НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут телекомунікаційних систем

Кафедра телекомунікацій

До захисту допущено:

Завідувач кафедри

_____ Сергій КРАВЧУК

«___»_____2025 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія та програмування інфокомунікацій»

спеціальності 172 «Телекомунікації та радіотехніка»

на тему: «Дослідження методів побудови гібридних систем зберігання даних з інтеграцією локальної серверної інфраструктури та хмарних сервісів»

Виконав:
студент IV курсу, групи ТЗ-12
Стельник Антон Ігорович _____

Керівник:
Старший викладач кафедри ТК НН ІТС, к.ф.-м.н.
Руренко Олександр Григорович _____

Рецензент:
Доцент кафедри ІТТ НН ІТС, к.т.н., доцент,
Правило Валерій Володимирович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут телекомунікаційних систем
Кафедра телекомунікацій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

Освітньо-професійна програма «Інженерія та програмування інфокомунікацій»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій КРАВЧУК
«___»_____2025 р.

ЗАВДАННЯ
на дипломну роботу студенту
Стельнику Антону Ігоровичу

1. Тема роботи «Дослідження методів побудови гібридних систем зберігання даних з інтеграцією локальної серверної інфраструктури та хмарних сервісів», керівник роботи Руренко Олександр Григорович, к.ф.-м.н., затверджені наказом по університету від «26» травня 2025 р. № 1755-с.

2. Термін подання студентом роботи 9 червня 2025 р.

3. Вихідні дані до роботи: Використання інструментів Docker для ізольованого розгортання сервісів (Nextcloud, MySQL, Redis, Nginx), а також WireGuard для організації захищеного VPN-з'єднання з локальним сервером. Реалізація HTTPS-з'єднання за допомогою self-signed SSL-сертифіката.

4. Зміст роботи: Дослідити типи систем зберігання даних, проаналізувати можливі шляхи створення системи зберігання даних, реалізувати систему зберігання даних з віддаленим доступом за допомогою інструментів, проаналізувати ефективність та протестувати.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

6. Дата видачі завдання 15.10.2024

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вивчення теоретичного матеріалу по темі роботи. Збір літератури	15.10.2024 – 15.11.2024	Виконано
2.	Формування першого розділу	16.11.2024 – 12.01.2025	Виконано
3.	Проектування моделі файлового серверу. Збір інформації по потрібних утилітах	13.01.2025 – 28.01.2025	Виконано
4.	Формування другого розділу	29.01.2025 – 15.02.2025	Виконано
5.	Побудова гібридної моделі системи зберігання даних	15.02.2025 – 31.03.2025	Виконано
6.	Робота над 3 розділом	01.04.2025 – 15.05.2025	Виконано
7.	Оформлення дипломної роботи	16.05.2025 – 09.06.2025	Виконано

Студент Антон СТЕЛЬНИК

Керівник Олександр РУРЕНКО

РЕФЕРАТ

Пояснювальна записка обсягом 69 сторінок. Робота включає 43 ілюстрації, 3 додатки та 23 джерела.

Мета роботи полягає в дослідженні та проєктуванні гібридної системи зберігання даних, що поєднує локальну серверну інфраструктуру з можливістю інтеграції хмарних сервісів, та забезпечення безперервного, захищеного та контрольованого доступу до особистих файлів із будь-якої точки світу. Такий підхід розглядається як альтернатива комерційним та централізованим рішенням, з акцентом на автономність, приватність та масштабованість.

У роботі детально описано методи, засоби, кроки та результати, що використовувалися при дослідженні методів побудови гібридних систем зберігання даних. Кінцевим результатом стало отримання покрокової методики побудови гібридної системи зберігання даних. Слідування даній методиці призведе до отримання сховища з віддаленим та захищеним доступом.

Розроблена методика, яка була отримана в результаті дослідження, може бути використана як персональне сховище даних для домашнього використання, або ж може використовуватися як файловий сервер на невеликих підприємствах для спільного віддаленого та захищеного доступу до робочих файлів.

Ключові слова: зберігання даних, сервер, VPN, WireGuard, Docker, Nextcloud, Nginx, LAN, WAN, самопідписаний сертифікат, контроль, шифрування, доступ, HTTPS.

ABSTRACT

Explanatory note of 69 pages. The work includes 43 illustrations, 3 appendices, and 23 sources.

The purpose of the work is to research and design a hybrid data storage system that combines local server infrastructure with the ability to integrate cloud services, and to provide continuous, secure and controlled access to personal files from anywhere in the world. This approach is considered as an alternative to commercial and centralized solutions, with an emphasis on autonomy, privacy and scalability.

The work provides a detailed description of methods, tools, steps and results used in the study of methods for building hybrid data storage systems. The final result was a step-by-step methodology for building a hybrid data storage system. Following this methodology will result in obtaining a storage with remote and secure access.

The developed methodology, which was obtained as a result of the research, can be used as a personal data storage for home use, or it can be used as a file server in small businesses for shared remote and secure access to work files.

Keywords: data storage, server, VPN, WireGuard, Docker, Nextcloud, Nginx, LAN, WAN, self-signed certificate, control, encryption, access, HTTPS.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

DAS	Direct Attached Storage
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IP	Internet Protocol
LAN	Local Area Network
NAS	Network Attached Storage
NAT	Network Address Translation
PAT	Port Address Translation
SAN	Storage Area Network
SSH	Secure Shell
SSL	Secure Sockets Layer
VPN	Virtual Private Network
WAN	Wide Area Network
ОС	Операційна система
ПК	Персональний комп’ютер

ВСТУП

Згідно стандарту ISO/IEC 2382:2015, дані це – “реінтерпретоване представлення інформації у формалізований спосіб, придатний для передачі, інтерпретації або обробки” [1].

