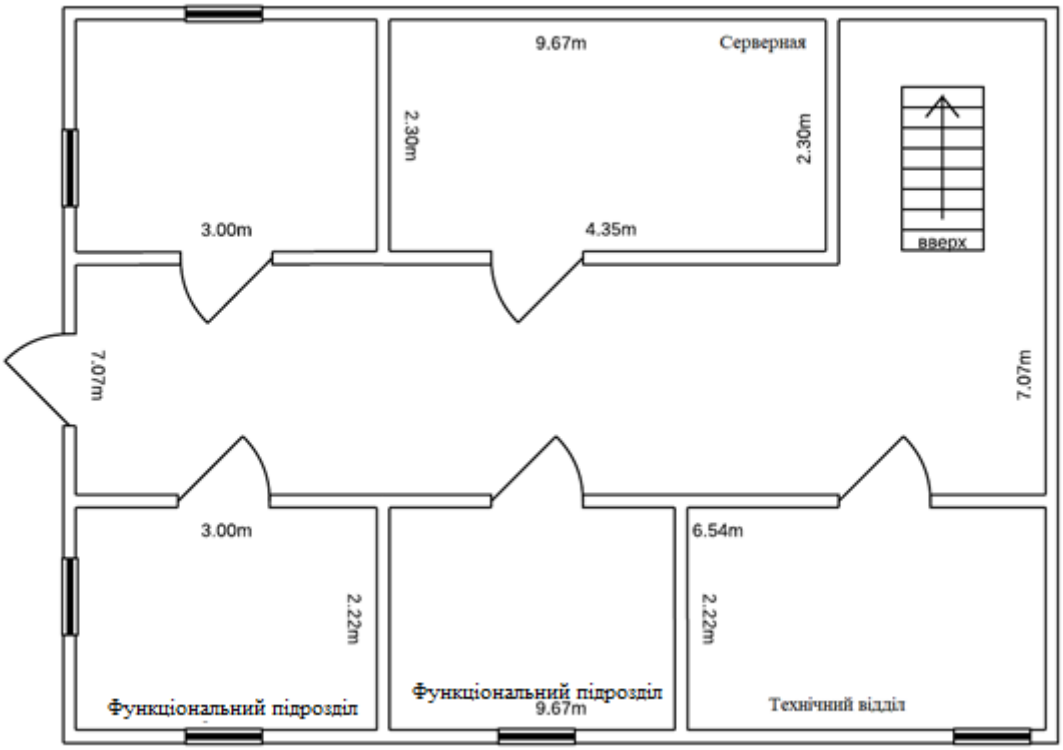

Рисунок 1.1 − План першого поверху приміщення

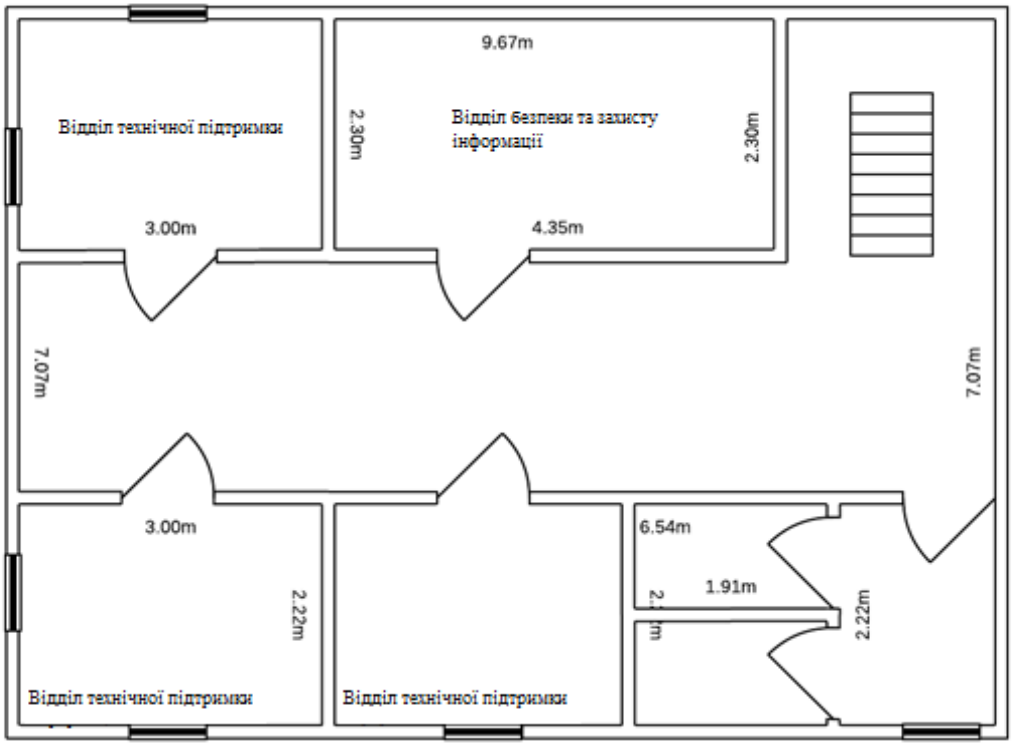

Рисунок 1.2 − План другого поверху приміщення

Відділ інформаційних технологій – відділ IT забезпечує технічну підтримку, розробляє інформаційні системи, забезпечує безпеку та захист інформації, а також здійснює керування мережами і інфраструктурою організації

Відділ безпеки та захисту інформації: цей відділ забезпечує безпеку мережі та захист від зовнішніх загроз. В звіті можна згадати про використані механізми безпеки, такі як файрволи, антивірусне програмне забезпечення, контроль доступу та моніторинг системи.

Відділ технічної підтримки: цей відділ надає технічну підтримку користувачам мережі, включаючи вирішення проблем, підключення нових користувачів, налаштування облікових записів тощо. У звіті можна згадати про співпрацю з цим відділом під час управління користувачами та розв'язання проблем користувачів.

Три відділа: фінансовий, бухгалтерія та маркетинговий. Фінансовий відділ займається управлінням фінансовими ресурсами компанії. Вони відповідають за складання та аналіз фінансової звітності, контроль за бюджетами, фінансовий планування, управління операційними витратами, планування інвестицій та прибутковості проектів. Також фінансовий відділ забезпечує контроль за фінансовими ризиками і виконанням фінансових регуляції.

Маркетинговий відділ в Айті компанії відповідає за розробку та впровадження маркетингових стратегій. Вони проводять дослідження ринку, аналізують конкурентну ситуацію, визначають цільову аудиторію та розробляють маркетингові плани. Також маркетинговий відділ відповідає за рекламу, просування продуктів або послуг компанії, взаємодію з клієнтами та збут продукції.

Бухгалтерський відділ в Айті компанії відповідає за облік фінансової діяльності компанії. Вони здійснюють бухгалтерський облік, ведуть книги доходів та витрат, опрацьовують податкову звітність, забезпечують виконання бухгалтерських стандартів та правил.

Бухгалтерський відділ також контролюєплатежі, веде облік заборгованостей та кредиторської заборгованості, аналізує фінансові показники компанії

1.4 Стислі відомості про програмні засоби

В ДТЕК для комп'ютерної мережи використовуються різноманітні програми та програмні рішення для забезпечення ефективності, безпеки та керування мережею. Ось кілька прикладів таких програм:

Системи моніторингу мережі: Ці програми дозволяють відстежувати стан мережевих пристроїв, моніторити пропускну здатність, виявляти проблеми та забезпечувати превентивне управління мережею Nagios, Zabbix

Системи управління конфігураціями: Ці програми дозволяють централізовано керувати конфігураціями мережевих пристроїв, забезпечуючи їхню однорідність та безпеку. Приклади таких програм включають Ansible

Системи моніторингу безпеки: Ці програми слідкують за безпекою мережі та виявляють потенційні загрози, атаки або несправності. Вони можуть включати інтрудер-детекційні системи (IDS), системи управління журналами (SIEM), програми сканування вразливостей тощо. Приклади таких програм включають Splunk, Nessus

Системи управління доступом: Ці програми дозволяють контролювати доступ до мережевих ресурсів і забезпечують автентифікацію та авторизацію користувачів. Вони можуть включати системи одноразових паролів (OTP), системи одного входу (SSO), системи керування ідентифікацією та доступом (IAM) тощо. Приклади таких програм включають Okta, Microsoft Active Directory

Системи резервного копіювання та відновлення: Ці програми забезпечують регулярне резервне копіювання даних та можливість відновлення в разі виникнення неполадок або втрати даних. Вони можуть включати програми для резервного копіювання на серверах, хмарних системах збереження, програми для резервного копіювання баз даних тощо.

Приклади таких програм включають Acronis Backup, Commvault Simpana.ОС використовують Linux Ubuntu, тому що безпека даних для компанії це дуже важливо.

У компанії використовуються таки системи зв'язку – Skype, та Telegram(це на даний момент використовують месенджер, бо тільки через телеграм чати технічна підтримка може контактувати з клієнтами(мерчентами), до тих пір поки «Програмне відділення» не розробить чат на головній сторінці сайту).

Linux, Unix – встановлення, настроювання, адміністрування основних мережевих сервісів.

Microsoft Excel використовують для роботи з таблицями та розробок різних програм, які будуть вирішувати задачі зв'язані з сортуванням та підрахунками.

Для розробки програмного забезпечення використовують Visual Code, тому що це дуже потужний редактор для розробки програмного забезпечення.

1.5 Завдання і мета роботи

Одним із завдань кваліфікаційної роботи є розробка комп'ютерної системи АТ «ДТЕК Дніпровські електромережі» з опрацюванням побудови та налаштування корпоративної мережі.

Створення корпоротивної мережі включає в себе такі вимоги:

побудова топології мережі;

виконання розрахунків IP-адрес, з використанням VLSM;

базове налаштування обладнення;

налаштування VLAN;

налаштування виходу у інтернет мережу;

налаштування списків доступу;

налаштування сервісу AAA на сервері;

Побудова корпоративної мережі повинна бути розроблена в CISCO packet tracer.

1.6 Визначення можливих напрямків рішення поставлених завдань

Для вирішення поставлених завдань було запропоновано наступні заходи: налаштування списків доступу на маршрутизаторах з метою контролю трафіку; впровадження протоколів IPsec та ISAKMP для забезпечення безпеки комунікації між головною мережею та віддаленими пунктами; використання протоколу SSH для безпечного віддаленого доступу та виконання команд на віддалених комп'ютерах.

Корпоративна мережа підприємства «ДТЭК» має наступну архітектуру :

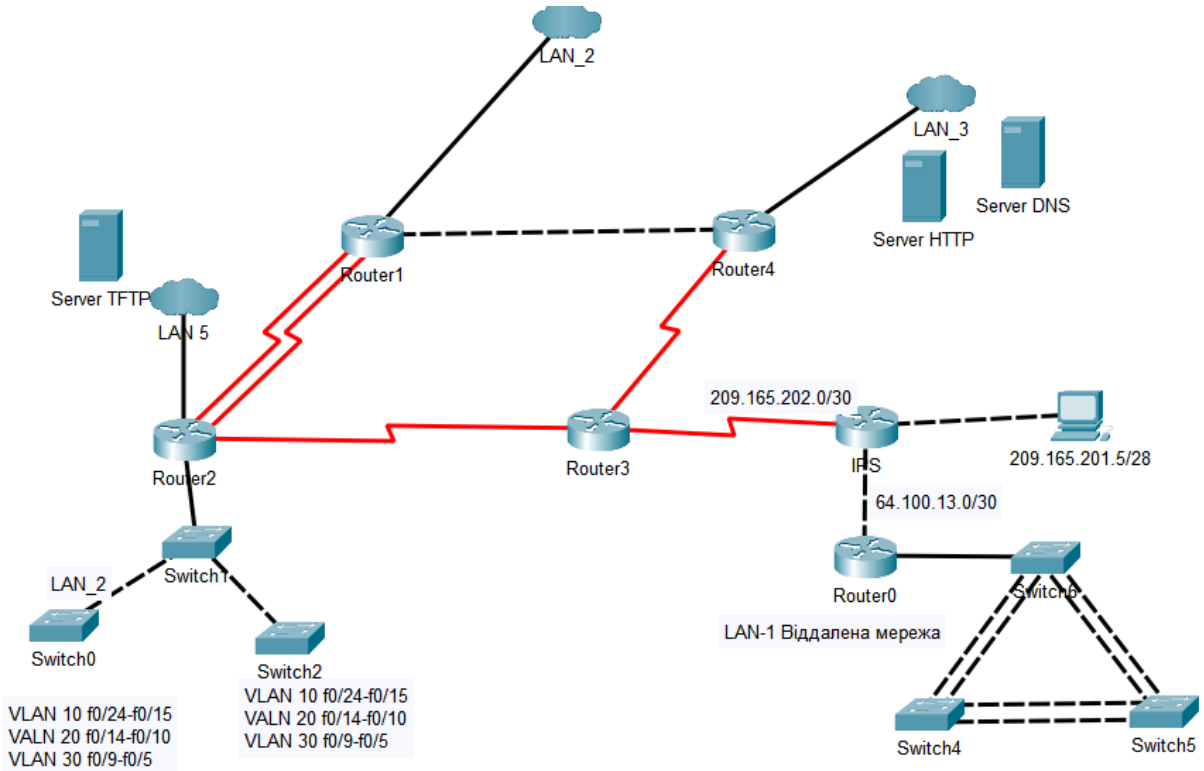

Рисунок 1.3 – Логічна схема ДТЕК

Згідно з цієї схеми, можна побачити те що, для ДТЄК відводяться сім підмережі, а для відділу “ Modus “ одна віддалена підмережа

Мета програми «MODUS»– поліпшити операційну ефективність, гнучкість, автоматизацію та цифралізацію. Відповідати викликам часу, в енергетики України. Почати автоматично розпізнавати дефекти за допомогою комп'ютерного зору та лідарного сканування мереж, що дозволило підвищити безпеку виробничого процесу.

.

2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ КОПМ'ЮТЕРНОЇ МЕРЕЖІ

2.1 Вимоги до структури і функціонування системи.

Комп'ютерна система розроблена та призначена для забезпечення обміну інформацією з управлінням та для роботи з клієнтами компанії ДТЕК. Система повинна бути доступною для користувачів протягом більшості часу. Для цього проекту потрібно створити п'ять локальних мереж.

Для відповідності вимогам, рекомендується використовувати Ip-блок-адресу для призначення підмережі,

Корпоративна мережа складається з 5 підмереж:

- відділ технічної підтримки
- відділ безпеки та захисту інформації
- функціональні підрозділи
- відділ інформаційних технологій
- віддаленна мережа

Потрібно розбити IP-адресу 172.23.104.0/21 на 5 підмереж враховуючи кількість вузлів для LAN:

LAN1 – 15 вузлів; LAN2 – 75 вузлів; LAN3 – 75 вузлів; LAN4 – 90 вузлів; LAN5 – 21 вузлів.

2.2 Вимоги до надійності системи

Вимоги до надійності системи комп'ютерної мережі включає наступні аспекти:

Запобігання втраті даних: У системі повинен використовуватися RAID масив другого рівня. Це забезпечує відновлення даних у випадку відмови одного або декількох дисків. Якщо один диск виходить з ладу, інші диски містять достатню інформацію для відновлення даних, що дозволяє уникнути втрати даних та забезпечити безперебійну роботу системи.

Крім того, у системі має використовується резервне копіювання даних, яке забезпечує створення резервних копій важливих даних на зовнішніх носіях, таких як дисків або хмарних сховищ. Це дозволяє відновлювати дані з резервних копій в разі втрати чи пошкодження основних даних.

Тестування та перевірка комп’ютерна мережа повинна проходити ретельне тестування та перевірку, щоб впевнитися у її надійності. Це включає в себе проведення тестів на стійкість до великих навантажень, відновлення після збоїв та інші тестові сценарії. Для цього використовують «Wireshark ». Воно дозволяє перехоплювати, аналізувати та моніторити пакети даних, що проходять через мережу також «Wireshark » допомагає виявляти проблеми з мережевим з'єднанням, аналізувати протоколи та перевіряти безпеку мережі.

Захист від несанкціонованого доступу: Система повинна мати механізми захисту від несанкціонованого доступу до мережевих ресурсів. Фірмове програмне забезпечення для безпеки (Security Suite Software): Такі програми, як «Symantec Endpoint Protection» або «McAfee» забезпечують захист від шкідливих програм, фішингу та інших загроз.

2.3 Вимоги до безпеки мережі

Система повинна мати механізми аутентифікації, щоб перевіряти ідентичність користувачів, перш ніж надавати доступ до мережевих ресурсів. Це може включати використання паролів, біометричних даних таких як «fingerprint», токенів: Після аутентифікації система повинна контролювати права доступу користувачів до різних ресурсів в мережі.

Система мережі повинна мати захист конфіденційності даних під час їх передачі по мережі. Шифрування забезпечує захист від несанкціонованого доступу та перехоплення даних шляхом застосування криптографічних алгоритмів.

Захист від шкідливого програмного забезпечення: Система повинна мати механізми для виявлення та запобігання шкідливому програмному забезпеченню, такому як віруси, троянські програми, шпигунське програмне забезпечення та інші загрози. Це може включати використання антивірусного програмного забезпечення, фаєрволів та систем виявлення вторгнень (IDS/IPS).

Компоненти мережі, такі як сервери, комутатори, маршрутизатори та інше обладнання, повинні бути захищені від несанкціонованого доступу. Це може включати фізичний доступ до приміщень, резервне копіювання даних та застосування контролю доступу.

2.4 Розробка специфікації апаратних засобів КС

Вибір маршрутизатора для забезпечення надійної та продуктивної мережі компанії, було обрано маршрутизатор Cisco ISR 4331. Ця модель має високу продуктивність та розширені можливості для обробки даних в мережі.

Завдяки своїм технічним характеристикам, які включають 3 х інтерфейси Ethernet 10Base-T/100Base-TX/1000Base-T та швидкість передачі 1 Гбіт/с, маршрутизатор Cisco ISR 4331 відповідає потребам компанії ДТЕК для забезпечення стабільного зв'язку між мережевими пристроями.

Для оптимального функціонування мережі компанії ДТЕК та задоволення вимог щодо друку документів, були обрані дві моделі принтерів. Першою моделлю є лазерний принтер HP LaserJet Enterprise M608dn, який відповідає потребам компанії завдяки своїй високій швидкості друку, великому лотку для паперу та вбудованій мережевій підтримці. Другою моделлю є кольоровий багатофункціональний принтер Canon imageRUNNER ADVANCE C5540i, який підтримує друк, копіювання та сканування в мережі зі зручним кольоровим сенсорним екраном. Обидва принтери відповідають вимогам компанії ДТЕК щодо якості друку та забезпечують зручну роботу з документами.

Для побудови ефективної локальної підмережі та забезпечення зв'язку між мережевими пристроями, було обрано одну модель комутатора.

Cisco Catalyst 2960X-24TS-L: Цей комутатор має 24 порти Fast Ethernet (10/100 Mbps) і 2 порти гігабітного Ethernet (10/100/1000 Mbps). Управління мережею надає можливості для розширеного управління мережею. Він підтримує такі функції, як VLAN, Quality of Service (QoS), Spanning Tree Protocol (STP), Access Control Lists (ACLs), що дозволяють налаштовувати та керувати роботою комутатора, також він забезпечує швидку передачу даних, надає широкі можливості управління, безпеки та масштабованості. Він чудово підходить та відповідає потребам компанії ДТЕК для забезпечення зручного з'єднання робочих станцій, серверів та інших мережевих пристроїв.

Для мережі буде випростовуватись роутер Asus ROG Rapture GT-AX6000

Це технологічно новий роутер на ринку спеціально розробленим для вимогливих корпоративних мереж. Основні характеристики цього роутера включають: Швидкість: Роутер підтримує стандарт Wi-Fi 6 (802.11ax) і забезпечує потужну швидкість до 6000 Мбіт/с, що дозволяє швидке і безперебійне підключення до мережі для багатьох користувачів та робітників одночасно.

Роутер підтримує технології Quality of Service (QoS) для пріоритизації трафіку, дозволяє налаштовувати віртуальні приватні мережі (VPN) та підтримує можливість розширення мережі шляхом підключення додаткових мережевих пристроїв.

Для забезпечення безперебійної роботи серверів у разі вимкнення світла будуть використовуватись ДБЖ від компанії Powercom, а саме MRT-3000 IEC. Він має потужність 3000 Вт і за рахунок шести вбудованих батареї може забезпечити хорошу автономність. Ще одним із його плюсів є те шо, его можна встановити горизонтально та вертикально, шо у разі потреби може забезпечити економію місця.

Таблиця 2.1 – Специфікація обладнання

Позиція	Найменування і тех. характеристика	Тип, марка, позначення документа	Одиниці виміру	Кількість
1	Cisco Catalyst 2960X-24TS-L	Konopkin_Switch_1 Konopkin_Switch_2 Konopkin_Switch_3 Konopkin_Switch_4 Konopkin_Switch_5 Konopkin_Switch_6 Konopkin_Switch_7 Konopkin_Switch_8 Konopkin_Switch_9 Konopkin_Switch_10	Шт.	10
2	HP LaserJet Enterprise M608dn	PrinterHp_Konopkin_1 PrinterHp_Konopkin_2 PrinterHp_Konopkin_3	Шт.	3
3	Canon imageRUNNER ADVANCE C5540i	Printer_C5540i_1 Printer_C5540i_2 Printer_C5540i_3	Шт.	3
4	Asus ROG Rapture GT-AX6000	Konopkin_Router_1 Konopkin_Router_2 Konopkin_Router_3 Konopkin_Router_4	Шт.	4
5	POWERCOM MRT-3000 IEC	UPS	Шт	6

2.5 Вибір і обґрунтування структурної схеми комплексу технічних засобів комп’ютерної системи

Структурна схема технічних засобів комп'ютерної системи обласного центру зайнятості включає мережеве обладнання та компоненти комп'ютерної системи. Ця схема відображає конфігурацію мережі, яка розділяє обласний центр зайнятості на рівень підприємства та рівень моделі ієрархії підмережі.

На рисунку 2.1 представлена структурна схема комплексу технічних засобів комп'ютерної системи центру зайнятості. Ця мережа була спроектована відповідно до технічних вимог центру зайнятості.

Апаратне забезпечення включає робочі станції (хости), корпоративні сервери, спільні ресурси, маршрутизатори, комутатори та мережевий зв'язок у вигляді бездротових кабелів та адаптерів. Ядро мережі складається з чотирьох маршрутизаторів (R1 до R4), які забезпечують з'єднання мереж. Для забезпечення доступу до ядра використовується технологія Gigabit Ethernet

.

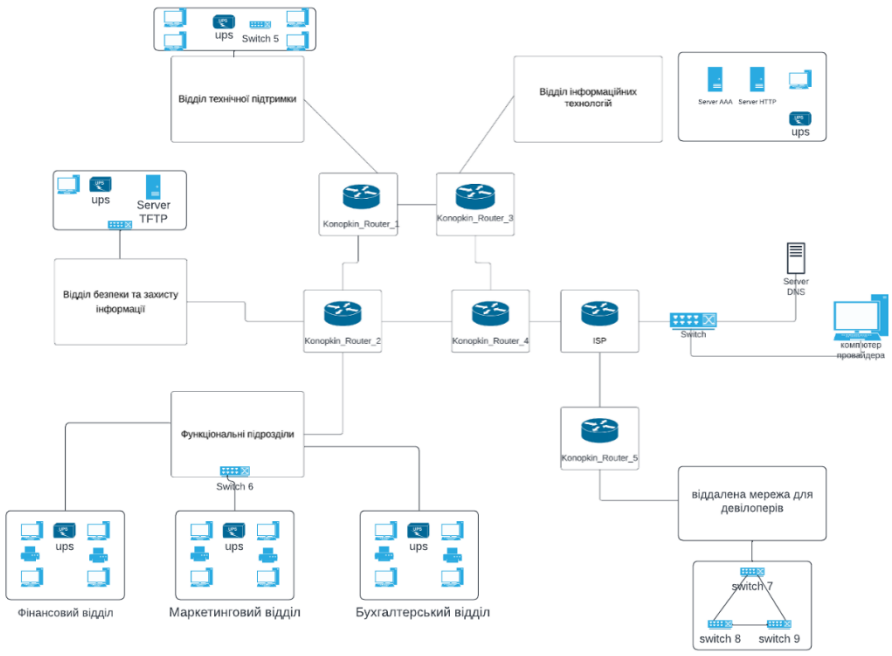

Рисунок 2.1 − Структурна схема комплексу технічних засобів комп’ютерної системи

3 ПРОЕКТУВАННЯ КОРПОРАТИВНОЇ МЕРЕЖІ ТАПЕРЕВІРКА РОБОТИ КОМП'ЮТЕРНОЇ СИСТЕМИ ПІДПРИЄМСТВА

3.1 Розрахунок схеми адресації корпоративної мережі

Адресний простір 172.23. 104 .0/21 було взято за основу дотримуючись технічних вимог для побудови мережі ДТЕК.

Застосування методу VLSM дозволяє проводити розрахунок схеми IP-адресації з використанням ефективного підходу. Цей метод дозволяє гнучко використовувати адресний простір для мереж різного розміру, шляхом розбиття підмереж на додаткові підмережі. В результаті, роутери отримують інформацію про маршрутизацію, включаючи IP-адресу мережі та маску підмережі, яка вказує кількість бітів, які використовуються для адресації IP-адреси. Цей підхід дозволяє ефективно використовувати ресурси адресного простору та забезпечує гнучкість при плануванні і розгортанні мережі будь-якого розміру.

Таблиця 3.1 – Мінімальна кількість хостів

Адреса	LAN1	LAN2	LAN3	LAN4	LAN5
172.23.0.0/21	15	75	75	90	21

Таблиця 3.1 демонструє, що ця конфігурація показує, що топологія мережі об'єднує п'ять мереж із вузловими пристроями, п'ять мереж з маршрутизаторами, а також зовнішню мережу шлюзу з визначеною мережевою адресою 172.23.0.0/21 . Потрібні дві IP-адреси для маршрутизатора та зовнішнього шлюзу.

Визначення необхідної кількості хостів для кожної підмережі

Для кожної підмережі нам знадобиться достатня кількість хостів для вузлів, а також адреса мережі та широкомовного адресу. Зазвичай кількість хостів визначається за формулою 2^n - 2, де n - кількість бітів, виділених під хостів у масці підмережі.

LAN1: 15 вузлів -> 2^5-2 = 30 хостів (необхідно 5 біта)

LAN2: 75 вузлів -> 2^7-2 = 126 хостів (необхідно 7 бітів)

LAN3: 75 вузлів -> 2^7-2 = 126 хостів (необхідно 7 бітів)

LAN4: 90 вузлів -> 2^7-2 = 126 хостів (необхідно 7 бітів)

LAN5: 21 вузол -> 2^5 -2= 30 хоста (необхідно 5 бітів)

Виділення підмереж

Тепер, коли ми знаємо, скільки бітів нам потрібно для кожної підмережі, ми можемо розбити IP-адресу 172.23.104.0/21 на відповідні підмережі:

LAN1:

Маска підмережі: /27 (оскільки 32 - 5 біти = 27)

Діапазон IP-адрес: 172.23.1.161 - 172.23.1.190

LAN2:

Маска підмережі: /25 (оскільки 32 - 7 бітів = 25)

Діапазон IP-адрес: 172.23.0.129 - 172.23.0.254

LAN3:

Маска підмережі: /25 (оскільки 32 - 7 бітів = 25)

Діапазон IP-адрес: 172.23.1.1 - 172.23.1.126

LAN4:

Маска підмережі: /25 (оскільки 32 - 7 бітів = 25)

Діапазон IP-адрес: 172.23.0.1 - 172.23.0.126

LAN5:

Маска підмережі: /26 (оскільки 32 - 5 бітів = 27)

Діапазон IP-адрес: 172.23.1.129 - 172.23.1.158

Таблиця 3.2 − Схема адресації мережі

Назва мережі	Кількість вузлів	Номер мережі	Адреса підмережі	Маска мережі	Початкове значення діапазону можливих адрес вузлів у підмережі	Кінцеве значення діапазону можливих адрес вузлів у підмережі
LAN 1	15	1	172.23.1.160	255.255.255.224 /27	172.23.1.161	172.23.1.190
LAN 2	75	2	172.23.0.128	255.255.255.128 /25	172.23.0.129	172.23.0.254
LAN 3	75	3	172.23.1.0	255.255.255.128 /25	172.23.1.1	172.23.1.126
LAN 4	90	4	172.23.0.0	255.255.255.128 /25	172.23.0.1	172.23.0.126
LAN 5	30	5	172.23.1.128	255.255.255.224 /27	172.23.1.129	172.23.1.158

Таблиця 3.3 − Адресації WAN

Назва мережі	Кількість вузлів	Номер мережі	Адреса підмережі	Маска мережі	Початкове значення діапазону можливих адрес вузлів у підмережі	Кінцеве значення діапазону можливих адрес вузлів у підмережі
WAN 1	2	1	10.1.8.0	255.255.255.0 /24	10.1.8.1	10.1.8.2
WAN 2	2	2	10.2.8.0	255.255.255.0 /24	10.2.8.1	10.2.8.2
WAN 3	2	3	10.3.8.0	255.255.255.0 /24	10.3.8.1	10.3.8.2
WAN 4	2	4	10.4.8.0	255.255.255.0 /24	10.4.8.1	10.4.8.2
WAN 5	2	5	10.5.8.0	255.255.255.0 /24	10.5.8.1	10.5.8.2
WAN 6	2	6	209.165.202.0	255.255.255.252 /30	209.165.202.1	209.165.202.2
WAN 7	2	7	64.100.13.0	255.255.255.252 /30	64.100.13.1	64.100.13.2

Таблиця 3.4 − VLAN

Назва мережі	Кількість вузлів	Номер мережі	Адреса підмережі	Маска мережі	Початкове значення діапазону можливих адрес вузлів у підмережі	Кінцеве значення діапазону можливих адрес вузлів у підмережі
VLAN 18	25	1	172.23.0.128	255.255.255.224 /27	172.23.0.129	172.23.0.158
VLAN 28	25	2	172.23.0.160	255.255.255.224 /27	172.23.0.161	172.23.0.190
VLAN 38	25	3	172.23.0.192	255.255.255.224 /27	172.23.0.193	172.23.0.222
VLAN 99	1	99	172.23.0.224	255.255.255.248 /29	172.23.0.225	172.23.0.230

3.2 Розрахунок схеми адресації пристроїв

Відповідно до вимог, треба створити адресний простір для пристроїв

IP-адреси можна розподілити наступним чином:

- перші доступні IP-адреси можна призначити інтерфейсам та під-інтерфейсам маршрутизаторів у локальній мережі (LAN).
- наступні доступні IP-адреси можна призначити комутаторам у локальній мережі (LAN).
- останні доступні IP-адреси можна призначити вузлам, тобто комп'ютерам або іншим пристроям, які підключені до мережі.

Таблиця 3.5 – Схема адресації маршрутизаторів

Пристрій	Інтерфейс		IP-адреса	Маска	Шлюз	VLAN	Інтерфейс підключеного пристрою
Konopkin_Router_1	Gig0/0/0		10.5.8.2	255.255.255.0/24	-	-	Gig0/0/0
	Gig0/0/1		172.23.0.1	255.255.255.0/24	-	-	Gig0/1
	Se0/1/0		10.3.8.2	255.255.255.0/24	-	-	Se0/1/1
	Se0/1/1		10.2.8.2	255.255.255.0/24	-	-	Se0/2/0
Konopkin_Router_2	Gig0/0/0	18	172.23.0.129	255.255.255.224/27	-	18	Gig0/1
		28	172.23.0.161	255.255.255.224/27	-	28	Gig0/1
		38	172.23.0.193	255.255.255.224 /27	-	38	Gig0/1
		99	172.23.0.225	255.255.255.248 /29	-	99	Gig0/1
	Gig0/0/1		172.23.1.129	255.255.255.224 /27	-	-	Gig0/1
	Se0/1/0		10.1.8.2	255.255.255.0 /24	-	-	Se0/2/0
	Se0/1/1		10.3.8.1	255.255.255.0 /24	-	-	Se0/1/0
	Se0/2/0		10.2.8.1	255.255.255.0 /24	-	-	Se0/1/1

Продовження таблиці 3.5

Konopkin_Router_3	Gig0/0/0	10.5.8.1	255.255.255.0/24	-	-	Gig0/0/0
	Gig0/0/1	172.23.1.1	255.255.255.128/25	-	-	Gig0/1
	Se0/1/0	10.4.8.2	255.255.255.0/24	-	-	Se0/1/1
Konopkin_Router_4	Se0/1/0	209.165.202.2	255.255.255.252/30	-	-	Se0/1/0
	Se0/1/1	10.4.8.1	255.255.255.0/24	-	-	Se0/1/0
	Se0/2/0	10.1.8.1	255.255.255.0 /24	-	-	Se0/1/0
Konopkin_Router_5	Gig0/0/0	64.100.13.2	255.255.255.252 /30	-	-	Gig0/0/1
	Gig0/0/1	172.23.1.161	255.255.255.224 /27	-	-	Gig0/1
ISP	Gig0/0/0	209.165.201.1	255.255.255.240 /28	-	-	Gig0/1
	Gig0/0/1	64.100.13.1	255.255.255.252 /30	-	-	Gig0/0/0
	Se0/1/0	209.165.202.1	255.255.255.252 /30	-	-	Se0/1/0

Таблиця 3.6 – Схема адресації комутаторів

Пристрій	Інтерфейс	IP-адреса	Маска	Шлюз	VLAN	Інтерфейс підключеного пристрою
Konopkin_Switch_1	VLAN99	172.23.0.226	255.255.255.248 /29	172.23.0.225	99	-
Konopkin_Switch_2	VLAN99	172.23.0.227	255.255.255.248 /29	172.23.0.225	99	-
Konopkin_Switch_3	VLAN99	172.23.0.228	255.255.255.248 /29	172.23.0.225	99	-
Konopkin_Switch_4	VLAN1	172.23.1.130	255.255.255.224 /27	172.23.1.129	1	-
Konopkin_Switch_5	VLAN1	172.23.0.2	255.255.255.128 /25	172.23.0.1	1	-
Konopkin_Switch_6	VLAN1	172.23.1.2	255.255.255.128 /25	172.23.1.1	1	-
Konopkin_Switch_7	VLAN1	172.23.1.162	255.255.255.224 /27	172.23.1.161	1	-
Konopkin_Switch_8	VLAN1	172.23.1.163	255.255.255.224 /27	172.23.1.161	1	-
Konopkin_Switch_9	VLAN1	172.23.1.164	255.255.255.224 /27	172.23.1.161	1	-

Таблиця 3.7 – Схема адресації кінцевих вузлів

Назва мережі	Кількість вузлів	Номер мережі	Адреса підмережі	Маска мережі	Початкове значення діапазону можливих адрес вузлів у підмережі	Кінцеве значення діапазону можливих адрес вузлів у підмережі
PC37-39,PC41-47	NIC	172.23.1.162 - 172.23.1.190	255.255.25.224 /27	172.23.1.160	1	Fa0/5-8
PC0-1,PC50-51, PrinterHp_Konopkin_1, C5540i_1	NIC	172.23.0.130 - 172.23.0.158	255.255.255.224 /27	172.23.0.129	18	Fa0/15-17
PC38,PC40,PC52,PC53, PrinterHp_Konopkin_2, C5540i_2	NIC	172.23.0.162 - 172.23.0.190	255.255.255.224 /27	172.23.0.161	28	Fa0/10-12
PC48-49,PC54-55, PrinterHp_Konopkin_3, C5540i_3	NIC	172.23.0.194 - 172.23.0.222	255.255.255.224 /27	172.23.0.193	38	Fa0/5-7

Кінець таблиці 3.7

PC27-36, Server AAA, Server HTTP	NIC	172.23.1.3 - 172.23.1.126	255.255.255.128 /25	172.23.1.1	1	Fa0/1-12
PC12-26	NIC	172.23.0.3 - 172.23.0.126	255.255.255.128 /25	172.23.0.1	1	Fa0/1-15
PC2-11, Server TFTP	NIC	172.23.1.131 - 172.23.1.158	255.255.255.224 /27	172.23.1.129	1	Fa0/1-11

3.3 Розробка схеми логічної топології корпоративної мережі

У рамках розробки логічної топології корпоративної мережі була обрана топологія зірка. Топологія зірка характеризується центральним вузлом, який підключений до кожного з вузлів мережі. Кожна робоча станція підключається до комутатора за допомогою Ethernet-кабелів. Таким чином, кожна робоча станція має пряме з'єднання з центральним комутатором. Це забезпечує швидку передачу даних та низький рівень латентності в мережі.

Також важливо відзначити, що мережевий комутатор має підтримку VLAN (віртуальних локальних мереж), що дозволяє логічно розділити мережу на групи ізольованих пристроїв. Це дозволяє забезпечити безпеку та ефективну організацію мережевого трафіку.

Топологія зірка є дуже надійною, оскільки в разі відмови одного пристрою, інші вузли мережі продовжують працювати незалежно. Це сприяє високій доступності та мінімізації впливу відмов на роботу мережі. Таким чином, обрана топологія зірка забезпечує ефективну, надійну та гнучку структуру корпоративної мережі.

На рисунку 3.1 можна побачити розроблену логічну топологію.