Важко уявити собі сучасний світ без даних. Вони оточують нас на кожному кроці. Це можуть бути книжка, стаття, відео на YouTube, персональні фото або документи. Якщо дані представленні в електронному вигляді, відповідно вони вимагатимуть місце де їх зберігати. Відповідно, для того щоб забезпечити цілісність даних, і не покладатися виключно на одне місце зберігання, або якщо ми просто хочемо звільнити першочергове місце зберігання даних, так як це обмежений ресурс, ми починаємо шукати альтернативні рішення.

Наприклад, виникає необхідність звільнити місце в пам’яті на телефоні і ми хочемо перенести наші фото, відео або інші файли. У такому випадку, зазвичай ми згадуємо за два найпростіших методи. Перший з них це скористатися такими сервісами як: Google Drive, One Drive, ICloud, тощо. Другий, це за наявності вдома комп’ютера або ноутбука перемістити всі необхідні нам дані на внутрішню пам’ять нашого другого пристрою або навіть на з’ємний жорсткий диск, який ми можемо під’єднати до нашого комп’ютера або ноутбука. Наведені методи є доволі ефективними в домашньому користуванні, але вони мають значні недоліки.

Розглянемо недоліки першого методу. Перш за все, вищенаведені сервіси, за умови, що ми користуємося безкоштовною версією, мають значні обмеження в пам’яті, якою ми можемо користуватися. Зазвичай це від 5 до 15 гігабайт, що навіть для домашнього користування може бути недостатньо. По-друге, якщо ми хочемо розширити об’єм пам’яті який нам необхідно, доведеться оформлювати підписку на данні сервіси та платити певну суму щомісячно. По-третє, в даному методі, ми в будь-якому випадку будемо зберігати дані на невідомому нам сервері, що в свою чергу означає що ми не знаємо та не контролюємо процес обробки наших даних в подальшому. Також з цього випливає, що ми покладаємо відповідальність за контроль доступності наших даних на когось іншого.

Відповідно, якщо ми перестанемо платити щомісячну підписку, ми втратимо доступ до наших даних, або також малоймовірний, але цілком можливий варіант розвитку подій, що в тій компанії в якій ми купуємо сховище відбудеться якийсь збій і ми на певний час втратимо доступ до наших даних. Тобто, якщо підсумувати, відповідальність за основні положення захисту інформації конфіденційність, цілісність і доступність, ми покладаємо на когось іншого, а також ми цілком і повністю втрачаємо контроль над тим, які маніпуляції проводяться над нашими даними та як вони обробляються в подальшому.

Отже, давайте тоді розглянемо недоліки другого методу. Перш за все, це доступність. При перенесені наших даних на ноутбук, комп'ютер або навіть жорсткий диск, ми зможемо отримати доступ до наших даних виключно лише в тому випадку, якщо ми будемо знаходитися безпосередньо біля нашого пристрою, куди ми записали нашу інформацію. Така сама проблема виникає і при записі цієї інформації, тобто нам також необхідно знаходитися безпосередньо біля пристрою, що зберігає інформацію. Це основний недолік, але також тут присутнє підводне каміння. Наприклад, якщо ми спробуємо перенести інформацію "по кабелю" з мобільного пристрою на базі IOS на комп'ютер або ноутбук на якому не стоїть система macOS, це викличе певні труднощі та затрати в часі через несумісність систем. В результаті буде проведено багато часу за переглядом повчальних відеороликів за тим як це зробити без повідомлень про помилки несумісності. Отже, в даному методі ми маємо повний контроль над нашими даними, але не маємо доступу до них в будь-який момент, а також маємо певні труднощі у випадку несумісності систем під час самого процесу переносу.

Як результат, наші основні дві проблеми при спробі зарезервувати, перенести або загалом провести маніпуляції над нашими даними це те, що ми або не контролюємо процес їхньої обробки, або маємо проблему з доступом до них, або ж маємо проблематичний процес переносу даних. Постає питання: "А які ж інші можливі варіанти збереження своїх власних даних за мінімальну ціну, з мінімально прикладеними зусиллями та так щоб це було безпечно, дані були захищені, легкодоступні та їхня цілісність не порушувалася?". Локальна

серверна інфраструктура буде відповіддю. Після її розгортання всі ці питання будуть вирішені так само як і питання яке собі хоч раз задавав кожен: “А як мені не втратити дані при втраті телефона?”, або “Куди б перенести потрібні мені файли так, щоб звільнити пам’ять на телефоні?”. Окрім того вирішення даних питань це лише маленький аспект того, що може локальна серверна інфраструктура. На її основі можна буде розгорнути велику кількість необхідних хмарних сервісів для полегшення повсякденного життя людини в залежності від потреб. Наприклад на основі локальної серверної інфраструктури можна буде розгорнути власний VPN сервер, та виходити в світ під домашньою IP адресою, захистивши свої данні VPN тунелем. Загалом можливості домашнього сервера обмежуватимуться виключно фантазією власника та об’ємом пам’яті, що ми закладаємо під цей сервер.

РОЗДІЛ 1
ЗАГАЛЬНІ ПІДХОДИ ДО ПРОЕКТУВАННЯ СИСТЕМИ ЗБЕРІГАННЯ ДАНИХ

1.1. Класифікація систем зберігання даних.

Закон Мура стверджує, що кількість транзисторів на інтегральній схемі приблизно подвоюється кожні два роки [2]. Хоча це твердження спочатку стосувалося розвитку обчислювальної техніки, сьогодні можна провести аналогію і зі швидким зростанням обсягів цифрових даних. Кожна компанія, кожен користувач постійно створюють та накопичують інформацію, яку потрібно десь зберігати — і саме тому системи зберігання даних стають усе важливішими.

Оскільки обсяг і різноманіття даних постійно зростають, виникає потреба не лише у зберіганні, але й у правильному виборі способу цього зберігання. Існує кілька різних архітектур систем зберігання, кожна з яких має свої особливості та підходить для різних завдань. Щоб забезпечити ефективну роботу з даними, важливо розуміти відмінності між ними. Тож давайте розглянемо три основні типи сховищ, які найчастіше використовуються при проектування систем зберігання даних.

1.1.1. Direct Attached Storage (DAS)

Direct Attached Storage (DAS) або ж система зберігання даних з прямим підключенням — це тип цифрового сховища, який безпосередньо підключається до комп’ютера, або сервера, без використання мережі. Найяскравішим прикладом даного типу сховища, буде звичайний жорсткий диск, або USB-накопичувач [18].

DAS-системи мають високу продуктивність завдяки прямому підключенню до сервера або комп'ютера, а також прості в налаштувані та мають низьку вартість, що робить їх підходящим варіантом для індивідуальних користувачів або невеликих офісів, яким потрібне швидке локальне сховище. Однак, головні недоліки це обмежена масштабованість та відсутність

мережевого доступу, що унеможливлює спільне використання даних між кількома пристроями [19].

1.1.2. Network Attached Storage (NAS)

NAS (Network Attached Storage) або ж мережеве зберігання даних з прямим підключенням — це тип мережевого сховища даних, у вигляді зовнішнього пристрою або серверу, який фізично з’єднаний із серверним або користувацьким оточенням через локальну (LAN) або глобальну (WAN) мережу та призначений для централізованого зберігання файлів і надання до них мережевого доступу [18].

NAS-системи мають низку переваг, а саме: централізоване зберігання файлів, віддалений доступ через мережу, повний контроль над даними, можливість гнучкого розташування, енергоефективність та порівняно низька вартість. При розгортанні NAS-системи варто врахувати, що при некоректній конфігурації можуть виникнути значні ризики безпеки, що безперечно є недоліком такої системи. Також NAS-системи цілком залежні від електроенергії та мережі. Не зважаючи на це, при правильному налаштуванні та наявності фактора відносно постійної електроенергії та мережі, NAS-системи є чудовим рішенням, яке використовується для створення персональних хмарних сховищ, використовується малими підприємствами для резервного копіювання та спільної роботи з файлами, а також у великих підприємствах, як частина масштабніших рішень [19].

1.1.3. Storage Area Network (SAN)

SAN (Storage Area Network) або ж мережа зберігання даних — це мережева інфраструктура для зберігання даних, яка призначена для організації зберігання даних та забезпечення швидкого та надійного доступу. SAN реалізується як окрема підмережа, інтегрована до локальної або глобальної мережі, та з’єднує один або кілька серверів (SAN-серверів) із численними пристроями зберігання даних [18].

SAN-системи мають високу швидкість доступу до даних завдяки блочному

рівню обміну, можливість масштабування та паралельної роботи серверів із багатьма сховищами, централізоване управління зберіганням і висока відмовостійкість, що робить SAN ідеальним рішенням для дата-центрів та великих підприємств. Проте впровадження SAN є дорогим, а також вимагає спеціалізованого обладнання та висококваліфікованого персоналу, за рахунок складності адміністрування [19].

1.1.4. Поняття гібридної системи

Поняття гібридної системи в даній роботі вводиться як визначення системи яка поєднує в собі різні типи зберігання та доступу до інформації, а саме локальний та хмарний типи, де є можливість отримати доступ до файлів як з локальної мережі, так і з глобальної. Таким чином, система забезпечує властивості хмарного середовища (доступність, мобільність, багатоплатформність) при збереженні контролю, безпеки і незалежності, притаманних локальним системам зберігання.

1.2. Організація файлового сховища даних

1.2.1. Вибір типу сховища даних

Тип сховища даних варто обирати спираючись на фактори: ціни, доступності з реалізації, кількість користувачів а також необхідність захищати дані. Кожен з типів сховища, по своєму проявляє себе в цих аспектах, тож розглянемо який тип підійде для створення моделі сховища даних виходячи з наявних ресурсів [19].

DAS є чудовим рішенням для звичайного домашнього користувача або невеликого підприємств, адже використання, наприклад, жорсткого диску комп’ютера або флешки задля зберігання інформації не потребує значних затрат, або ж спеціальних навичок, але в нашому випадку такий варіант сховища не підійде, адже він має обмежену кількість ресурсів, тобто потенційної пам’яті для використання, його не можна масштабувати в разі потреби, він не є централізованим, ну і найголовніше, це те, що ми не маємо віддаленого доступу до такого виду сховку, тому розглянемо інші варіанти [17].

Якщо ж розглядати наприклад SAN систему то для її реалізації потрібно забагато ресурсів, які в умовах маленького підприємства, або ж звичайного користувача реалізувати таку систему доволі складно. Але головним фактором є те, що така система просто не потрібна звичайному користувачеві або маленькому підприємстві, де в системі будуть оброблятися в основному звичайні робочі файли або ж фотографії, а не гіганські масиви даних, які ще й потребують високої кваліфікації для адміністрування [17].

Отже, залишається NAS, який є чимось середнім між двома типами сховищ перерахованих вище. З його допомогою буде реалізовано централізоване зберігання файлів, повний контроль над даними які зберігаються, масштабованість а головне це доступність з мережі, а також простота в реалізації, яка не потребує значних ресурсів, та адмініструванні [17].

1.2.2. Організація зберігання даних

Реалізацію системи зберігання даних буде розгорнуто на домашньому ПК який матиме необхідний об’єм вільної пам’яті, де буде розгорнута віртуальна машина для локального файлового сервера з використанням технології контейнеризації, що забезпечить ізоляцію служб, а також в майбутньому спростить процеси масштабування та оновлення. Дані зберігатимуться у файловій системі, адаптованій для ефективної роботи з великими обсягами інформації. Файлова система буде реалізована за допомогою спеціального програмного забезпечення, та додаткових сервісів для його працездатності.