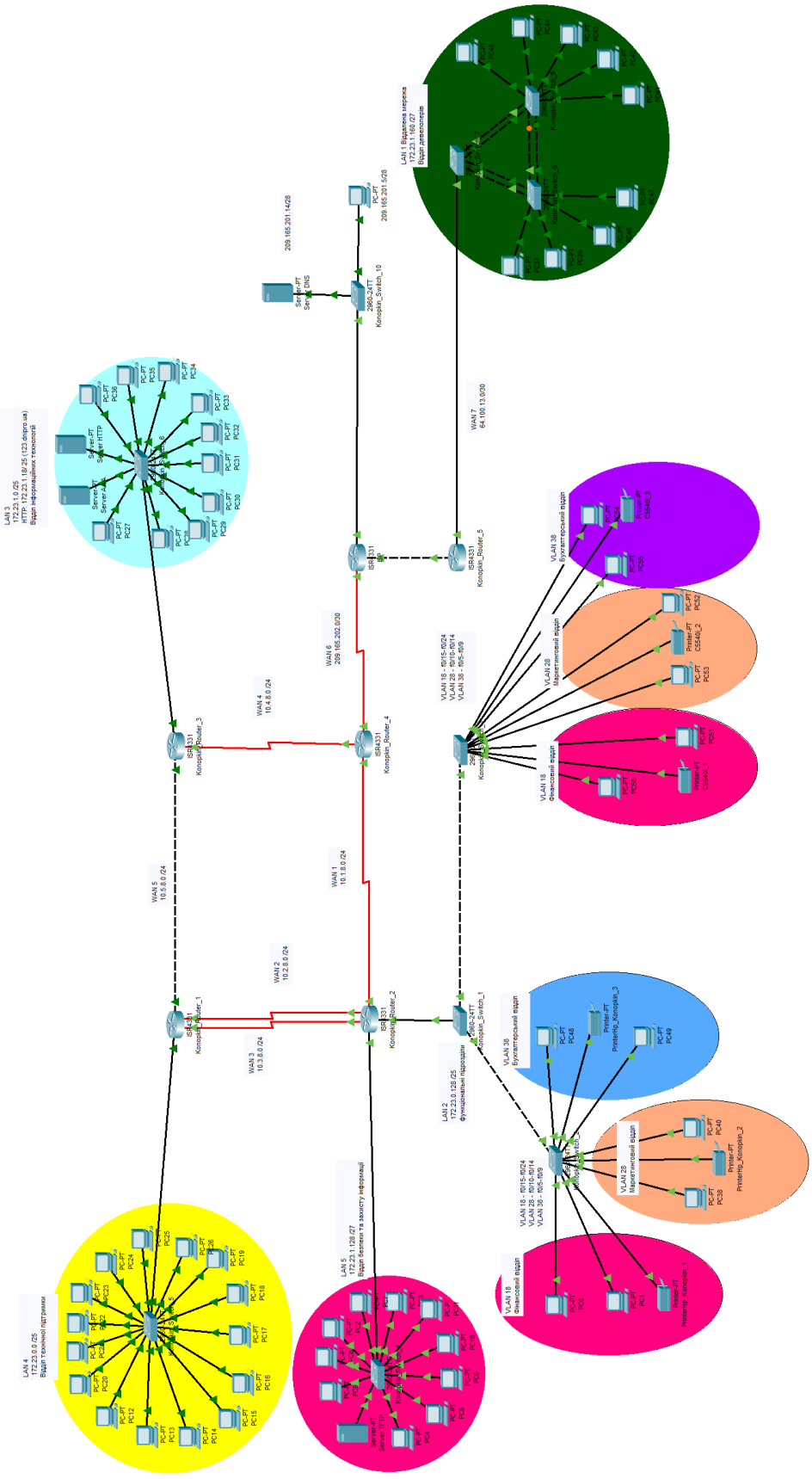

Рисунок 3.1 – Логічна топологія корпоративної мережі

3.4 Розробка схеми фізичної топології корпоративної мережі

Проектована мережа інтегрує головну будівлю в одну злагоджену мережеву систему.

Фізична топологія складається з двох сегментів: головна будівля з двома поверхами.

На першому поверсі головної будівлі розташовані функціональні відділення, серверна та технічний відділ.

Для цих відділів мережа є необхідною для забезпечення ефективної комунікації та спільного доступу до ресурсів.

Фінансовий відділ використовує мережу для обміну фінансовою інформацією, здійснення фінансових операцій, ведення обліку та контролю за фінансовими процесами. Вони можуть використовувати спеціалізоване програмне забезпечення та бази даних, які потребують мережевого доступу для обміну даними та зберігання інформації.

Бухгалтерія також залежить від мережі для проведення бухгалтерського обліку, операцій з оплати, створення звітів та зберігання фінансової інформації. Вони можуть використовувати спеціальне бухгалтерське програмне забезпечення, яке потребує мережевого доступу для обміну даними та спільної роботи над фінансовими документами.

Маркетинговий відділ використовує мережу для проведення маркетингових досліджень, аналізу ринку, планування маркетингових кампаній та спільної роботи над маркетинговими матеріалами. Вони можуть використовувати спеціалізовані маркетингові інструменти та ресурси, які вимагають мережевого доступу для обміну даними та спільної роботи над проектами.

Технічний відділ з серверною використовує мережу для керування та підтримки інформаційної інфраструктури підприємства. Вони відповідають за налагодження та підтримку серверів, мережевих пристроїв, забезпечення безпеки та контролю доступу до мережевих ресурсів. Мережа дозволяє їм віддалено керувати та моніторити різні системи, виконувати резервне копіювання даних та забезпечувати високу доступність технічної інфраструктури. Також у них знаходиться сервер HTTP та AAA використовується для забезпечення централізованого керування та контролю доступу користувачів до мережевих ресурсів.

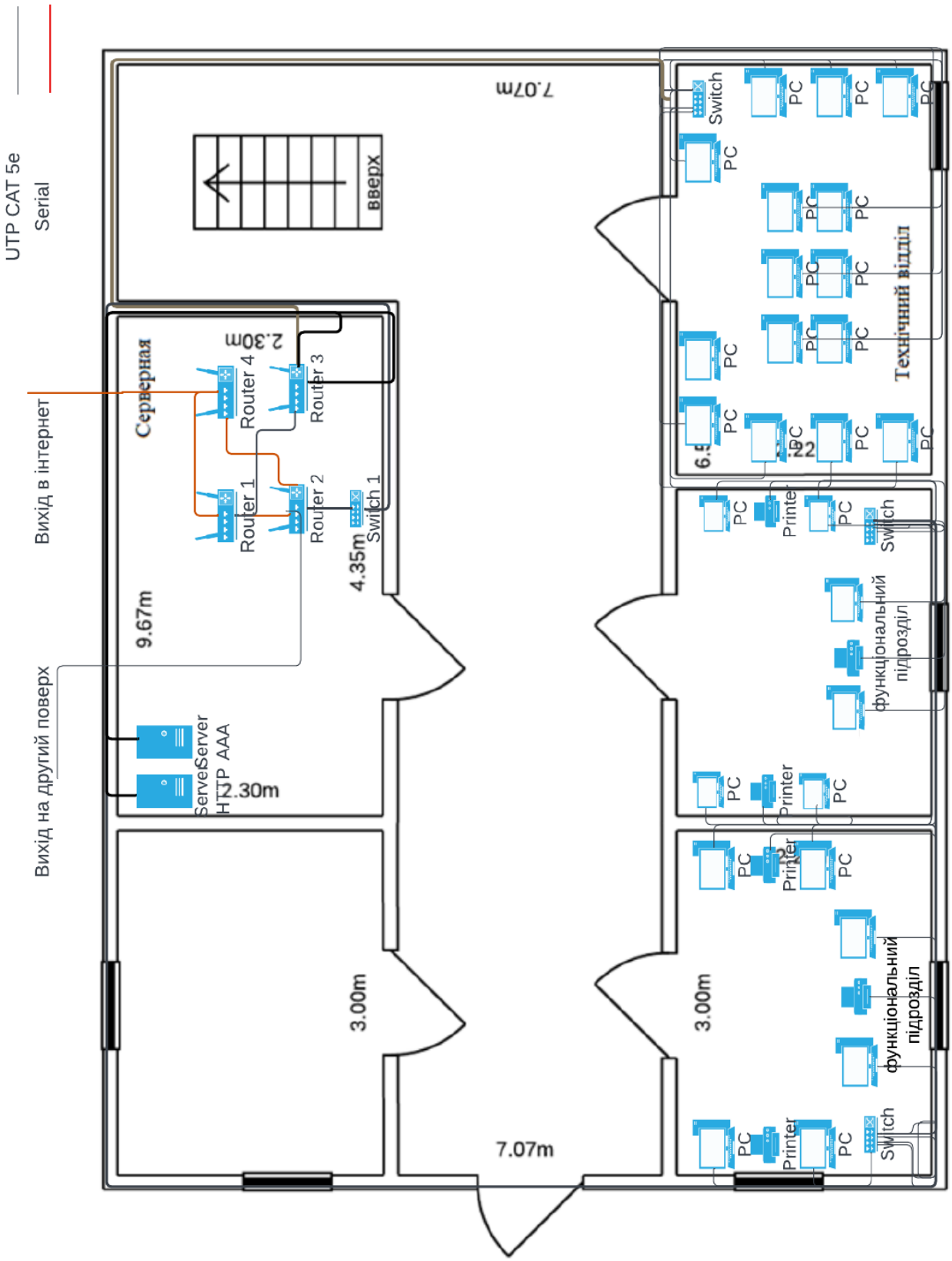

Рисунок 3.2 – Фізична схема першого поверху

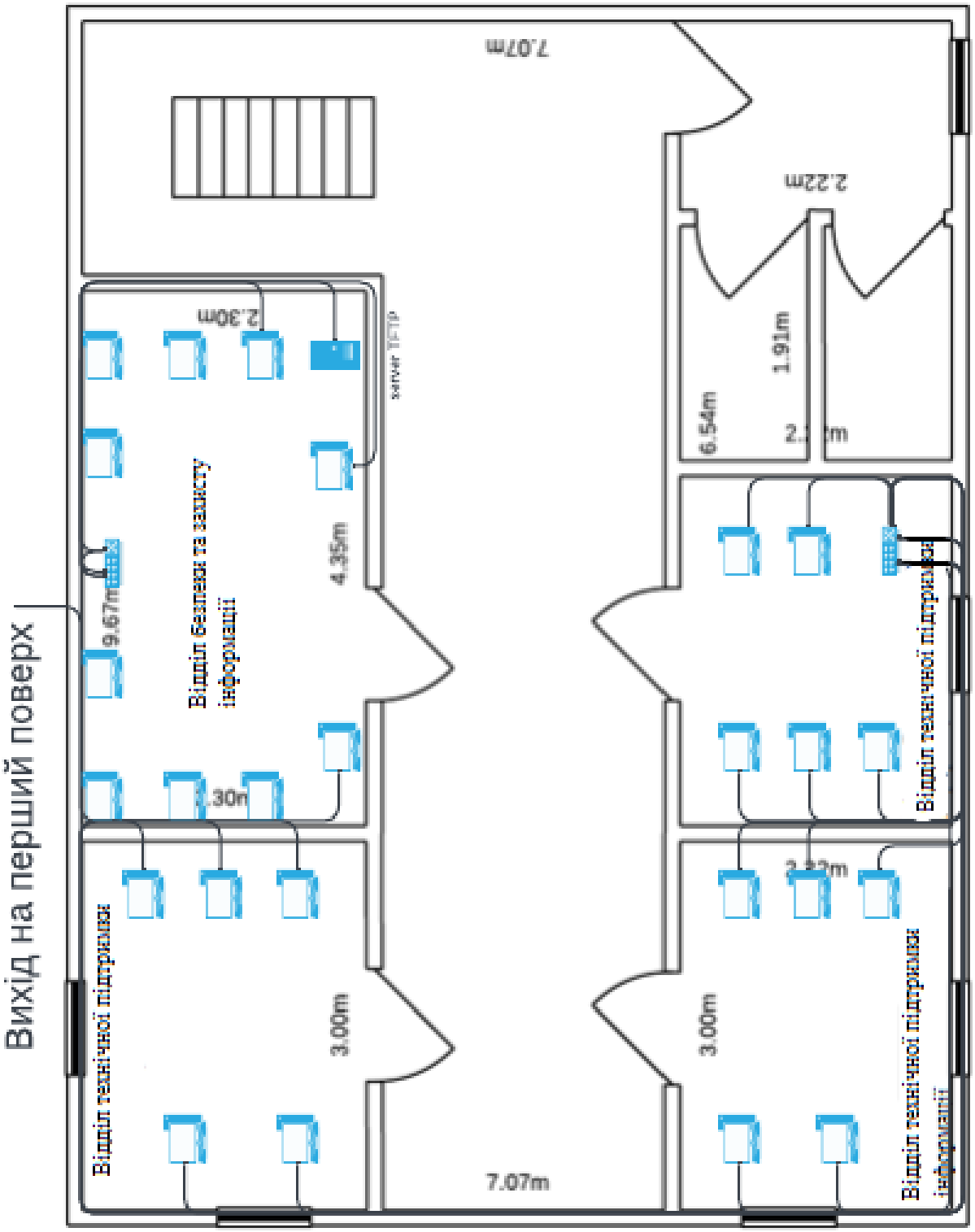

Рисунок 3.3 – Фізична схема другого поверху

3.5 Налаштування та перевірка роботи комп’ютерної мережі

3.5.1Базове налаштування конфігурації пристроїв

Для виконання базового налаштування конфігурації пристроїв необхідно:

– назначити назви пристроям за наступним правилом: Konopkin_тип пристрою_номер пристрою, наприклад, Konopkin_Router_1;

– на всіх пристроях назначити пароль cisco до консолі і vty;

– на всіх пристроях назначити пароль class до привілейованого режиму;

– усі паролі, що зберігаються у відкритому вигляді, пропонується під час налаштування моделі комп’ютерної системи зашифрувати;

– розробити банер MOTD;

– назначити на усіх лініях vty використання протоколу ssh;

– призначити на всіх пристроях користувача 12320sk1_Konopkin, з паролем admincisco;

– в якості імені доменна використати ім'я пристрою. Для шифрування даних створювати ключ RSA завдовжки 1024 біт.

Наступними команди ми проводимо базове налаштування мережевих пристроїв:

```
Router(config)#hostname ISP
ISP(config)#ip domain-name ISP
ISP(config)#enable password class
ISP(config)#line console 0
ISP(config-line)#password cisco
ISP(config-line)#login
ISP(config-line)#exit
ISP(config)#line vty 0 15
ISP(config-line)#password cisco
ISP(config-line)#login
ISP(config-line)#exit
```

ISP(config)#service password-encryption

ISP(config)#banner motd #Hope!#

ISP(config)#username 12320sk1_Konopkin password admincisco

ISP(config)#crypto key generate rsa

How many bits in the modulus [512]: 1024

ISP(config)#line vty 0 15

ISP(config-line)#transport in ssh

ISP(config-line)#login local

ISP(config-line)#exit

Наступні налаштування тільки для роутерів:

ISP(config)#interface Serial0/1/0

ISP(config-if)#clock rate 128000

Встановлення IP-адрес на інтерфейсі:

ISP(config)#interface Serial0/1/0

ISP(config-if)#ip address 209.165.202.1 255.255.255.252

ISP(config-if)#no sh

Ці налаштування виконуються на всіх інших маршрутизаторів з використання таблиці 3.5

3.5.3 Налаштування DHCP

Пул DHCP - це діапазон IP-адрес, які можуть бути надані клієнтам через DHCP.

ip dhcp pool poollan1: ця команда створює новий пул DHCP з ім'ям "poollan1".

network 172.23.1.160 255.255.255.224: ця команда визначає мережу,172.23.1.160/27, яка буде використовуватись в пулі DHCP.

default-router 172.23.1.161: ця команда вказує IP-адресу маршрутизатора, яка буде використовуватись як "маршрутизатор за замовчуванням" для клієнтів, що отримують IP-адреси через DHCP.
dns-server 209.165.201.14: ця команда вказує IP-адресу DNS-сервера, яка буде надана клієнтам через DHCP
ip dhcp excluded-address 172.23.1.161 172.23.1.170: ця команда виключає діапазон IP-адрес з розподілу через DHCP. В даному випадку, IP-адреси від 172.23.1.161 до 172.23.1.170 будуть виключені з пулу DHCP і не будуть надані клієнтам.

3.5.2 Налаштування маршрутизації

У якості протоколу маршрутизації ми будемо використовувати OSPF.

OSPF (Open Shortest Path First) - це протокол маршрутизації в комп'ютерних мережах, що використовується для вибору оптимального шляху передачі даних в мережі, зокрема в IP-мережах. Він є одним із найпоширеніших протоколів внутрішньої шлюзової маршрутизації.

Для налаштування маршрутизації будуть використовуватись такі команди:

Konopkin_Router_2(config)#route ospf 1: Ця команда вмикає протокол OSPF з ідентифікатором процесу 1 на маршрутизаторі Konopkin_Router_2.

Konopkin_Router_2(config-router)#network 10.3.8.0 0.0.0.255 area 0: Ця команда додає мережу 10.3.8.0/24 в область 0 OSPF. Це означає, що маршрутизатор буде відомий про цю мережу та обмінюватись інформацією про неї з іншими маршрутизаторами, які також працюють з OSPF в області 0.
Konopkin_Router_2(config-router)#network 10.2.8.0 0.0.0.255 area 0

Konopkin_Router_2(config-router)#network 10.1.8.0 0.0.0.255 area 0

Konopkin_Router_2(config-router)#network 172.23.0.128 0.0.0.127 area 0

Konopkin_Router_2(config-router)#network 172.23.1.128 0.0.0.31 area 0

Konopkin_Router_2(config-router)#passive-interface gig0/0/0: Ця команда встановлює інтерфейс gig0/0/0 в пасивний режим. Це означає, що OSPF не буде використовувати цей інтерфейс для вимірювання маршрутів.
Konopkin_Router_2(config-router)#passive-interface gig0/0/1

Налаштування статичної маршрутизації.
ISP(config)#ip route 209.165.200.0 255.255.255.0 209.165.202.2

3.5.4 Налаштування агрегації каналів на комутаторах

Konopkin_Switch_8(config)#int range fa0/1-2: ця команда вибирає діапазон портів Fast Ethernet 0/1-2 для налаштування.

Konopkin_Switch_8(config-if-range)#switchport mode trunk: ця команда встановлює режим портів в режим "trunk". Режим "trunk" дозволяє передавати дані між комутаторами за допомогою VLAN-тегів, що дозволяє створювати транкінг-з'єднання (trunk link).