1.3. Основні положення при проектуванні файлового сховища даних.

В книзі «Архітектури та технології зберігання даних» автор Цзіу Шу наводить такі основіні положення при створенні інструментів та технологій зберігання даних, а саме: продуктивність, зручність використання та надійність. Окрім цих положень він також наводить економічну ефективність, безпеку та масштабованість [20].

Керуючись даною книгою, розглянемо суть цих положень а також наші можливості їх реалізувати на практиці.

1.3.1. Продуктивність

Цзіу Шу наводить, що основними показниками продуктивності є пропускна здатність та затримка. Пропускна здатність — це кількість операцій, які система зберігання може обробити за одиницю часу, а затримка — це час, необхідний для виконання однієї операції [20].

Обидва ці показники на пряму залежать від обраного заліза на якому буде розгорнута система, від програмного забезпечення яке використовується, а також від пропускної здатності каналу зв'язку, що надається провайдером.

При реалцізації файлового сховища для забезпечення продуктивності, варто звернути увагу на апаратну частину. При виборі комплектуючих для безпосередньо апаратної частини варто враховувати їхню продуктивність адже вона потім і буде впливати як і на пропусну здатність так і на затримку. Наприклад, кращим рішенням буде надати перевагу SSD (solid-state drive) накопичувачам, ніж звичайному HDD (hard disk drive). Все залежить від обсягів і типу даних які обробляються, тобто якщо це персональне файлове сховище, куди завантажується 2 – 3 гігабайта особистих фото, HDD диску буде цілком достатньо.

Варто зазначити, що методи забезпечення продуктивності цілком залежать від системи яка будується та від показників які для цього необхідні. Таким чином, в рамках реалізації персонального файлового сховища для забезпечення продуктивності програмним шляхом буде достатньо налаштувати кешування, а також правильно підібрати програмне забезпечення для обробки файлів.

1.3.2. Зручність використання

Зручність використання має покращувати взаємодію між прикладними програмами та самими системами зберігання [20].

Простіше кажучи, при використанні файлової системи має бути швидкий запис даних та ефективна їх обробка або читання, щоб користувач безпосередньо не відчував дискомфорту. Для користувача файлове сховище має бути інтуїтивним та легким при взаємодії з файлами та каталогами. Мати логічну,

зрозумілу структуру, а також прості механізми доступу. Дане положення повністю забезпечується за рахунок програмного забезпечення яке призначене для організації та використання сервісів файлообміну.

1.3.3. Надійність

Надійність у файловому сховищі має забезпечувати запобігання втраті даних і переривані роботи сервісу у разі системних збоїв, таких як відмова дисків, проблеми з мережею а також людський фактор [20].

Система має бути доступною, такою що до неї можна підключитися, файли можна прочитати або записати без збоїв. В рамках нашої роботи віддалений доступ буде реалізовано за допомогою VPN тунелю, що дозволить працювати з сервером незалежно від фізичного розташування користувача, а також дозволить збільшувати кількість користувачів за потреби, тобто проводити масштабування. Додатково, застосування механізму автоматичного перезапуску контейнерів дозволяє миттєво відновлювати сервіси після збою. Проблемним може виявитися забезпечення безперервного функціонування серверу, за відсутності електропостачання, так як в такому випадку він просто вимкнеться. Для того щоб не порушувати доступність інформації, варто застосувати джерело безперебійного живлення, або ж фізично розташувати сервер так, щоб підключення до електроживлення та мережі інтернет було стабільним, тобто такому місці де з цим проблем немає.

1.3.4. Економічна ефективність

Економічна ефективність при проектуванні файлового сховища означає вищу продуктивність за нищу ціну [20].

В рамках даної роботи ми будемо користуватися в основному наявними ресурсами, і тому це можна навести як максимальну економічну ефективність відносно отриманого результату. Так як першочергово на меті стоїть практично показати реалізацію файлового сховища в умовах мінімальних ресурсів як фактична альтернатива платним рішенням великих компаній, що і демонструє економічну ефективність підходу, яка буде детально обґрунтована впродовж

подальшої роботи.

1.3.5. Безпека

Безпека інформаційно-комунікаційних систем – це сукупність заходів, спрямованих на забезпечення конфіденційності, цілісності та доступності інформації. У контексті файлових систем це означає захист даних від несанкціонованого доступу, забезпечення їхньої цілісності та доступності для авторизованих користувачів [21].

В даній роботі завдяки реалізації VPN тунелю для віддаленого доступу, буде перекрито проблему інформаційної безпеки даних, бо це дозволить уникнути несанкціонованого перехоплення даних у відкритих мережах, за рахунок шифрування трафіку. А також реалізація передачі даних по протоколу HTTPS додатково дозволить підняти рівень захищеності.

1.3.6. Масштабованість

Неминуче, що система з часом зростатиме. Таке зростання не повинно спричиняти серйозних перебоїв у роботі сервісу, а також значної втрати продуктивності для користувачів [22].

Масштабованість в роботі буде досягнута, за рахунок того, що в обраному типі зберігання даних передбачено можливість збільшення обсягу сховища, а також кількості користувачів без додаткового обладнання, за потреби можна буде додати нові сервіси, також можна буде розширити фізичне сховище, а також буде реалізовано можливість балансування між локальним і хмарним середовищем що і робить нашу систему гібридною.

Висновки

1. В Розділі 1 було розглянуто види систем зберігання даних, а саме DAS, NAS, SAN. Було наведено їх переваги та недоліки а також доцільність використання. Також було надане визначення поняттю гібридні системи яке використовується в рамках даної роботи.