Konopkin_Switch_8(config-if-range)#channel-group 1 mode active: ця команда створює агреговану групу портів (port-channel) з портів Fast Ethernet 0/1-2 і встановлює режим "active". Режим "active" вказує, що група портів активна і готова передавати дані.
Konopkin_Switch_8(config)#int range fa0/3-4

Konopkin_Switch_8(config-if-range)#switchport mode trunk

Konopkin_Switch_8(config-if-range)#channel-group 2 mode active

Робимо теж саме шо і з switch 8, але тепер на 7 та 9.
Konopkin_Switch_9(config-if-range)#channel-group 3 mode passive: цією командою створюємо агреговану групу портів (port-channel) з портів Fast Ethernet 0/1-2 і встановлює режим "passive". Режим "passive" вказує, що група портів працює у пасивному режимі, тобто вона слухає та приймає дані, але не посилає їх активно.

3.5.5 Налаштування VLAN

Switch 1, 2, 3:

Наступні команди виконують наступні дії на комутаторі Konopkin_Switch_2 і Konopkin_Switch_1:
Konopkin_Switch_2(config)#int fa0/1: Вибір інтерфейсу Fast Ethernet 0/1 для налаштування.
Konopkin_Switch_2(config-if)#switchport mode trunk: Встановлення режиму порта в режим "trunk".
Konopkin_Switch_2(config)#vlan 18: Створення VLAN з ідентифікатором 18.
Konopkin_Switch_2(config-vlan)#exit: Вихід з режиму конфігурації VLAN.
Konopkin_Switch_2(config)#int range fa0/15-24: Вибір діапазону портів Fast Ethernet 0/15-24 для налаштування.
Konopkin_Switch_2(config-if-range)#switchport mode access: Встановлення режиму порта в режим "access".
Konopkin_Switch_2(config-if-range)#switchport access vlan 18: Призначення VLAN 18 для портів в режимі "access".
Konopkin_Switch_2(config)#vlan 28: Створення VLAN з ідентифікатором 28.
Konopkin_Switch_2(config-vlan)#exit: Вихід з режиму конфігурації VLAN.
Аналогічні кроки повторюються для VLAN 28 і портів Fast Ethernet 0/10-14 (в режимі "access") та VLAN 38 і портів Fast Ethernet 0/5-9 (в режимі "access").

Konopkin_Switch_2(config)#ip default-gateway 172.23.0.225: Встановлення IP-адреси шлюза за замовчуванням для комутатора.

Konopkin_Switch_1:

vlan 99: Створення VLAN з ідентифікатором 99.

int vlan 99: Вибір інтерфейсу VLAN 99 для налаштування.

ip address 172.23.0.226 255.255.255.248: Надання IP-адреси та маски підмережі для інтерфейсу VLAN 99.

no shutdown: Увімкнення інтерфейсу VLAN 99.

vlan 100: Створення VLAN з ідентифікатором 100.

int range fa0/1-2,gig0/1: Вибір діапазону портів Fast Ethernet 0/1-2 та Gigabit Ethernet 0/1 для налаштування.

Konopkin_Switch_1(config-if-range)#switchport trunk native vlan 100: Встановлення VLAN 100 як VLAN за замовчуванням для портів у режимі "trunk".

Konopkin_Switch_1(config-if-range)#switchport trunk allowed vlan 18,28,38,99,100: Встановлення списку дозволених VLAN для портів у режимі "trunk", де дозволені VLAN вмикати VLAN 18, 28, 38, 99 і 100.

Наступні налаштування було виконано на router 2

Konopkin_Router_2(config)#int gig0/0/0.18

Konopkin_Router_2(config-subif)#encapsulation dot1Q 18

Konopkin_Router_2(config-subif)#ip address 172.23.0.129 255.255.255.224

Konopkin_Router_2(config-if)#exit

Konopkin_Router_2(config)#int gig0/0/0.28

Konopkin_Router_2(config-subif)#encapsulation dot1Q 28

Konopkin_Router_2(config-subif)#ip address 172.23.0.161 255.255.255.224

Konopkin_Router_2(config-subif)#exit

Konopkin_Router_2(config)#int gig0/0/0.38

Konopkin_Router_2(config-subif)#encapsulation dot1Q 38

Konopkin_Router_2(config-subif)#ip address 172.23.0.193 255.255.255.224

Konopkin_Router_2(config-subif)#exit

Konopkin_Router_2(config)#int gig0/0/0.99

Konopkin_Router_2(config-subif)#encapsulation dot1Q 99

Konopkin_Router_2(config-subif)#ip address 172.23.0.225 255.255.255.248

Ці команди налаштовують маркування кадрів з використанням протоколу 802.1Q на підінтерфейсах маршрутизатора та призначають їм відповідні IP-адреси та маски підмережі. Кожен підінтерфейс використовує окремий ідентифікатор VLAN для відокремлення трафіку між різними VLAN-ами у мережі.

3.5.6 Налаштування динамічного NAT

Спочатку потрібно створити список доступу 100 (access-list 100) для визначення трафіку, який потрібно перетворити за допомогою NAT:

create access-list 100:

ip access-list extended 100: вхід до режиму налаштування розширеного списку доступу 100.
deny ip 172.23.0.0 0.0.255.255 172.23.1.160 0.0.0.31: заборона перетворення трафіку від підмережі 172.23.0.0/16 до підмережі 172.23.1.160/27.
permit ip 172.23.0.0 0.0.255.255 any: дозвіл перетворення всього іншого трафіку з підмережі 172.23.0.0/16.
створення пула NAT з назвою "Internet" та визначення діапазону IP-адрес для перетворення:

ip nat pool Internet 209.165.200.5 209.165.200.30 netmask 255.255.255.0: створення пула NAT з іменем "Internet" та визначення діапазону IP-адрес від 209.165.200.5 до 209.165.200.30 з маскою підмережі 255.255.255.0.

Налаштування перетворення NAT для трафіку, що відповідає списку доступу 100:

ip nat inside source list 100 pool Internet: Налаштування перетворення NAT з використанням списку доступу 100 та пула "Internet".

Налаштування типу NAT на вхідних та вихідних інтерфейсах:

int se0/1/0: Вибір інтерфейсу Serial 0/1/0 для налаштування.

ip nat outside: Налаштування інтерфейсу як зовнішнього для NAT.

Налаштування статичного NAT

Konopkin_Router_4(config)#ip nat inside source static 172.23.1.18 209.165.200.4

Команда встановлює статичне перетворення NAT для внутрішньої IP-адреси 172.23.1.18 на зовнішню IP-адресу 209.165.200.4. Це означає, що будь-який трафік, що відправляється з внутрішньої IP-адреси 172.23.1.18, буде перетворений на зовнішню IP-адресу 209.165.200.4 при проходженні через маршрутизатор.

Статичний NAT використовується, коли потрібно постійно перетворювати одну конкретну внутрішню IP-адресу на певну зовнішню IP-адресу. В даному випадку, внутрішня IP-адреса буде завжди перетворюватись на зовнішню IP-адресу незалежно від напрямку трафіку (вхідного або вихідного).

3.5.7 Налаштування сервісу AAA

Налаштувати всі маршрутизатори на підтримку служби AAA необхідно таким чином:

− для перевірки підключень до VTY ліній на маршрутизаторі використовувати локальну базу даних користувачів;

− для доступу до консолі використовувати аутентифікацію на основі протоколу RADIUS і якщо немає − локальну базу даних;

− RADIUS-сервер налаштувати наступним чином: ключове слово – radius123; в якості облікового запису користувачів використовувати ім’я пристрою з паролем admin123

Перейдемо до налаштування:

aaa new-model: ця команда ввімкнутиє модель AAA (Authentication, Authorization, Accounting) на маршрутизаторі. AAA забезпечує централізоване керування аутентифікацією, авторизацією та обліком для користувачів, що намагаються отримати доступ до мережевих ресурсів.

aaa authentication login default local: ця команда встановлює тип аутентифікації "local" (локальна аутентифікація) за замовчуванням для режиму входу (console) на маршрутизатор. Це означає, що користувачі будуть перевірятися за допомогою локально збережених облікових даних на маршрутизаторі.

aaa authentication login Radius_list group radius local: ця команда створює список аутентифікації з назвою "Radius_list" і вказує, що для цього списку аутентифікації будуть використовуватися методи "radius" та "local". Це означає, що спочатку буде спробувана аутентифікація за допомогою сервера RADIUS, а якщо вона не вдасться, використовуватимуться локальні облікові дані.

line console 0: ця команда переходить до налаштування параметрів лінії консолі (консольного порту).

login authentication Radius_list: ця команда встановлює тип аутентифікації "Radius_list" для лінії консолі. Це означає, що користувачі, які намагаються увійти через консольний порт, будуть перевірятися спочатку за допомогою сервера RADIUS, а якщо вона не вдасться, використовуватимуться локальні облікові дані.

line vty 0 15: Ця команда переходить до налаштування параметрів віртуальних терміналів (VTY).
login authentication default: Ця команда встановлює тип аутентифікації "default" для віртуальних терміналів. Це означає, що користувачі, які намагаються увійти через віртуальний термінал, будуть перевірятися за допомогою локально збережених облікових даних на маршрутизаторі.

3.5.8 Налаштування VPN тунель

Компанія використовує VPN з метою забезпечення безпечного та приватного з'єднання між головною мережею та віддаленими відділеннями.

Ці кроки налаштовують VPN на маршрутизаторі з використанням протоколів ISAKMP і IPsec. Вони визначають політику, ключі, набори трансформацій і правила, які будуть застосовуватися до IP-трафіку, що проходить через вказаний інтерфейс.
ip access-list extended 102: ця команда створює розширений список доступу з номером 102.
permit ip 172.23.0.0 0.0.255.255 172.23.1.160 0.0.0.31: ця команда встановлює правило в списку доступу 102, яке дозволяє IP-трафіку від мережі 172.23.0.0/16 до підмережі 172.23.1.160/27.
crypto isakmp policy 1: ця команда встановлює політику ISAKMP з пріоритетом 1.
encryption aes: ця команда встановлює шифрування AES для ISAKMP.

authentication pre-share: ця команда встановлює метод аутентифікації з попередньо обміненим ключем для ISAKMP.
crypto isakmp key konopkin address 64.100.13.2: ця команда встановлює ключ із попередньо обміненим ключем для аутентифікації з мережевим пристроєм з IP-адресою 64.100.13.2.

crypto ipsec transform-set Transform_Tag esp-aes esp-sha-hmac: ця команда встановлює набір трансформацій IPsec з шифруванням AES і аутентифікацією SHA-HMAC.

crypto map MAP 1 ipsec-isakmp: ця команда створює криптокарту з назвою "MAP" і прив'язує її до політики IPsec-ISAKMP з пріоритетом 1.

set peer 64.100.13.2: ця команда встановлює IP-адресу пірного мережевого пристрою для криптокарти.

set transform-set Transform_Tag: ця команда встановлює набір трансформацій, який буде використовуватися для шифрування IPsec.

match address 102: ця команда вказує криптокарті збігатися з правилом списку доступу 102.

int se0/1/0: ця команда переходить до налаштування інтерфейсу Serial0/1/0.

crypto map MAP: ця команда надає інтерфейсу криптокарту MAP для застосування IPsec-шифрування на цьому інтерфейсі.

3.5.9 Перевірка роботи комп’ютерної системи

Ми перевіримо, чи отримують кінцеві вузли свої IP-адреси через протокол DHCP.

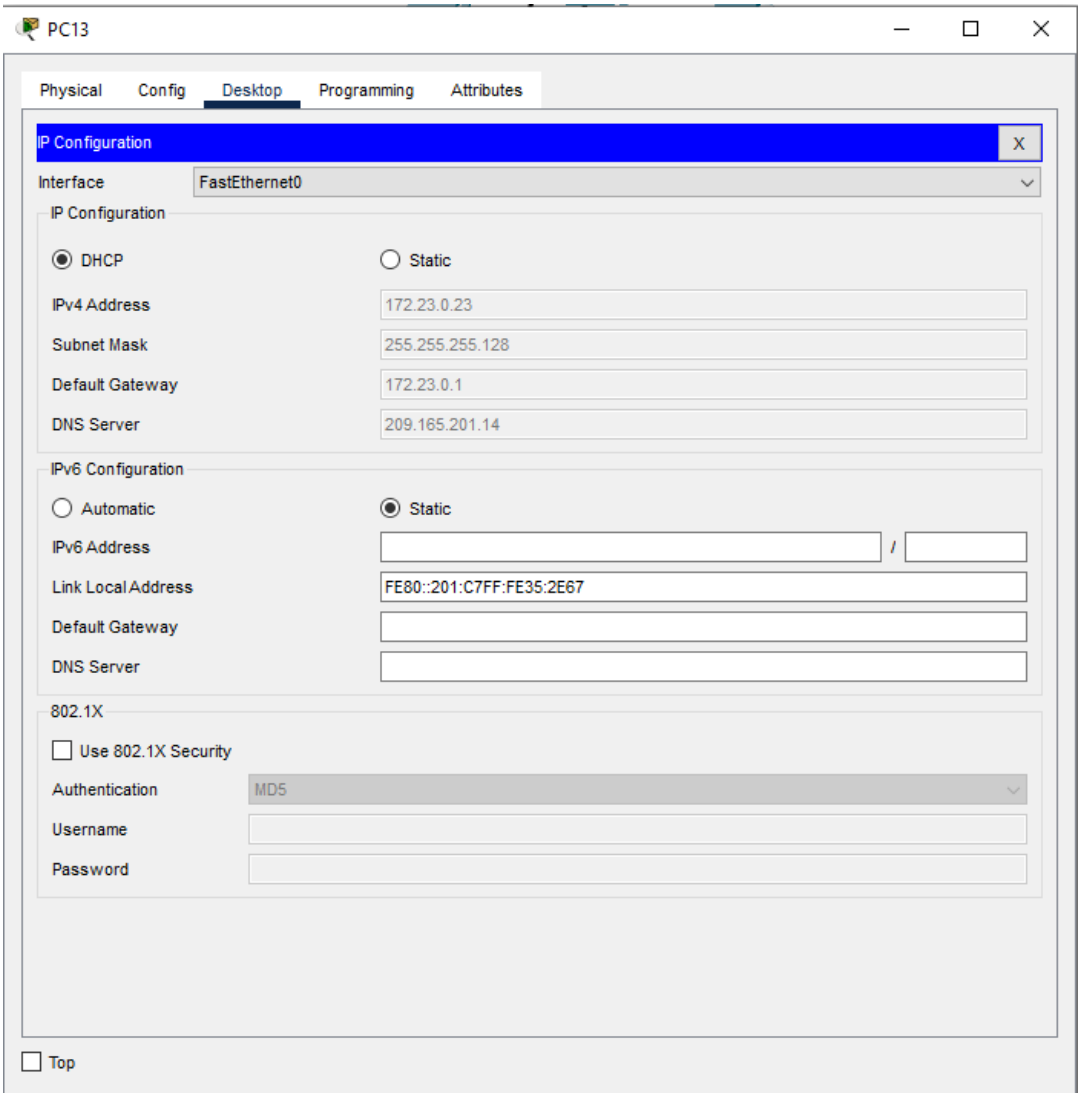

Рисунок 3.6 − Отримання IP-адреси від DHCP кінцевим вузлом

Для перевірки роботи протоколу маршрутизації OSPF перевіримо таблицю на маршрутизаторі Konopkin_Route_3 командою "do show ip route" (рисунок 3.7).

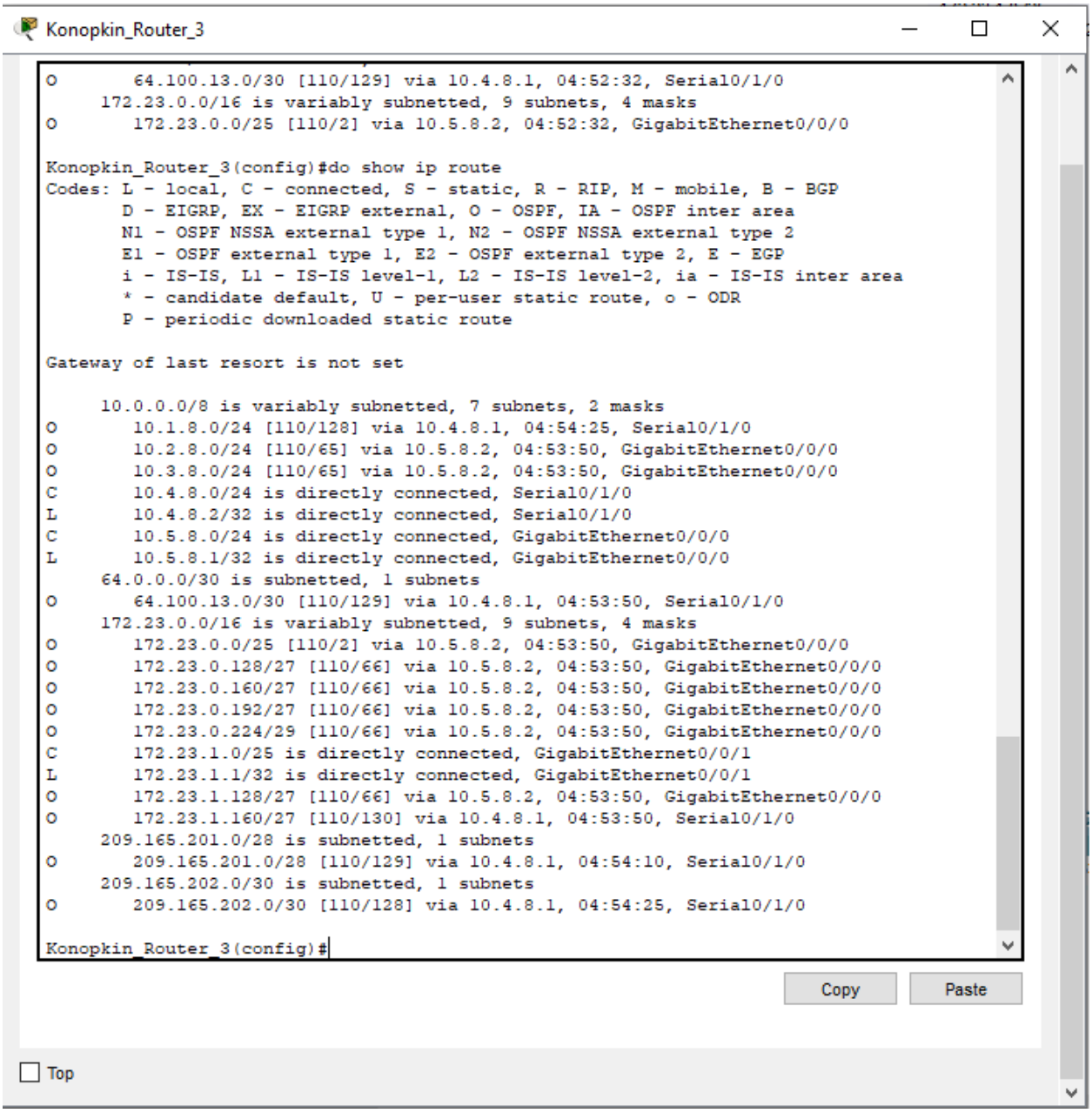

Рисунок 3.7 − Таблиця маршрутизації на маршрутизаторі Konopkin_Router_3

Перевіримо маршрутизацію у корпоративній мережі

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC33	PC16	ICMP		0.000	N	0	(edit)	(delete)
	Successful	PC33	PC11	ICMP		0.000	N	1	(edit)	(delete)
	Successful	PC33	PC37	ICMP		0.000	N	2	(edit)	(delete)

Рисунок 3.8 – Пінгування комп’ютерів

Далі ми спробуємо відправити пакет від ПК ADMIN до ПК провайдера у режимі симуляції, і ми перевіримо, як пакет проходить до маршрутизатора Konopkin_Router_4

Для перевірки трансляції локальної IP-адреси на глобальну адресу, у режимі симуляції відправимо пакет до сервера DNS, та коли він потрапить до маршрутизатора 4, натиснемо на конверт, та перевіримо адресу (Рисунок 3.9)

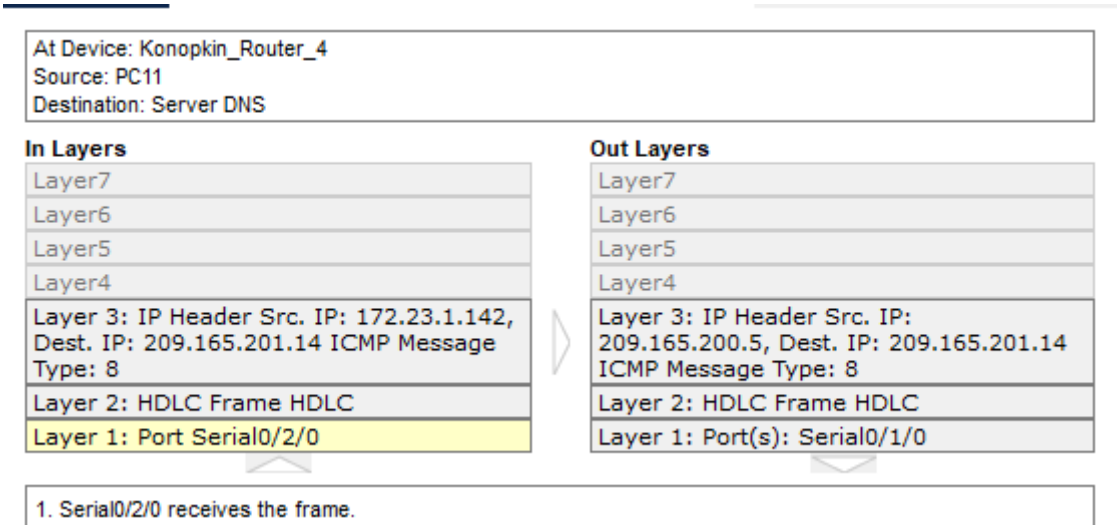

Рисунок 3.9 − Перевірка інформації пакету, який прямує в глобальну мережу

4 РОЗРОБКА МОБЫЛЬНОГО ЗАСТОСУНКУ

4.1 Призначення й сфера застосування програми

Програма призначена для обробки вхідних даних у форматі адреси та генерації вихідного графіку відключення електроенергії.

Однією з основних сфер застосування цієї програми є електроенергетика. Вона може бути використана енергетичними компаніями або людьми міста Дніпра та іншими зацікавленими сторонами. Завдяки програмі, вони можуть аналізувати графік відключення електроенергії для певного регіону чи місцевості.

Крім електроенергетики, програма також може мати застосування в інших галузях, де важлива аналітика та планування відключення електроенергії. Наприклад, вона може бути корисною для промислових підприємств, де необхідно планувати відключення електроенергії для проведення ремонтних робіт або переключення на альтернативні джерела енергії. Також, програма може бути корисною для органів місцевого самоврядування та планування розвитку міст, де необхідно розрахувати та врахувати вплив відключення електроенергії на різні райони та інфраструктуру.

Враховуючи універсальність програми і можливість її використання на всіх операційних системах, вона може бути доступна для широкого кола користувачів.

Отже, програма, що обробляє вхідні дані адреси та генерує графік відключення електроенергії, має широкий спектр застосування, зокрема у сфері електроенергетики, промисловості та міського планування. Її універсальність дозволяє використовувати програму на будь-якій операційній системі, забезпечуючи доступність та гнучкість в її застосуванні.

Опис алгоритму функціонування програми

Розділ описує основний алгоритм та процес взаємодії програми з сервером за допомогою HTTP запитів і передачі даних в форматі JSON. Нижче наведено загальну структуру алгоритму:

1. Встановлення з'єднання з сервером: Додаток встановлює з'єднання з сервером за допомогою відповідного URL-адреси сервера.

2. Формування HTTP запитів: Додаток формує HTTP запити для взаємодії з сервером. Дані, що будуть відправлені на сервер, будуть в якості параметрів URL.

3. Відправка запитів: Додаток відправляє сформований запит на сервер за допомогою HTTP-клієнта, передаючи дані в URL. Для цього використовується відповідний метод GET.

4. Обробка відповіді сервера: Додаток отримує відповідь від сервера у форматі JSON. Відповідь містить різноманітні дані, які можуть бути оброблені додатком. Для цього необхідно розпарсити відповідь та отримати доступ до необхідних даних.

5. Обробка та відображення даних: Додаток обробляє отримані дані з відповіді сервера та візуалізує їх в вигляді графіку відключень на тиждинь.

6. Завершення взаємодії: Після обробки та використання отриманих даних, взаємодія додатку з сервером через HTTP запити може бути завершена, а з'єднання з сервером може бути розірване.

Цей алгоритм дозволяє додатку здійснювати ефективну взаємодію з сервером через Інтернет, отримуючи дані у форматі JSON за допомогою HTTP запитів. Використання такого підходу спрощує передачу та обробку даних між додатком та сервером, забезпечуючи ефективну комунікацію та обмін інформацією.

4.2 Опис розробленої програми

4.2.1 Загальні відомості

Назва програми «Electricity Schedule».

Для розробки та функціонуваня застосунку застосуємо фреймворк Flutter, який має свої апаратні та функціональні вимоги.

Апаратні вимоги:

− процесор: Intel Core i3 або аналогічний процесор з підтримкою набору інструкцій SSE2.

− оперативна пам'ять: Мінімум 4 ГБ, рекомендовано 8 ГБ і більше.

3. Місце на диску: Мінімум 2,8 ГБ вільного місця на диску для інсталяції Flutter SDK та залежностей.

4. Графічна карта: Не вимагається спеціальна графічна карта.

Функціональні вимоги:

1. Операційна система: Flutter підтримує розробку на різних операційних системах, включаючи Windows, macOS та Linux.

2. Редактор коду: Flutter можна використовувати з різними редакторами коду, такими як Android Studio, Visual Studio Code або IntelliJ IDEA.

3. Flutter SDK: Для розробки додатків на Flutter необхідно встановити Flutter SDK, який містить необхідні бібліотеки та інструменти для розробки.

4. Залежності: При розробці додатків на Flutter необхідно мати встановлені необхідні залежності, такі як Dart SDK та різноманітні пакети та бібліотеки, які можуть бути використані в проекті.

5. Екран: Flutter може працювати на різних розмірах екранів, від малих мобільних пристроїв до великих планшетів та десктопних комп'ютерів.

Детальніші вимоги можуть залежати від конкретного проекту та платформи, на якій відбуваєть розробка.

Вимоги до пристрою для запуску програм, написаних на Flutter, можуть варіюватися залежно від типу програми та її функціональності. Однак, нижче

перераховано загальні вимоги до пристроїв, на яких може працювати програма, написана на Flutter ОС Android:

1. Операційна система:

– Android: Android 4.1 (API level 16) або новіша.

– iOS: iOS 9.0 або новіша.

2. Процесор:

– Android: ARM або x86.

– iOS: ARM.

3. Оперативна пам'ять (RAM): Рекомендовано мінімум 2 ГБ оперативної пам'яті для нормальної роботи додатків на Flutter. Програми, які використовують багато ресурсів або мають великий обсяг даних, можуть потребувати більше оперативної пам'яті.

4. Простір на диску: Програми на Flutter зазвичай не вимагають великого обсягу простору на диску. Однак, простір може залежати від розміру самої програми та її додаткових ресурсів (зображень, звуків тощо).

5. Графічна підтримка: Для відображення графічного інтерфейсу програми на Flutter потрібна підтримка графічної бібліотеки операційною системою. Більшість сучасних пристроїв підтримують графічні бібліотеки, необхідні для виконання програм на Flutter.

6. Інтернет-з'єднання: Деякі програми на Flutter можуть вимагати активне Інтернет-з'єднання для отримання даних з мережі або здійснення інших мережевих операцій.

7. Додаткові компоненти: В деяких випадках, для певних функцій або можливостей програми на Flutter може знадобитись певне апаратне або програмне забезпечення, таке як камера, GPS, акселерометр, датчики тощо. Ці вимоги будуть залежати від конкретних функцій програми.

4.3 Опис логічної структури

Програма отримує списки населених пунктів, наприклад регіону міста Дніпра, з зовнішнього API. Після отримання списку з населеними пунктами та вибору користувачем одного з них, програма запитує з серверу список вулиць цього населеного пункту, пусля цього аналогічні дії проводяться для отримання номерів будинків для цієї вулиці. Це все потрібно для отримання категорій типу відключення конкретної адреси (всього їх 3). Для кожної категорія є власний план відключень на поточний тиждень.

Алгоритм отримання списку населених пунктів регіону міста Дніпра представлено на рис 4.1

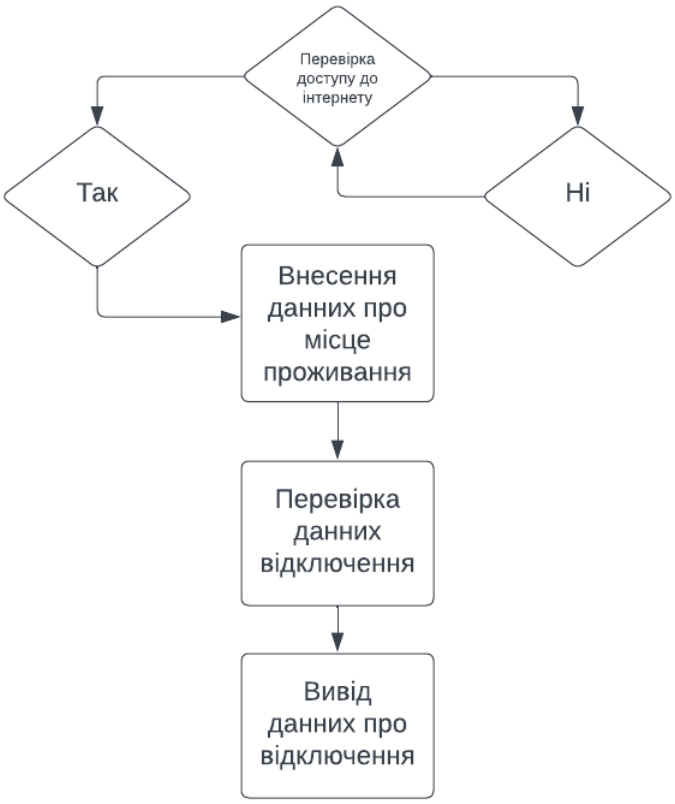

Рисунок 4.1 Блок схема алгоритму отримання данних

Показано роботу з віддаленими даними за допомогою HTTP-запиту. Для взаємодії з API використовується бібліотека dio і (Код описує) реалізовано абстрактний клас CityRemoteDataSource, який має метод getAllCities. Це дозволяє отримати список міст за певним регіоном.

Приклад 4.1 – Фрагмент тексту програми отримання списку міст

```
import 'package:dio/dio.dart';
import 'package:electricity_schedule/core/error/exception.dart';
import
'package:electricity_schedule/feature/data/models/city_model.dar
t';
import 'package:logger/logger.dart';

var logger = Logger();

abstract class CityRemoteDataSource {
  Future<List<CityModel>> getAllCities(String region);
}

class CityRemoteDataSourceImpl implements CityRemoteDataSource {
  final dio = Dio();

  @override
  Future<List<CityModel>> getAllCities(
      String region) async {
    try {
      Response response = await
dio.get("https://yasno.com.ua/api/v1/electricity-outages-
schedule/cities?region=$region");

      if (response.statusCode == 200) {
        List<CityModel> list =
CityListResponse.fromJsonArray(response.data).results;

        return list;
      } else {
        logger.log(Level.error, "Server error:
${response.statusCode}");
        throw ServerException();
      }
    } catch (error, stacktrace) {
      logger.log(Level.error, "Exception", error, stacktrace);
      throw ServerException();
    }
  }
}
```

Основний функціонал цього коду наступний:

1) Оголошення абстрактного класу CityRemoteDataSource: Цей клас визначає контракт для взаємодії з віддаленими даними, включаючи метод getAllCities.

2) Клас CityRemoteDataSourceImpl: Цей клас реалізує інтерфейс CityRemoteDataSource. Він містить екземпляр класу Dio, який використовується для здійснення HTTP-запитів.

3) Метод getAllCities: Цей метод приймає параметр region і повертає список об'єктів CityModel. Він виконує HTTP-запит до вказаного URL за допомогою dio.get, передаючи регіон у параметрі запиту.

4) Обробка результату запиту: Після отримання відповіді, перевіряється статус коду. Якщо код 200, тоді з використанням моделі CityListResponse та методу fromJsonArray, отримані дані перетворюються в список об'єктів CityModel, який повертається як результат методу.

5) Обробка помилок: У випадку, якщо статус код не дорівнює 200, відбувається логування помилки за допомогою об'єкта logger та виклик винятку ServerException. Також, якщо виникає будь-яка помилка під час виконання запиту, викликається виняток ServerException.

Цей код демонструє взаємодію з віддаленими даними за допомогою HTTP-запитів з використанням бібліотеки dio. Він отримує список міст за певним регіоном з віддаленого API та повертає цей список у вигляді об'єктів CityModel. При виникненні помилок під час запиту або невдалій відповіді сервера, викликається виняток ServerException.

Цей фрагмент коду є частиною функціоналу програми, яка реалізовує взаємодію з сервером для отримання даних про міста. Він використаний в інших частинах програми, де необхідно отримувати список міст для подальшої обробки та відображення.

Приклад відповіді серверу на запит за списком міст продемонстровано в прикладі 4.2.

Приклад 4.2 – Відповіді серверу в форматі JSON

[

```
  {
    "id": 1,
    "name": "м. Апостолове"
  },
  {
    "id": 2,
    "name": "м. Верхівцеве"
  },
  {
    "id": 3,
    "name": "м. Верхньодніпровськ"
  },
  {
    "id": 4,
    "name": "м. Вільногірськ"
  },
  {
    "id": 5,
    "name": "м. Дніпро"
  },
  {
    "id": 6,
    "name": "м. Жовті Води"
  },
  {
    "id": 7,
    "name": "м. Зеленодольськ"
  },
  {
    "id": 8,
    "name": "м. Кам’янське"
  }
]
```


Даний JSON містить масив об'єктів, кожен з яких представляє місто зі своїм унікальним ідентифікатором (id) та назвою (name).

Кожен об'єкт у масиві містить два поля: "id" – числовий ідентифікатор міста і "name" – назва міста. Загальна структура вказує на список міст, де кожен елемент міститься у вигляді окремого об'єкта з унікальним ідентифікатором та назвою.

4.4 Архітектура мобільного застосунку

Архітектура Flutter TDD Clean (рисунок 4.1) є підхідом до розробки додатків у Flutter, який поєднує кілька практик і принципів, щоб забезпечити

модульність, тестируваність та легкість супроводуваності коду. Основні аспекти цієї архітектури включають:

1. Чиста архітектура (Clean Architecture): Використовує принципи "Зовнішній шар", "Внутрішній шар" та "Центральний шар", щоб розділити код на незалежні компоненти з різним рівнем абстракції. Це дозволяє змінювати реалізацію окремих компонентів без впливу на інші частини системи.

2. TDD (Test-Driven Development): Процес розробки, в якому спочатку створюються тести, а потім реалізується код, який пройшов тести. Це допомагає забезпечити стабільність та надійність програмного забезпечення, оскільки тести перевіряють правильність роботи коду на ранній стадії розробки.

3. Використання патернів проектування: В архітектурі використовуються патерни, такі як Dependency Injection (DI), SOLID (принципи проектування об'єктно-орієнтованого програмування), репозиторії та інші, для забезпечення слабкої залежності, замінюваності компонентів та більшої розширюваності.

4. Розподіл логіки за допомогою шарів: Код розділяється на кілька шарів, таких як презентаційний (UI), доменний (бізнес-логіка) та джерело даних (зовнішні сервіси або бази даних). Це дозволяє легко міняти або тестувати окремі частини системи, не впливаючи на інші компоненти.

5. Модульність і повторне використання коду: Чиста архітектура та розподіл логіки допомагають створювати модульний код, що забезпечує повторне використання компонентів і полегшує розширення функціональності.

Використання Flutter TDD Clean Architecture допомагає створювати додатки з гнучким, тестовим та добре структурованим кодом, що сприяє простоті супроводуваності та масштабованості проектів.