2. Було визначено, що в даній роботі найоптимальнішим рішенням буде проектування NAS системи, так як вона своїми параметрами абсолютно

задовольняє наші потреби, а саме наявність контролю над даними, масштабованість та доступність з LAN та WAN, та не потребує значних ресурсів та вкладень для реалізації. Також розглянуто аспект організації зберігання та обробки даних в такій системі.

3. Наведено основні положення для моделювання та реалізації NAS системи на практиці, тобто продуктивність, зручність використання, надійсність, положення економічної ефективності, інформаційної безпеки та масштабованості цієї системи.

РОЗДІЛ 2
МОДЕЛЬ ФАЙЛОВОГО СХОВИЩА З ВИКОРИСТАННЯМ КОНТЕЙНЕРНОЇ АРХІТЕКТУРИ ТА ОРГАНІЗАЦІЄЮ VPN З'ЄДНАННЯ

2.1. Проектування моделі файлового сховища

2.1.1. Віртуальне середовище та контейнери для файлообміну

З урахуванням того, що у нас нема можливості докупити серверне обладнання, будемо реалізовувати сервер за рахунок наявних ресурсів, а саме домашнього ПК та вільних 400 гігабайт пам'яті. Для цього будемо використовувати віртуальне середовище Virtual Box, на яке ми встановимо віртуальну машину з операційною системою Ubuntu server.

Розгортання необхідних додатків, відбуватиметься за рахунок інструментів Docker. Так як задача полягає в реалізації файлообміну, для цього буде використано набір клієнт-серверного програмного забезпечення з відкритим кодом Nextcloud. Для забезпечення та покращення роботи Nextcloud, також буде розгорнуто образи контейнерів Redis та MySQL. База даних в оперативній пам'яті Redis потрібна для реалізації кешування сесій та запитів до Nextcloud, а база даних з відкритим кодом MySQL нам потрібна для реалізації зберігання метаданих, так як Nextcloud не забезпечує цього.

2.1.2. Безперервний віддалений доступ з LAN та WAN

Для доступу з локальної мережі нам необхідно буде провести мережеві налаштування віртуальної машини, а для віддаленого доступу нам знадобиться VPN з'єднання, для реалізації якого ми будемо використовувати WireGuard та білу IP адресу провайдера.

Розглянемо схему підключення нашого серверу до мережі, яку буде реалізовано (Рис 2.1). Схему було зроблено за допомогою комплексне програмне забезпечення для моделювання мереж Cisco Packet Tracer [16].

Варто зазначити, що під словом сервер, мається на увазі віртуальна машина, з серверною операційною системою, але для локальної мережі він буде

відображатися саме як окремий пристрій, тому схематично, віртуальну машину буде зображено як сервер.

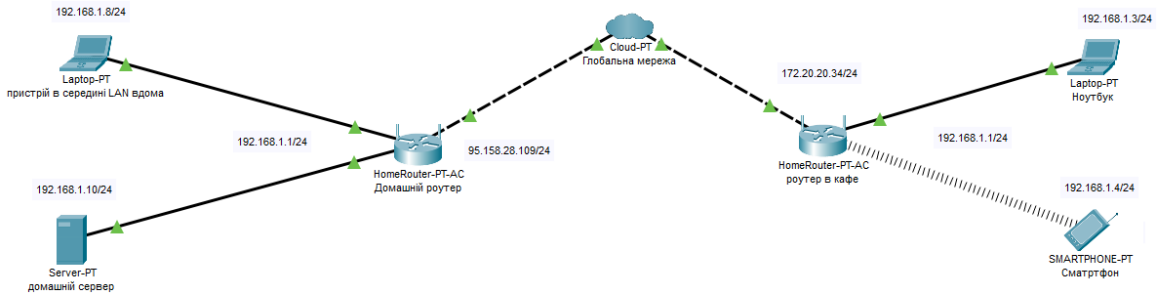

Рис. 2.1 Схема підключення всіх пристроїв моделі до мережі

Сервер в локальній мережі, підписаний як домашній сервер. Він підключений до домашнього маршрутизатора, який має білу IP адресу від провайдера (95.158.28.109) та з неї виходить в глобальну мережу. Також до локальної мережі підключені і інші пристрої, але для спрощення схеми, зображено лише один (пристрій в середині LAN вдома).

Другою частиною нашої моделі є користувачі, які хочуть підключитися до нашого сервера. В моделі ми будемо розглядати два пристрої в ролі користувачів, це смартфон та ноутбук, які під’єднані до глобальної мережі, наприклад, як зображено на моделі підключившись до WIFI роутеру в кафе. Варто зазначити, що підключення до глобальної мережі має відбуватися з будь-звідки, це може бути як і під'єднання з телефону до мобільної мережі оператора, так і мережа, наприклад, в університеті, головне, щоб в кінці кінців був вихід в глобальну мережу.

Тому наша задача забезпечити таке підключення, а саме, між клієнтами та сервером з різних мереж через глобальну мережу. Для цього нам і потрібен WireGuard, який створить VPN тунель між нашими пристроями. (Рис 2.2).

Важливою частиною налаштування є те, що ми можемо утворити два VPN тунелі, замість одного, щоб користувачі «не бачили» один одного, але так як нам не потрібна ізоляція користувачів один від одного, при реалізації моделі буде

залишено один тунель.