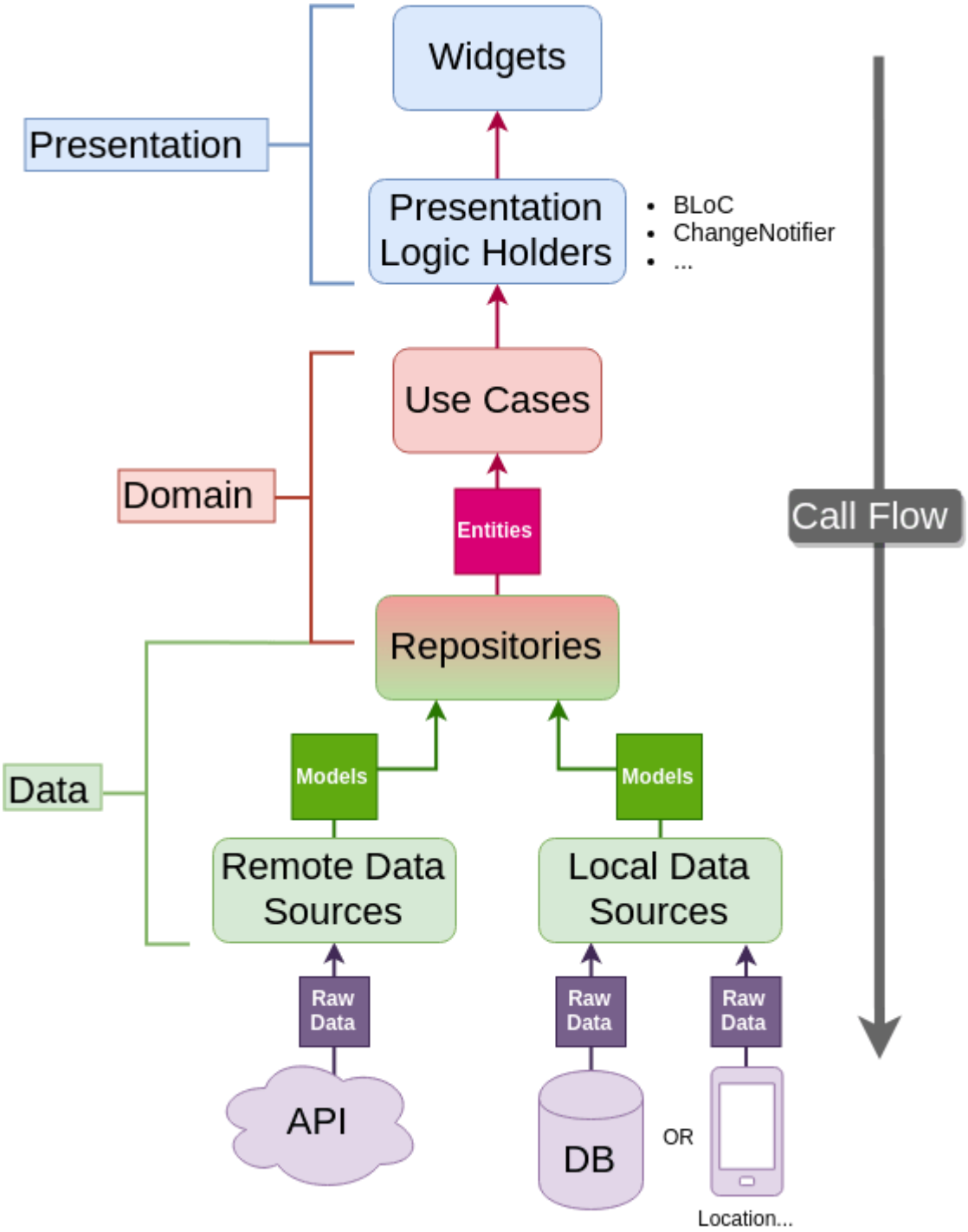

Рисунок 4.2 – Flutter TDD Clean Architecture

4.5 Опис середовища розробки

Android Studio є інтегрованою середовищем розробки (IDE), спеціально розробленою для розробки мобільних додатків під платформу Android. Це потужний інструмент, який надає розробникам широкі можливості для створення, тестування, налагодження та розгортання додатків для Android. Нижче наведено детальний опис основних можливостей Android Studio:

1. Редактор коду: Android Studio має потужний редактор коду, який підтримує автодоповнення, підсвічування синтаксису, виправлення помилок та інші функції, що полегшують написання коду. Редактор також інтегрований з системою контролю версій, що дозволяє зручно працювати з Git.

2. Менеджер проектів: Android Studio має вбудований менеджер проектів, який дозволяє створювати, відкривати та організовувати проекти. Він надає зручний інтерфейс для керування файлами та компонентами проекту.

3. Дизайнер інтерфейсу користувача: Середовище має вбудований візуальний редактор для створення інтерфейсу користувача. За допомогою графічного редактора можна швидко створювати макети, розташовувати елементи інтерфейсу та візуально налаштовувати їх властивості.

4. Емулятори та фізичні пристрої: Android Studio надає можливість запускати та тестувати додатки на емуляторах Android або на підключених фізичних пристроях. Це дозволяє перевіряти працездатність та відлагоджувати додатки на різних конфігураціях пристроїв.

5. Аналіз та профілювання додатків: Android Studio має вбудовані інструменти для аналізу та профілювання додатків. Це дозволяє виявляти й виправляти проблеми продуктивності, спостерігати за споживанням ресурсів та вдосконалювати роботу додатків.

6. Підтримка мови Kotlin: Android Studio повністю підтримує мову програмування Kotlin, яка є альтернативою мові Java для розробки додатків

для Android. Це дає розробникам можливість використовувати сучасну та експресивну мову для своїх проектів.

7. Розширення та плагіни: Android Studio підтримує систему плагінів, що дозволяє розширювати функціональність середовища розробки. Розробники можуть встановлювати різноманітні плагіни для покращення робочого процесу, додавання нових інструментів та розширення можливостей редагування коду.

Android Studio - це потужне інструментарій для розробки мобільних додатків під платформу Android. Він надає широкі можливості для зручної та продуктивної роботи, допомагаючи розробникам створювати якісні та ефективні додатки для мобільних пристроїв.

Плагін Flutter для Android Studio є корисним додатком, який дозволяє розробляти додатки на базі фреймворку Flutter прямо у середовищі розробки Android Studio. Цей плагін надає зручність та потужність розробки крос-платформних додатків, які працюють як на Android, так і на iOS.

Основні можливості плагіна Flutter для Android Studio:

1. Створення нових проектів: Плагін дозволяє швидко створювати нові проекти Flutter безпосередньо з Android Studio. Це дозволяє розробникам швидко налаштовувати проект та його структуру, а також визначати параметри і конфігурації для платформ Android та iOS.

2. Редактор коду: Плагін надає потужний редактор коду з підсвічуванням синтаксису, автодоповненням та виправленням помилок. Цей редактор спеціально налаштований для мови Dart, що дозволяє розробникам зручно та швидко писати код для додатків на Flutter.

3. Візуальний редактор інтерфейсу користувача: Плагін включає в себе візуальний редактор для побудови інтерфейсу користувача Flutter. За допомогою цього редактора можна візуально створювати та налаштовувати елементи інтерфейсу, розташовувати їх та змінювати їх властивості безпосередньо у Android Studio.

4. Запуск та налагодження додатків: Плагін дозволяє запускати та налагоджувати додатки Flutter безпосередньо у середовищі Android Studio. Розробники можуть перевіряти та відлагоджувати свої додатки на емуляторах Android або фізичних пристроях, що значно полегшує процес розробки та тестування.

5. Інтеграція з іншими і нструментами: Плагін Flutter добре інтегрується з іншими інструментами Android Studio

4.6 Інструкція користувача

Не залежно від операційної системи, додаток буде мати однаковий вигляд, для прикладу на рисунках 4.2 зображено головне вікно програми в Windows, Web та Android.

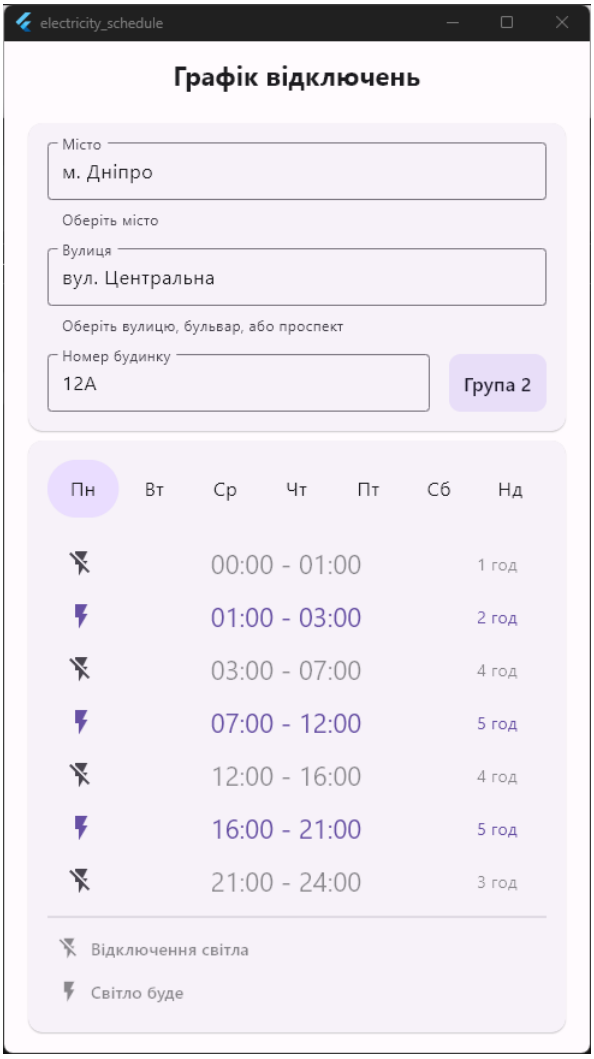

Рисунок 4.3 – Головне вікно програми в Windows

Застосунок також може мати темну тему інтерфейсу, приклад зображений на рисунку 4.3.

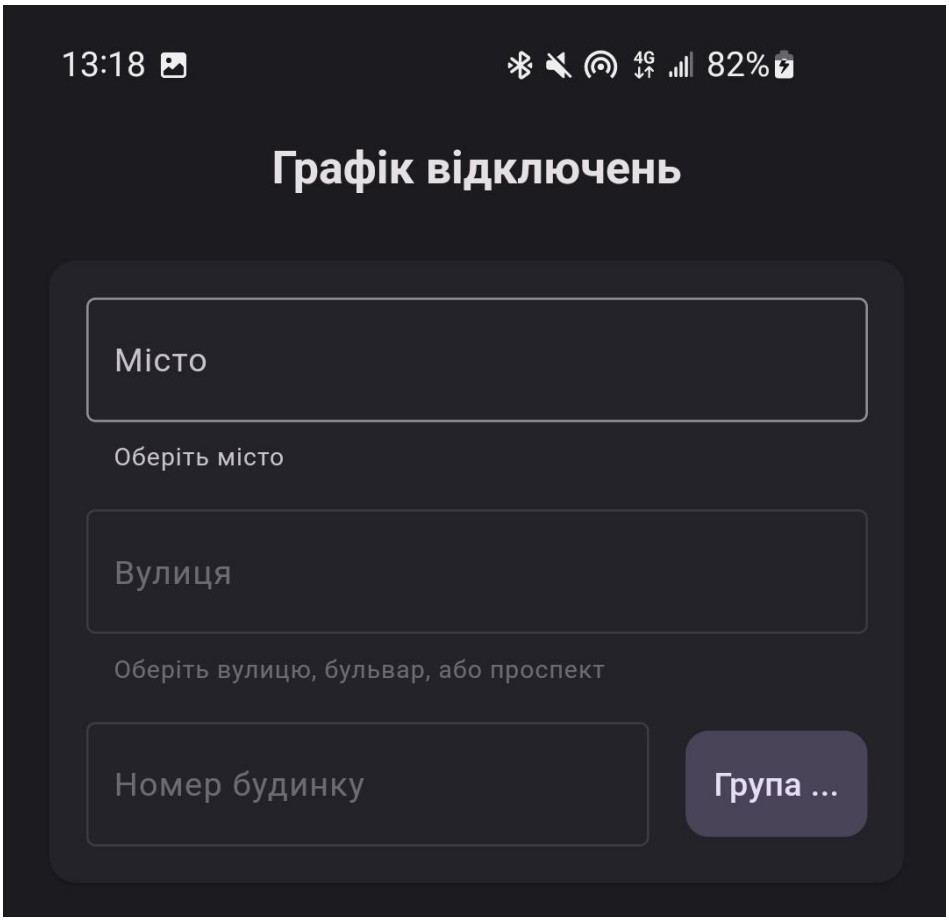

Рисунок 4.4 – Темна тема застосунку

4.7 Пошук графіка за адресою

Користувачу спочатку пропонується вибрати населений пункт, населені пункти підвантажуються з інтеренту, приклад вибору міста зображено на рисунку 4.4.

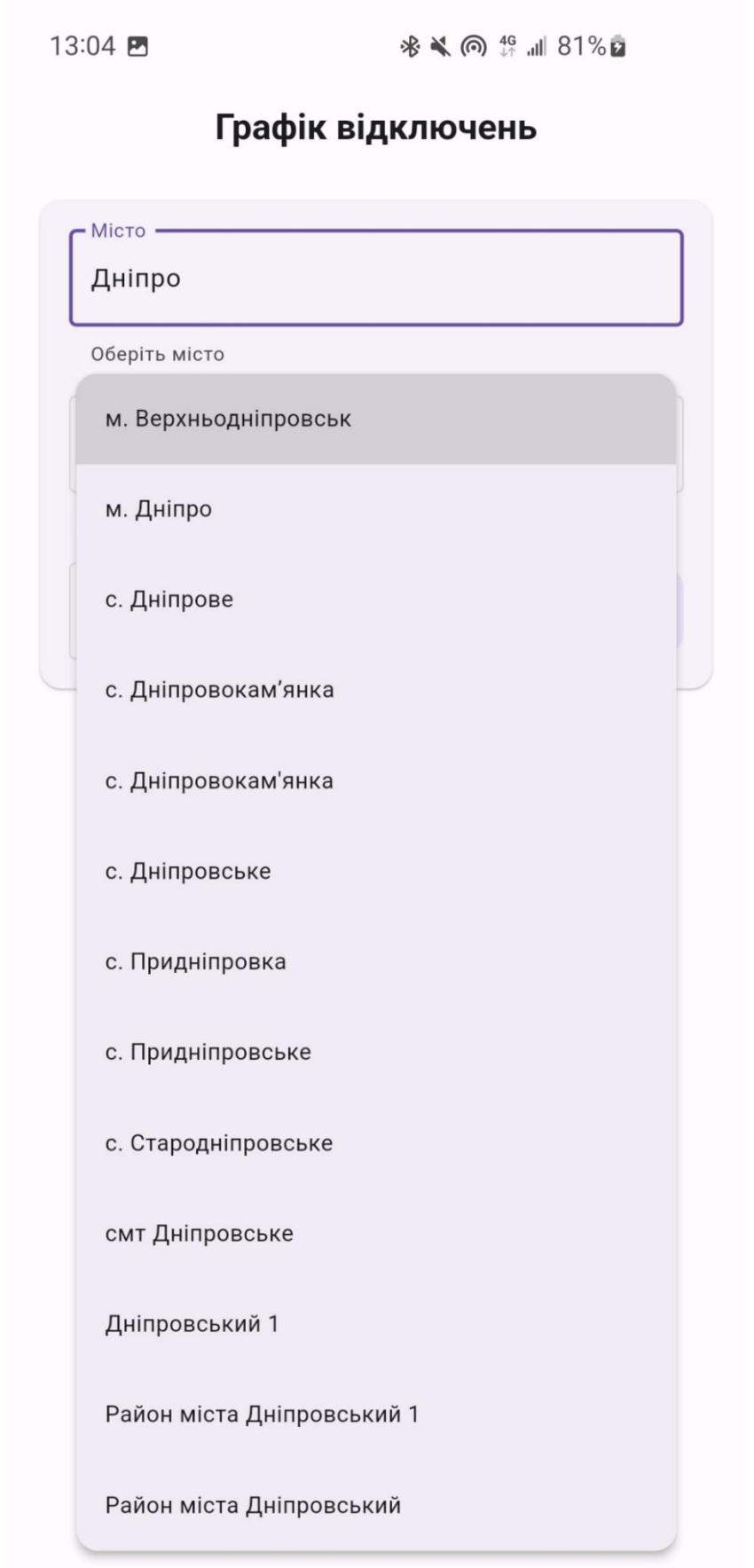

Рисунок 4.5 – Вибіа застосунку

Після вибору міста зі списка, автоматично завантажується список всіх вулиць цього міста, аналогічно після вибору вулиці завантажуються номери будинків, після вибору номеру будинку буде відображено групу цієї адреси та графік відключень на тиждень, приклад графіку зображено на рисунку 4.5.

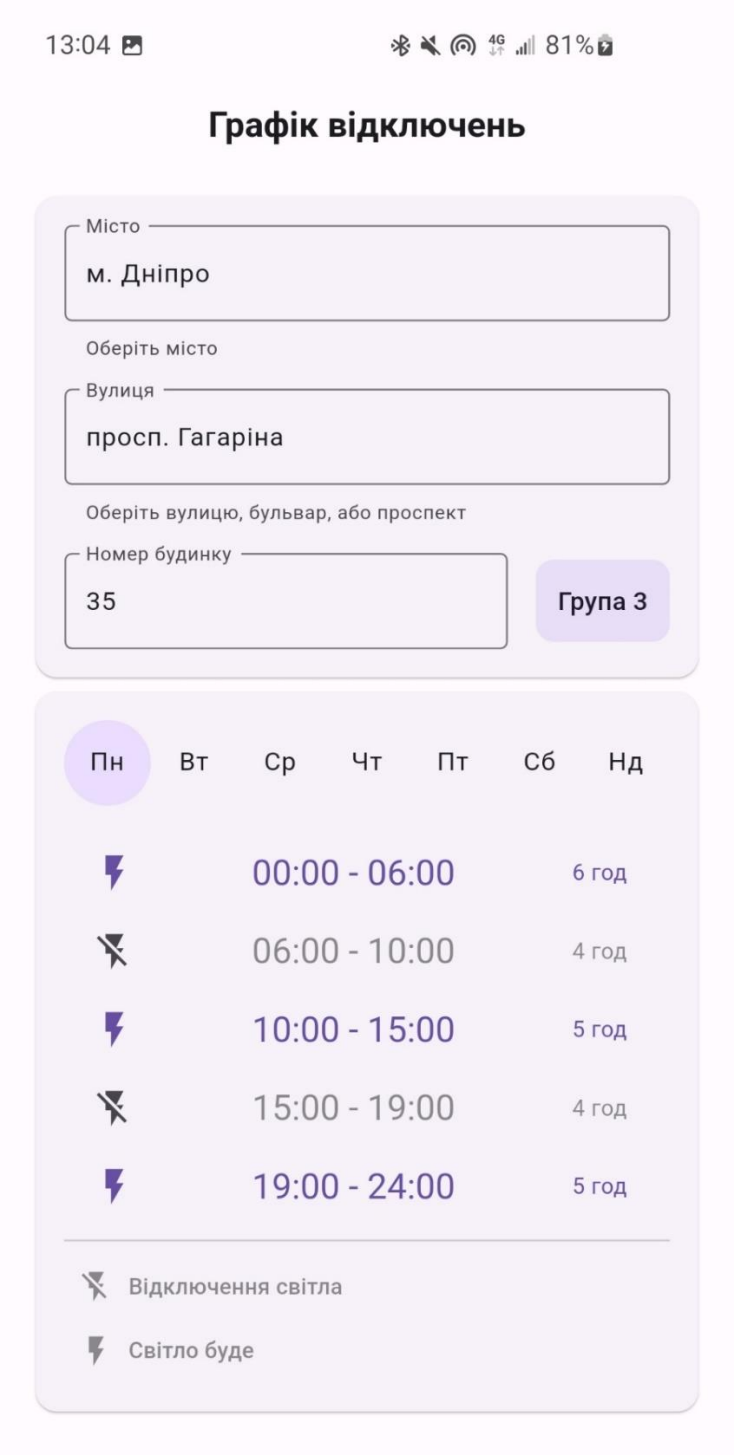

Рисунок 4.6 – Графік відключень

4.8 Обробка помилок

Якщо користувач неправильно ввів адресу, в одному із полів для введення, йому буде відображено повідомлення, прикла неправильно введеного міста зобржаено на рисунку 4.6.

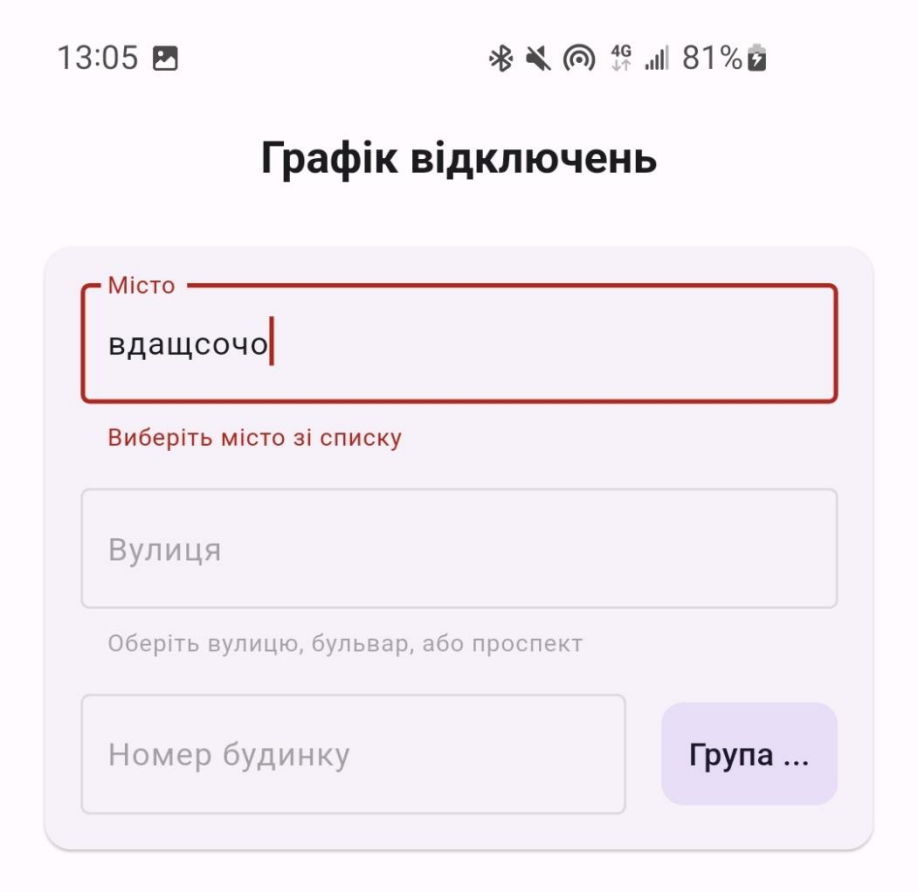

Рисунок 4.7 – Неправильне місто

Якщо не буде зв’язку з сервером, буде відображена відповідна помилка, зображеня помилки сервера зображено на рисунку 4.7.

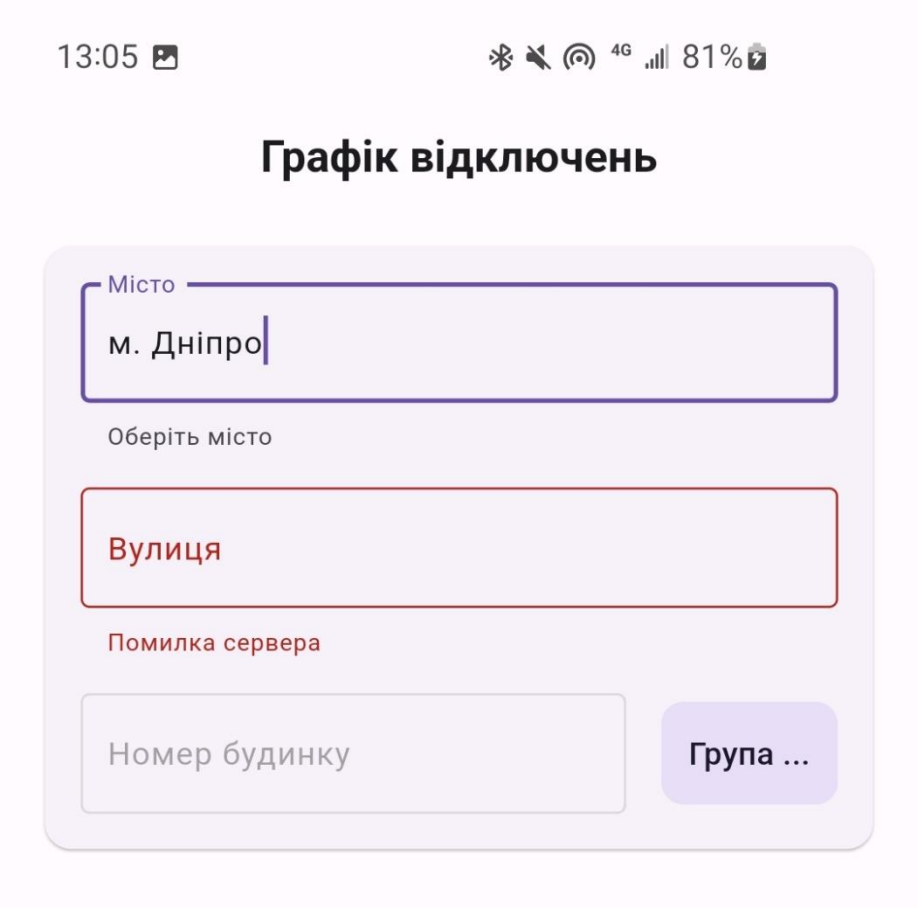

Рисунок 4.8 – Помилка з’єднання з сервером

ВИСНОВКИ

В ході виконання дипломного проекту, присвяченого комп'ютерній системі АТ "ДТЕК Дніпровські електромережі", були проведені розрахунки та розробка схеми адресації для корпоративної мережі та пристроїв. Для цього були створені фізична та логічна схеми, включаючи в себе налаштування VLAN.

Для забезпечення гнучкості та безпеки мережі було виконано агрегацію каналів PAgP, а також налаштовано динамічний NAT. Сервер та маршрутизатори були налаштовані з використанням служб AAA.

Додатково до розробки корпоративної мережі, в цьому проекті було розроблено мобільний додаток з використанням бібліотеки Flutter, який дозволяє користувачам відключати світло.

Ця програма має широкий спектр застосування, зокрема в електроенергетиці, промисловості та міському плануванні. Вона може бути використана енергетичними компаніями та мешканцями міста для аналізу графіку відключення електроенергії.

Також, цей додаток може бути корисним для промислових підприємств, де необхідно планувати відключення для ремонтних робіт або переключення на альтернативні джерела енергії. Крім того, органи місцевого самоврядування та планування розвитку міст можуть використовувати цю програму для врахування впливу відключення електроенергії на різні райони та інфраструктуру.

Таким чином, розроблена комп'ютерна система та мобільний додаток для відключення світла мають широкі можливості застосування, що охоплюють різні сфери, і можуть бути ефективними інструментами для аналітики, планування та оптимізації процесів відключення електроенергії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.Трой Макміллан « Cisco Networking Essentials » 2011. – 458 с

2. Биячуев, Т.А. «Безопасность корпоративных сетей»/ Т. А. Биячуев – М.: 2014. – 481 с.

3. Эрік Уиндміл «Flutter in Action» 2019 368 с

4. Агрегування каналів - [Електронний ресурс] - Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/PAgP

5. Офісний пакет Microsoft Office - [Електронний ресурс] - Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Microsoft_Office

6. Официальное руководство Cisco по подготовке к сертификационному экзамену "CCNA ICND2 200-105: маршрутизация и коммутация"

ДОДАТОК А

Текст програми налаштування мобільного додатку

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ПО ВІДКЛЮЧЕНЮ СВІТЛА

Текст програми

804.02070743.23008-01 12 01

Листів

2023

АННОТАЦІЯ

Программа містить частину програмного коду для налаштування функціоналу програми, яка реалізує взаємодію з сервером для отримання даних про міста, вулиці та номера будинків.

Зміст

Файл «exception.dart»:

```
class ServerException implements Exception {
}
```

Файл «failure.dart»:

```
abstract class Failure{
}

class ServerFailure implements Failure {
}
```

Файл «usecase.dart»

```
import 'package:dartz/dartz.dart';
import 'package:electricity_schedule/core/error/failure.dart';

abstract class UseCase<Type, Params> {
  Future<Either<Failure, Type>> call(Params params);
}
```

Файл «city_remote_data_source.dart»

```
import 'package:dio/dio.dart';
import 'package:electricity_schedule/core/error/exception.dart';
import
'package:electricity_schedule/feature/data/models/city_model.dar
t';
import 'package:logger/logger.dart';

var logger = Logger();

abstract class CityRemoteDataSource {
  Future<List<CityModel>> getAllCities(String region);
}

class CityRemoteDataSourceImpl implements CityRemoteDataSource {
  final dio = Dio();

  @override
  Future<List<CityModel>> getAllCities(
      String region) async {
    try {
      Response response = await
dio.get("https://yasno.com.ua/api/v1/electricity-outages-
schedule/cities?region=$region");

      if (response.statusCode == 200) {
        List<CityModel> list =
CityListResponse.fromJsonArray(response.data).results;

        return list;
      } else {
```

```
        logger.log(Level.error, "Server error:
${response.statusCode}");
        throw ServerException();
      }
    } catch (error, stacktrace) {
      logger.log(Level.error, "Exception", error, stacktrace);
      throw ServerException();
    }
  }
}
```

Файл «city_model.dart»

```
import
'package:electricity_schedule/feature/domain/entities/city_entit
y.dart';

class CityListResponse {
  final List<CityModel> results;

  CityListResponse(this.results);

  CityListResponse.fromJsonArray(List json)
      : results = json.map((i) =>
CityModel.fromJson(i)).toList();
}

class CityModel extends CityEntity {
  const CityModel({required id, required name}) : super(id: id,
name: name);

  factory CityModel.fromJson(Map<String, dynamic> json) {
    return CityModel(id: json["id"] as int, name: json["name"]);
  }

  Map<String, dynamic> toJson() {
    final Map<String, dynamic> data = <String, dynamic>{};
    data['id'] = id;
    data['name'] = name;
    return data;
  }
}
```

Файл «city_repository_impl.dart»

```
import 'package:dartz/dartz.dart';
import 'package:electricity_schedule/core/error/exception.dart';
import 'package:electricity_schedule/core/error/failure.dart';
import
'package:electricity_schedule/feature/data/datasources/city_remo
te_data_source.dart';
```

```
import
'package:electricity_schedule/feature/domain/entities/city_entit
y.dart';
import
'package:electricity_schedule/feature/domain/repositories/city_r
epository.dart';

class CityRepositoryImpl implements CityRepository {
  final CityRemoteDataSource remoteDataSource;

  CityRepositoryImpl({
    required this.remoteDataSource,
  });

  @override
  Future<Either<Failure, List<CityEntity>>> getAllCities(String
region) async {
    try {
      final cities = await
remoteDataSource.getAllCities(region);
      return Right(cities);
    } on ServerException {
      return Left(ServerFailure());
    }
  }
}
```

Файл «city_entity.dart»

```
class CityEntity {
  final int id;
  final String name;

  const CityEntity({required this.id, required this.name});
}
```

Файл «city_repository.dart»

```
import 'package:dartz/dartz.dart';
import 'package:electricity_schedule/core/error/failure.dart';
import
'package:electricity_schedule/feature/domain/entities/city_entit
y.dart';

abstract class CityRepository {
  Future<Either<Failure, List<CityEntity>>> getAllCities(String
region);
}
```

Файл «get_all_cities.dart»

```
import 'package:dartz/dartz.dart';
import 'package:electricity_schedule/core/error/failure.dart';
```

```
import
'package:electricity_schedule/feature/domain/entities/city_entit
y.dart';
import
'package:electricity_schedule/feature/domain/repositories/city_r
epository.dart';

class GetAllCities {
  final CityRepository cityRepository;

  GetAllCities(this.cityRepository);

  Future<Either<Failure, List<CityEntity>>> call(String region)
async {
    return await cityRepository.getAllCities(region);
  }
}
```

Файл «main_bloc.dart»

```
import 'dart:async';

import 'package:bloc/bloc.dart';
import 'package:electricity_schedule/core/error/failure.dart';
import
'package:electricity_schedule/feature/domain/entities/city_entit
y.dart';
import
'package:electricity_schedule/feature/domain/entities/house_enti
ty.dart';
import
'package:electricity_schedule/feature/domain/entities/schedule_e
ntity.dart';
import
'package:electricity_schedule/feature/domain/entities/street_ent
ity.dart';
import
'package:electricity_schedule/feature/domain/usecases/get_all_ci
ties.dart';
import
'package:electricity_schedule/feature/domain/usecases/get_all_ho
uses.dart';
import
'package:electricity_schedule/feature/domain/usecases/get_all_st
reets.dart';
import 'package:meta/meta.dart';

part 'main_event.dart';

part 'main_state.dart';
class MainBloc extends Bloc<MainEvent, MainState> {
  final GetAllCities getAllCities;
```

```
  final GetAllStreets getAllStreets;
  final GetAllHouses getAllHouses;

  List<CityEntity> cities = [];
  List<StreetEntity> streets = [];
  List<HouseEntity> houses = [];
  String? selectedRegion;
  CityEntity? selectedCity;
  StreetEntity? selectedStreet;
  HouseEntity? selectedHouse;

  MainBloc(
      {required this.getAllCities,
      required this.getAllStreets,
      required this.getAllHouses})
      : super(MainInitialState()) {
    on<CityLoadEvent>(_onCityLoadEvent);
    on<CitySelectedEvent>(_onCitySelectedEvent);
    on<CityChangedEvent>(_onCityChangedEvent);

    on<StreetLoadEvent>(_onStreetLoadEvent);
    on<StreetSelectedEvent>(_onStreetSelectedEvent);
    on<StreetChangedEvent>(_onStreetChangedEvent);

    on<HouseLoadEvent>(_onHouseLoadEvent);
    on<HouseSelectedEvent>(_onHouseSelectedEvent);
    on<HouseChangedEvent>(_onHouseChangedEvent);
  }

  FutureOr<void> _onCityLoadEvent(
      CityLoadEvent event, Emitter<MainState> emit) async {
    emit(CityLoadingState());
    selectedRegion = event.region;
    final failureOrCities = await getAllCities(event.region);
    emit(failureOrCities.fold(
        (failure) => CityErrorState(message:
_mapFailureToMessage(failure)),
        (cities) {
      this.cities = cities;
      return CityLoadedState(cities: cities);
    }));
  }

  FutureOr<void> _onStreetLoadEvent(
      StreetLoadEvent event, Emitter<MainState> emit) async {
    emit(StreetLoadingState());
    final failureOrStreets = await getAllStreets(event.region,
event.city);
    emit(failureOrStreets.fold(
        (failure) => StreetErrorState(message:
_mapFailureToMessage(failure)),
```

```
        (streets) {
      this.streets = streets;
      return StreetLoadedState(streets: streets, cities:
cities);
    }));
  }

  FutureOr<void> _onHouseLoadEvent(
      HouseLoadEvent event, Emitter<MainState> emit) async {
    emit(HouseLoadingState());
    final failureOrHouses =
        await getAllHouses(event.region, event.city,
event.street);
    emit(failureOrHouses.fold(
        (failure) => HouseErrorState(message:
_mapFailureToMessage(failure)),
        (houses) {
      this.houses = houses;
      return HouseLoadedState(houses: houses, cities: cities,
streets: streets);
    }));
  }

// ================================================
  FutureOr<void> _onCitySelectedEvent(
      CitySelectedEvent event, Emitter<MainState> emit) async {
    if (!cities.any((city) => city.name == event.cityName)) {
      emit(CityErrorState(message: "Виберіть місто зі списку"));
    } else {
      selectedCity = cities.firstWhere((city) => city.name ==
event.cityName);
      emit(CitySelectedState(city: selectedCity!, cities:
cities));
      add(StreetLoadEvent(region: selectedRegion!, city:
selectedCity!));
    }
  }

  FutureOr<void> _onStreetSelectedEvent(
      StreetSelectedEvent event, Emitter<MainState> emit) async
{
    if (!streets.any((street) => street.name ==
event.streetName)) {
      emit(StreetErrorState(message: "Виберіть вулицю зі спи-
ску"));
    } else {
      selectedStreet =
          streets.firstWhere((street) => street.name ==
event.streetName);
      emit(StreetSelectedState(street: selectedStreet!, cities:
cities, streets: streets));
```

```
      add(HouseLoadEvent(
          region: selectedRegion!,
          city: selectedCity!,
          street: selectedStreet!));
    }
  }

  FutureOr<void> _onHouseSelectedEvent(
      HouseSelectedEvent event, Emitter<MainState> emit) async {
    if (!houses.any((house) => house.name == event.houseName)) {
      emit(HouseErrorState(message: "Виберіть будинок зі спи-
ску"));
    } else {
      selectedHouse =
          houses.firstWhere((house) => house.name ==
event.houseName);
      emit(HouseSelectedState(
          house: selectedHouse!,
          schedule: schedules[selectedHouse!.group - 1], cities:
cities, streets: streets, houses: houses));
// add(HouseLoadEvent(
//     region: selectedRegion!,
//     city: selectedCity!,
//     street: selectedStreet!));
    }
  }

// ================================================
  FutureOr<void> _onCityChangedEvent(
      CityChangedEvent event, Emitter<MainState> emit) {
    if (state is MainErrorState) emit(CityLoadedState(cities:
cities));
  }

  FutureOr<void> _onStreetChangedEvent(
      StreetChangedEvent event, Emitter<MainState> emit) {
    if (state is MainErrorState) {
      emit(StreetLoadedState(cities: cities, streets: streets));
    }
  }

  FutureOr<void> _onHouseChangedEvent(
      HouseChangedEvent event, Emitter<MainState> emit) {
    if (state is MainErrorState) {
      emit(HouseLoadedState(cities: cities, streets: streets,
houses: houses));
    }
  }

  String _mapFailureToMessage(Failure failure) {
    switch (failure.runtimeType) {
```

```
      case ServerFailure:
        return 'Помилка сервера';
      default:
        return 'Невідома помилка';
    }
  }
}
```