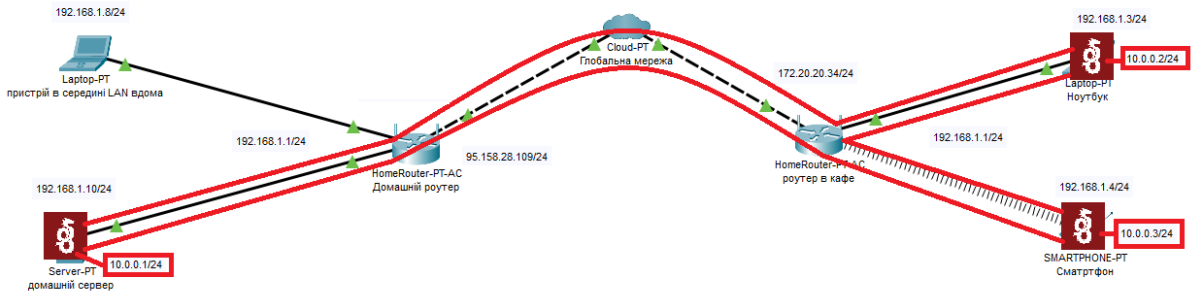

Рис. 2.2 Схема підключення клієнтів до сервера через VPN тунель WireGuard

На схемі IP адреса сервера в нашій віртуальній приватній мережі - це 10.0.0.1/24, а 10.0.0.2/24 та 10.0.0.3/24 IP адреси клієнтів. Трафік в побудованому тунелі буде шифруватися, що забезпечить безпеку підключення.

Далі за рахунок додаткових налаштувань сервера буде реалізовано переадресацію так, що клієнти зможе виходити в світ під білою IP адресою при цьому шифруючи трафік для мережі в якій він знаходиться. (Рис 2.3).

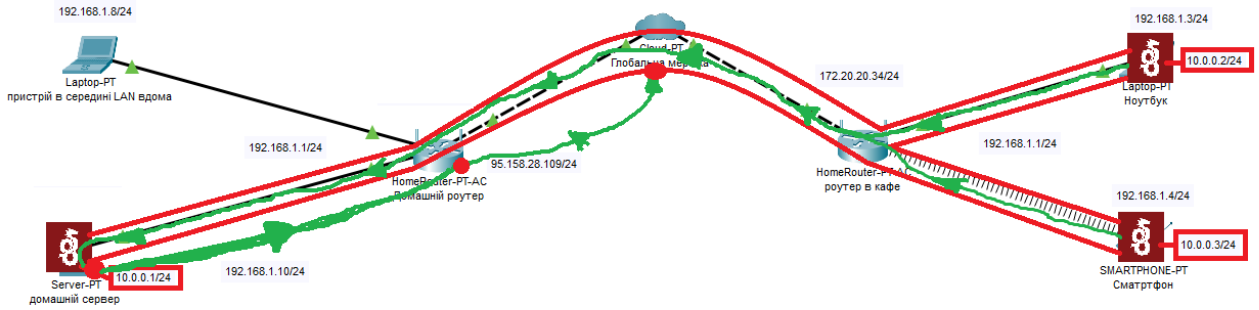

Рис. 2.3 Схема підключення клієнтів до глобальної мережі через сервер завдяки VPN тунелю WireGuard

Зеленими стрілочками показано напрямок руху трафіку до сервера, де сервер завдяки маскарадингу, тобто підміні IP адреси джерела, пропускає через себе весь трафік клієнтів і перенаправляє їх в глобальну мережу зі своєї білої IP,

(95.158.28.109), де клієнти виходять в глобальну мережу вже з неї, таким чином приховуючи, або «маскуючи» свої справжні IP адреса за за білою IP адресою серверу. Завдяки цьому в глобальній мережі буде підвищена безпека клієнтів шляхом приховування справжніх IP адрес, а файловий сервер додатково набуде функцій VPN серверу.

2.1.3. Безпека підключення та захист інформації

Окрім реалізації каналу зв’язку через VPN, наступним кроком з реазізації заходів для захисту інформації буде отримання SSL сертифікатів для того щоб підключення відбувалося за протоколом HTTPS.

Даний протокол шифрує трафік, таким чином унеможливлюючи перехоплення паролів або інформації яка передається. Також HTTPS гарантує автентичність сервера, запобігаючи атакам по типу man-in-the-middle. Використовуючи даний протокол ми знаємо, що дані не будуть пошкодженні або перехоплені.

HTTPS-з'єднання буде забезпечено за рахунок self-signed сертифікату та nginx. Враховуючи те, що даний проект призначений для особистого користування з обмеженим доступом через захищений VPN, ми ізолюємо його від зовнішньої мережі. Наприклад у альтернативному варіанті з Let’s Encrypt потрібен був би відкритий доступ до сервера, що відсутнє в нашому випадку, а отже ми одразу відкинули даний варіант.

Також потрібно враховувати, що в проекті відсутня потреба довіри з боку сторонніх центрів сертифікації, регулярне оновлення сертифікату не вимагається. Також такий підхід, всеодно реалізовує повноцінне шифрування, незважаючи на відсутність сертифікату від від офіційних центрів сертифікації**,** TLS з’єднання шифрується повністю, саме тому self-signed сертифікат буде найкращим рішенням.

З недоліків можна зазначити, що браузери та клієнти не будуть довіряти сертифікату, і при спробі під’єднатися, буде попередження "Небезпечне з’єднання" або йому подібні в залежності від браузера (Рис 2.4).