Файл «main_event.dart»

```
part of 'main_bloc.dart';

@immutable
abstract class MainEvent {}

class CityLoadEvent extends MainEvent {
  final String region;

  CityLoadEvent({required this.region});
}

class CityChangedEvent extends MainEvent {
  final String text;

  CityChangedEvent({required this.text});
}

class CitySelectedEvent extends MainEvent {
  final String cityName;

  CitySelectedEvent({required this.cityName});
}

// ===============================================
class StreetLoadEvent extends MainEvent {
  final String region;
  final CityEntity city;

  StreetLoadEvent({required this.region, required this.city});
}

class StreetChangedEvent extends MainEvent {
  final String text;

  StreetChangedEvent({required this.text});
}

class StreetSelectedEvent extends MainEvent {
  final String streetName;

  StreetSelectedEvent({required this.streetName});
}
```

```
// ============================================
class HouseLoadEvent extends MainEvent {
  final String region;
  final CityEntity city;
  final StreetEntity street;

  HouseLoadEvent(
      {required this.region, required this.city, required
this.street});
}

class HouseChangedEvent extends MainEvent {
  final String text;

  HouseChangedEvent({required this.text});
}

class HouseSelectedEvent extends MainEvent {
  final String houseName;

  HouseSelectedEvent({required this.houseName});
}
```

Файл «main_state.dart»

```
part of 'main_bloc.dart';

@immutable
abstract class MainState {}

class MainInitialState extends MainState {}

class CityLoadingState extends MainState {}

class CityLoadedState extends MainState {
  final List<CityEntity> cities;

  CityLoadedState({required this.cities});
}

class CitySelectedState extends CityLoadedState {
  final CityEntity city;

  CitySelectedState({required this.city, required
super.cities});
}

// ============================================
class StreetLoadingState extends MainState {}

class StreetLoadedState extends CityLoadedState {
```

```
  final List<StreetEntity> streets;

  StreetLoadedState({required super.cities, required
this.streets});
}

class StreetSelectedState extends StreetLoadedState {
  final StreetEntity street;

  StreetSelectedState({required this.street, required
super.cities, required super.streets});
}

// ============================================
class HouseLoadingState extends MainState {}

class HouseLoadedState extends StreetLoadedState {
  final List<HouseEntity> houses;

  HouseLoadedState(
      {required super.cities, required super.streets, required
this.houses});
}

class HouseSelectedState extends HouseLoadedState {
  final HouseEntity house;
  final ScheduleEntity schedule;

  HouseSelectedState(
      {required this.schedule,
      required this.house,
      required super.cities, required super.streets, required
super.houses});
}

// ============================================
class MainErrorState extends MainState {
  final String message;

  MainErrorState({required this.message});
}

class CityErrorState extends MainErrorState {
  CityErrorState({required super.message});
}

class StreetErrorState extends MainErrorState {
  StreetErrorState({required super.message});
}

class HouseErrorState extends MainErrorState {
```

```
  HouseErrorState({required super.message});
}
```

Файл «main_screen.dart»

```
import 'package:autoscale_tabbarview/autoscale_tabbarview.dart';
import
'package:electricity_schedule/feature/domain/entities/city_entit
y.dart';
import
'package:electricity_schedule/feature/presentation/bloc/main_blo
c.dart';
import
'package:electricity_schedule/feature/presentation/widgets/my_te
xt_field.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';

class MainScreen extends StatefulWidget {
  const MainScreen({Key? key}) : super(key: key);

  @override
  State<MainScreen> createState() => _MainScreenState();
}

class _MainScreenState extends State<MainScreen> {
  String? selectedValue;

  @override
  Widget build(BuildContext context) {
    DateTime date = DateTime.now();

    return BlocBuilder<MainBloc, MainState>(
      builder: (context, mainState) {
        return SingleChildScrollView(
          child: Padding(
            padding: const EdgeInsets.fromLTRB(16, 8, 16, 16),
            child: Column(
              children: <Widget>[
                Card(
                  child: Padding(
                    padding: const EdgeInsets.all(16),
                    child: Column(
                      children: [
                        MyTextField(
                          enabled: true,
                          loading: mainState is
CityLoadingState,
                          helperText: "Оберіть місто",
                          label: "Місто",
                          errorText: mainState is CityErrorState
                              ? mainState.message
```

```
                            : null,
                        kOptions: mainState is CityLoadedState
                            ? mainState.cities.map((e) =>
e.name).toList()
                            : [],
                        onSelected: (selected) async {
                          BlocProvider.of<MainBloc>(context)
                              .add(CitySelectedEvent(cityName:
selected));
                        },
                        onChanged: (text) {
                          BlocProvider.of<MainBloc>(context)
                              .add(CityChangedEvent(text:
text));
                        },
                      ),
                      const SizedBox(
                        height: 16,
                      ),
                      MyTextField(
                        clear: mainState is
StreetLoadingState,
                        loading: mainState is
StreetLoadingState,
                        enabled: mainState is
StreetSelectedState ||
                            mainState is StreetLoadedState ||
                            mainState is StreetErrorState ||
                            mainState is HouseErrorState ||
                            mainState is HouseLoadingState ||
                            mainState is StreetLoadingState,
                        errorText: mainState is
StreetErrorState
                            ? mainState.message
                            : null,
                        helperText: "Оберіть вулицю, бульвар,
або проспект",
                        label: "Вулиця",
                        kOptions: mainState is
StreetLoadedState
                            ? mainState.streets.map((e) =>
e.name).toList()
                            : [],
                        onSelected: (selected) async {
                          BlocProvider.of<MainBloc>(context)

.add(StreetSelectedEvent(streetName: selected));
                        },
                        onChanged: (text) {
                          BlocProvider.of<MainBloc>(context)
```

```
                                    .add(StreetChangedEvent(text:
text));
                              },
                            ),
                            const SizedBox(
                              height: 16,
                            ),
                            Row(
                              mainAxisAlignment:
MainAxisAlignment.spaceAround,
                              crossAxisAlignment:
CrossAxisAlignment.center,
                              children: [
                                Expanded(
                                  child: MyTextField(
                                    clear: mainState is
HouseLoadingState,
                                    loading: mainState is
HouseLoadingState,
                                    enabled: mainState is
StreetSelectedState ||
                                        mainState is
HouseLoadedState ||
                                        mainState is HouseErrorState
||
                                        mainState is
HouseLoadingState,
                                    errorText: mainState is
HouseErrorState
                                        ? mainState.message
                                        : null,
                                    helperText: mainState is
HouseErrorState
                                        ? "Оберіть номер будинку"
                                        : null,
                                    label: "Номер будинку",
                                    kOptions: mainState is
HouseLoadedState
                                        ? mainState.houses
                                            .map((e) => e.name)
                                            .toList()
                                        : [],
                                    onSelected: (selected) async {

FocusScope.of(context).unfocus();

BlocProvider.of<MainBloc>(context).add(

HouseSelectedEvent(houseName: selected));
                                    },
                                    onChanged: (text) {
```

```
BlocProvider.of<MainBloc>(context)

.add(HouseChangedEvent(text: text));
                                  },
                                ),
                              ),
                              Padding(
                                padding: const
EdgeInsets.only(left: 16),
                                child: Material(
                                  borderRadius:
BorderRadius.circular(10),
                                  color: Theme.of(context)
                                      .colorScheme
                                      .secondaryContainer,
                                  child: Padding(
                                    padding: const
EdgeInsets.all(12),
                                    child: Text(
                                      "Група ${mainState is
HouseSelectedState ? mainState.house.group : '...'}",
                                      textAlign: TextAlign.center,
                                      style: Theme.of(context)
                                          .textTheme
                                          .titleMedium
                                          ?.copyWith(
                                              color:
Theme.of(context)
                                                  .colorScheme

.onSecondaryContainer),
                                    ),
                                  ),
                                ),
                              )
                            ],
                          )
                        ],
                      ),
                    ),
                  ),
                  if (mainState is HouseSelectedState)
                    Card(
                        child: Padding(
                      padding: const EdgeInsets.all(16.0),
                      child: DefaultTabController(
                          length: 7,
                          child: Column(
                            children: [
                              TabBar(
```

```
                                indicator: BoxDecoration(
                                    borderRadius:
BorderRadius.circular(100),
                                    color: Theme.of(context)
                                        .colorScheme
                                        .primaryContainer),
                                splashBorderRadius:
BorderRadius.circular(50),
                                dividerColor: Colors.transparent,
                                labelPadding: EdgeInsets.zero,
                                padding: EdgeInsets.zero,
                                indicatorSize:
TabBarIndicatorSize.tab,
                                labelColor: Theme.of(context)
                                    .colorScheme
                                    .onPrimaryContainer,
                                tabs: [
                                  Tab(
                                    child: Text(
                                      "Пн",
                                      style:

Theme.of(context).textTheme.bodyLarge,
                                    ),
                                  ),
                                  Tab(
                                    child: Text(
                                      "Вт",
                                      style:

Theme.of(context).textTheme.bodyLarge,
                                    ),
                                  ),
                                  Tab(
                                    child: Text(
                                      "Ср",
                                      style:

Theme.of(context).textTheme.bodyLarge,
                                    ),
                                  ),
                                  Tab(
                                    child: Text(
                                      "Чт",
                                      style:

Theme.of(context).textTheme.bodyLarge,
                                    ),
                                  ),
                                  Tab(
                                    child: Text(
```

```
                                    "Пт",
                                    style:
Theme.of(context).textTheme.bodyLarge,
                                  ),
                                ),
                                Tab(
                                  child: Text(
                                    "Сб",
                                    style:
Theme.of(context).textTheme.bodyLarge,
                                  ),
                                ),
                                Tab(
                                  child: Text(
                                    "Нд",
                                    style:
Theme.of(context).textTheme.bodyLarge,
                                  ),
                                ),
                              ],
                            ),
                            SizedBox(
                              height: 16,
                            ),
                            AutoScaleTabBarView(
                                children:
mainState.schedule.scheduleDayList
                                    .map(
                                      (scheduleDay) => Column(
                                          crossAxisAlignment:
CrossAxisAlignment.center,
                                          children:
scheduleDay.scheduleTimeList
                                              .map(
                                                (scheduleTime)
=> ListTile(
                                                  visualDensity:
VisualDensity(
                                                      vertical:
-3),
                                                  leading:
scheduleTime.off
                                                      ? Icon(
Icons.flash_off,
                                                          size:
24,
```

```
                                                                    )
                                                                : Icon(
Icons.flash_on,
                                                                    color:
Theme.of(context)
.colorScheme
.primary,
                                                                    size:
24,
                                                                ),
                                                            title:
scheduleTime.off
                                                                ? Text(
"${scheduleTime.start < 10 ? '0${scheduleTime.start}' :
scheduleTime.start.toString()}:00 - ${scheduleTime.end < 10 ?
'0${scheduleTime.end}' : scheduleTime.end.toString()}:00",
textAlign:
TextAlign.center,
                                                                    style:
Theme.of(
context)
.textTheme
.titleLarge
?.copyWith(
color: Theme.of(
context)
.colorScheme
.onBackground
.withOpacity(
0.5)))
                                                                : Text(
"${scheduleTime.start < 10 ? '0${scheduleTime.start}' :
```

```
scheduleTime.start.toString()}:00 - ${scheduleTime.end < 10 ?
'0${scheduleTime.end}' : scheduleTime.end.toString()}:00",

textAlign:

TextAlign.center,
                                                                style:
Theme.of(

context)

.textTheme

.titleLarge

?.copyWith(

color: Theme.of(

context)

.colorScheme

.primary),
                                                                    ),
                                                            trailing:
Text(

"${scheduleTime.end - scheduleTime.start} год",
                                                              style:
Theme.of(context)

.textTheme

.bodyMedium

?.copyWith(

color: scheduleTime

.off

? Theme.of(

context)

.colorScheme

.onBackground

.withOpacity(
```

```
0.5)

: Theme.of(

context)

.colorScheme

.primary),
                                                ),
                                              ),
                                            )
                                            .toList()),
                                    )
                                    .toList()),
                            Divider(),
                            Row(
                              children: [
                                Padding(
                                  padding: const
EdgeInsets.symmetric(
                                      horizontal: 8.0, vertical:
8),
                                  child: Icon(
                                    Icons.flash_off,
                                    size: 20,
                                    color: Theme.of(context)
                                        .colorScheme
                                        .onBackground
                                        .withOpacity(0.5),
                                  ),
                                ),
                                Text(
                                  "Відключення світла",
                                  style: Theme.of(context)
                                      .textTheme
                                      .labelLarge
                                      ?.copyWith(
                                          color:
Theme.of(context)
                                              .colorScheme
                                              .onBackground

.withOpacity(0.5)),
                                ),
                              ],
                            ),
                            Row(
                              children: [
                                Padding(
```

```
                                    padding: const
EdgeInsets.symmetric(
                                        horizontal: 8.0, vertical:
8),
                                    child: Icon(
                                      Icons.flash_on,
                                      size: 20,
                                      color: Theme.of(context)
                                          .colorScheme
                                          .onBackground
                                          .withOpacity(0.5),
                                    ),
                                  ),
                                  Text(
                                    "Світло буде",
                                    style: Theme.of(context)
                                        .textTheme
                                        .labelLarge
                                        ?.copyWith(
                                            color:
Theme.of(context)
                                                .colorScheme
                                                .onBackground

.withOpacity(0.5)),
                                  )
                                ],
                              )
                            ],
                          )),
                    ))
                ],
              ),
            ),
          );
        },
      );
  }
}
```

файл «my_text_field.dart»

```
import 'package:flutter/material.dart';
import 'package:flutter/scheduler.dart';

class MyTextField extends StatefulWidget {
  final List<String> kOptions;
  final String label;
  final String? helperText;
  final String? errorText;
  final AutocompleteOnSelected<String> onSelected;
  final VoidCallback? onEditingComplete;
```

```
  final ValueChanged<String>? onChanged;
  final bool? enabled;
  final bool loading;
  final bool clear;

  const MyTextField({Key? key,
    required this.kOptions,
    required this.label,
    this.helperText,
    this.errorText,
    required this.onSelected,
    this.onEditingComplete,
    this.onChanged,
    this.enabled,
    this.loading = false, this.clear = false})
      : super(key: key);

  @override
  State<MyTextField> createState() => _MyTextFieldState();
}

class _MyTextFieldState extends State<MyTextField> {
  @override
  Widget build(BuildContext context) {
    return LayoutBuilder(
      builder: (context, constraints) {
        return Autocomplete<String>(
          fieldViewBuilder:
              (context, textEditingController, focusNode,
onFieldSubmitted) {
            if (!(widget.enabled ?? false) || widget.clear) {
              textEditingController.clear();
            }
            return TextField(
              enabled: widget.enabled,
              onEditingComplete: () {
                widget.onSelected(textEditingController.text);
              },
              onChanged: widget.onChanged,
              controller: textEditingController,
              focusNode: focusNode,
              decoration: InputDecoration(
                errorText: widget.errorText,
                helperText: widget.helperText,
                label: Text(widget.label),
                isDense: true,
                // contentPadding: EdgeInsets.all(8),
                border: const OutlineInputBorder(),
                suffixIconConstraints:
                const BoxConstraints(maxHeight: 48, maxWidth:
48),
```

```
                suffixIcon: widget.loading
                    ? const Padding(
                  padding: EdgeInsets.all(8.0),
                  child: CircularProgressIndicator(
                    strokeWidth: 2,
                  ),
                )
                    : null,
              ),
            );
          },
          optionsViewBuilder: (context, onAutoCompleteSelect,
options) {
            return Align(
                alignment: Alignment.topLeft,
                child: SizedBox(
                    width: constraints.maxWidth,
                    child: Card(
                      clipBehavior: Clip.antiAlias,
                      elevation: 4,
                      color: Theme
                          .of(context)
                          .colorScheme
                          .background,
                      child: ListView.builder(
                        padding: const EdgeInsets.all(0),
                        shrinkWrap: true,
                        itemCount: options.length,
                        itemBuilder: (BuildContext context, int
index) {
                          final String option =
options.elementAt(index);
                          return InkWell(
                            onTap: () {
                              onAutoCompleteSelect(option);
                            },
                            child: Builder(builder:
(BuildContext context) {
                              final bool highlight =

AutocompleteHighlightedOption.of(context) ==
                                      index;
                              if (highlight) {
                                SchedulerBinding.instance

.addPostFrameCallback((Duration timeStamp) {

Scrollable.ensureVisible(context,
                                      alignment: 0.5);
                                });
                              }
```

```
                                return Container(
                                  color: highlight
                                      ? Theme
                                      .of(context)
                                      .focusColor
                                      : null,
                                  padding: const
EdgeInsets.all(16.0),
                                  child: Text(option),
                                );
                              }),
                          );
                        },
                      ),
                    )));
          },
          optionsBuilder: (TextEditingValue textEditingValue) {
            if (textEditingValue.text == '') {
              return const Iterable<String>.empty();
            }
            return widget.kOptions.where((String option) {
              return option
                  .toLowerCase()

.contains(textEditingValue.text.toLowerCase());
            });
          },
          onSelected: widget.onSelected,
        );
      },
    );
  }
}
```

Файл «electitity_schedule_app.dart»

```
import
'package:electricity_schedule/feature/data/datasources/city_remo
te_data_source.dart';
import
'package:electricity_schedule/feature/data/datasources/house_rem
ote_data_source.dart';
import
'package:electricity_schedule/feature/data/datasources/street_re
mote_data_source.dart';
import
'package:electricity_schedule/feature/data/repositories/city_rep
ository_impl.dart';
import
'package:electricity_schedule/feature/data/repositories/house_re
pository_impl.dart';
```

```
import
'package:electricity_schedule/feature/data/repositories/street_r
epository_impl.dart';
import
'package:electricity_schedule/feature/domain/usecases/get_all_ci
ties.dart';
import
'package:electricity_schedule/feature/domain/usecases/get_all_ho
uses.dart';
import
'package:electricity_schedule/feature/domain/usecases/get_all_st
reets.dart';
import
'package:electricity_schedule/feature/presentation/bloc/main_blo
c.dart';
import
'package:electricity_schedule/feature/presentation/screens/main_
screen.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';

class ElectricityScheduleApp extends StatelessWidget {
  const ElectricityScheduleApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Графік відключень',
      theme: ThemeData.dark(useMaterial3: true),
      // themeMode: ThemeMode.dark,
      home: Scaffold(
        appBar: AppBar(
          centerTitle: true,
          title: const Text(
            'Графік відключень',
            style: TextStyle(fontWeight: FontWeight.bold),
          ),
        ),
        body: MultiBlocProvider(providers: [
          BlocProvider<MainBloc>(
            create: (context) => MainBloc(
                getAllCities: GetAllCities(CityRepositoryImpl(
                    remoteDataSource:
CityRemoteDataSourceImpl())),
                getAllStreets:
GetAllStreets(StreetRepositoryImpl(
                    remoteDataSource:
StreetRemoteDataSourceImpl())),
                getAllHouses: GetAllHouses(HouseRepositoryImpl(
```

```
                    remoteDataSource:
HouseRemoteDataSourceImpl()))
            )
              ..add(CityLoadEvent(region: 'dnipro')),
          )
        ], child: const MainScreen()),
      ),
    );
  }
}
```