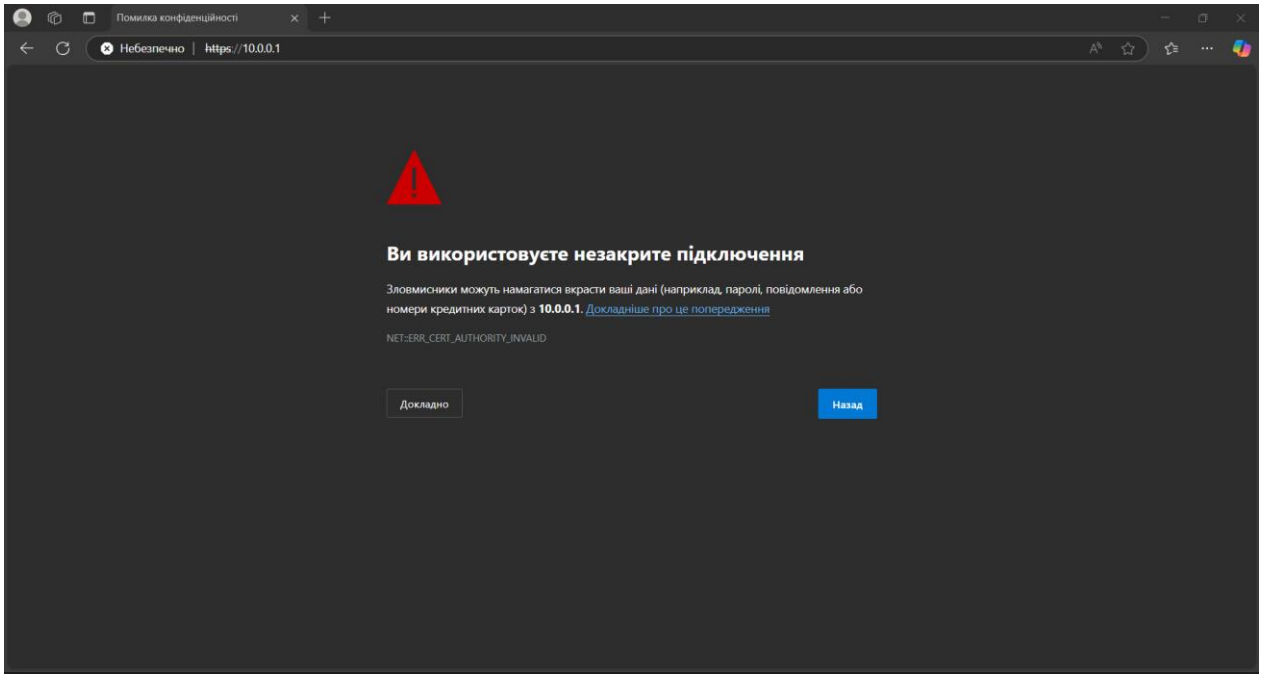

Рис. 2.4 Приклад попередження при вході на сервер з self-signed сертифікатом

Але оскільки ми знаємо, що це наш сервер, де ми підключаємося до нього за допомогою VPN, дане повідомлення буде ігноруватися, оскільки для того щоб підмінити кінцеву точку з’єднання, тобто сервер, необхідно отримати закритий ключ шифрування самого сервера. Цей ключ буде зберігається в закритому вигляді, в файлі з виключно root доступом. Отже для того щоб отримати цей ключ, необхідно вже мати доступ до сервера, в іншому випадку, підробити ключ не вдасться, а отже з’єднання не відбудеться , при цьому всьому https, з’єднання буде все одно забезпечене.

Nginx в даному випадку застосовується як зворотній проксі – він приймає запити на 443 порт, тобто HTTPS запити, розшифровує їх за допомогою self-signed SSL сертифікатів і далі передає запити всередину локальної мережі Docker у вигляді звичайного HTTP запиту до Nextcloud.

Таким чином відбувається шифрування між нашим клієнтом і веб сервером Nginx, а далі вже звичайний HTTP.

2.2. Утиліти для реалізації моделі файлового сховища

2.2.1. Гіпервізор

Для розгортання віртуальної машини буде використано програму віртуалізації **Oracle VM VirtualBox.** Це потужна віртуалізація з відкритим кодом для особистого та корпоративного використання. Як вказано на офіційному сайті VirtualBox — це повне програмне забезпечення для віртуалізації загального призначення для апаратного забезпечення x86_64 (з версією 7.1 додатково для macOS/Arm), призначене для ноутбуків, настільних комп'ютерів, серверів і вбудованих пристроїв [3].

Було обрано саме цей гіпервізор, за рахунок того, що він безкоштовний, дозволяє запускати декілька операційних систем відразу на одному фізичному комп'ютері та легкий в користуванні. В роботі буде використано версію 7.0.18.

2.2.2. Операційна система (OS)

Як операційну систему було обрано Ubuntu 24.04.2 LTS.

"Ubuntu — це повнофункціональний дистрибутив Linux для настільних комп'ютерів, ноутбуків, хмар і серверів із швидким і легким встановленням та регулярними випусками. Включено тісно інтегрований вибір чудових програм, а неймовірна різноманітність додаткового програмного забезпечення доступна лише за кілька клацань миші" [4]

У нашому випадку було обрано саме серверну версію. Ця система заточена під сервери, про що і йдеться на їх сайті. Було вирішено обрати саме цю операційну систему, так як вона не потребує багато ресурсів, що забезпечує високу продуктивність навіть на слабкому обладнанні, безкоштовна та з відкритим кодом, також ця система стабільна та отримуватиме оновлення безпеки та підтримку протягом 5 років, гнучка (за потреби можна встановити лише необхідні компоненти), наявні вбудовані інструменти для захисту. В Ubuntu Server відсутній графічний інтерфейс, вся взаємодія відбувається виключно через консоль. Інтуїтивність та зручність системи за рахунок цього зменшується, але таке рішення дозволяє полегшити та пришвидшити систему.

За рахунок вищенаведеного, дана система чудово підходить під наші задачі закриваючи всі потреби.

Системні характеристики даної операційної системи такі: процессор з тактовою частотою 1 гігагерц (GHz), або краще; 1 гігабайт (GB) оперативної пам’яті (RAM), 5 гігабайт (GB) вільного місця на диску [5].

2.2.3. Docker

Однією з важливих етапів роботи є контейнеризація додатків за допомогою інструментів Docker. Контейнеризація значно спрощує розгортання, управління та масштабування додатків, роблячи процес ільш ефективним та гнучким, що ідеально допомагає нам в реалізації нашої локальної серверної інфраструктури. Так як ми збираємося розгортати декілька додатків на нашому сервері, завдяки контейнеризації нам не потрібно буде розгортати декілька віртуальних машин в середині нього, замість цього ми просто розгорнемо декілька контейнерів.

Контейнери працюють однаково на будь-якій платформі, вони дозволяють легко масштабувати додаток, запускаючи більше екземплярів без потреби в розгортанні всього середовища заново. Також контейнери споживають менше ресурсів, на відміну від віртуальних машин, оскільки вони використовють ядро операційної системи без необхідності створення окремих ОС для кожного середовища. Окрім вище зазначеного, якщо додаток працює в контейнері, він не залежить від ОС або бібліотек хост-системи, а також працюють ізольовано один від одного, що підвищує безпеку та стабільність системи. Якщо один контейнер виходить з ладу, інші продовжують роботу. Оскільки кожен контейнер працює ізольовано, ризик впливу шкідливого ПЗ або конфлікту середовищ зменшується.

Варто зазначити, що на відміну від віртуальних машин контейнери запускаються швидше, бо не потребують завантаження повноцінної ОС [6].

Також варто згадати інструмент Docker, який буде використано в цій роботі, а саме Docker Compose. Docker Compose — це інструмент, який дозволяє визначати та запускати багатоконтейнерні Docker-застосунки за допомогою єдиного YAML-файлу. Замість того, щоб запускати кожен контейнер окремо,

Compose дає змогу описати всі сервіси, мережі та томи застосунку у файлі docker-compose.yml і керувати ними однією командою [23].

Отже, саме тому використання контейнерів, а також платформи Docker та її інструментів для контейнеризації, є необхідним в роботі, а саме в проектуванні та практичній реалізації моделі.

2.2.4. Nextcloud

Nextcloud - це набір клієнт-серверного програмного забезпечення, призначеного для організації та використання сервісів файлообміну. Це вільне та відкрите програмне забезпечення, що дозволяє будь-кому встановлювати й використовувати його на власних серверах. Самі розробники характеризують свою платформу як: “Ми розробляємо програмне забезпечення для децентралізованих і об’єднаних хмар як альтернатива централізованим хмарним сервісам” [7].

Дане програмне забезпечення було обрано для реалізації самого файлообміну на локальному сервері, за рахунок відкритого коду, кросплатформеності, а також по тій причині що це готове, безкоштовне рішення яке за потреби можна підлаштувати під себе та контролювати кожен аспект самого процесу файлообміну. Так як ми розгортаємо це рішення локально, то в результаті ми отримуємо повний контроль над даними. На відміну від комерційних хмарних сервісів (Google Drive, Dropbox), Nextcloud дозволяє зберігати файли на власному сервері, що підвищує рівень безпеки. Також у ПЗ наявна підтримка шифрування файлів, двофакторна автентифікація та контроль доступу, що захищає дані від несанкціонованого доступу. За рахунок наявності клієнтів для Windows, Linux, macOS, Android та iOS, що дозволяє легко синхронізувати файли між пристроями забезпечується синхронізація та доступ звідусіль. Окрім цього, дане ПЗ має зручний веб інтерфейс, що дозволяє працювати з файлами без необхідності встановлення додаткового ПЗ.

2.2.5. Redis

Redis – це база даних в оперативній пам’яті. Він надає хмарні та локальні

рішення для кешування, векторного пошуку та баз даних NoSQL, які вписуються в будь-який стек технологій, що полегшує створення, масштабування та розгортання різноманітних програм [8].

В нашому випадку Redis використовується для кешування даних та покращення продуктивності, виконуючи дві основні функції, а саме: кешування сесій та запитів та блокування файлів. За рахунок цього прискорюється завантаження сторінок, зберігаючи проміжні дані в пам’яті, зменшується навантаження на нашу базу даних, зберігаючи частовикористовуванну інформацію в Redis, а також запобігає конфліктам при одночасному доступі до файлів.

2.2.6. MySQL

MySQL — база даних з відкритим кодом. Вона дозволяє зберігати, організовувати й обробляти великі обсяги інформації у структурованому вигляді. MySQL дозволяє зберігати дані в таблицях, додавати, змінювати та видаляти записи, об’єднувати дані з різних таблиць, а також швидко знаходити потрібну інформацію [9].

В роботі MySQL необхідний для роботи Nextcloud, так як останній не зберігає всі дані у своїй файловій системі. Отже, така інформація, як: інформація про користувачів, акаунти, права доступу, посилання на файли, налаштування додатків, логи, метадані файлів тощо, зберігаються за допомогою MySQL.

Також, оскільки ми розгортаємо MySQL в контейнері за допомогою Docker, ми підвищуємо безпеку, оскільки база даних доступна лише з Docker-мережі, у якої немає виходу в глобальну мережу.

2.2.7. Njinks

Nginx — це веб-сервер HTTP, зворотний проксі-сервер, кеш контенту, балансувальник навантаження, проксі-сервер TCP/UDP та проксі-сервер пошти [10].

В нашому випадку, ми використовуємо даний веб-сервер як reverse proxy (зворотній проксі)**.** Ми налаштували його так, щоб він приймав зовнішні

HTTP/HTTPS-запити від клієнтів. Тобто він “слухає” порт 80 (HTTP) і автоматично робить переадресацію на `443` порт (HTTPS). На цьому порті він вже працює з самопідписаним SSL-сертифікатом. Сертифікат створює шифрований TLS-тунель між клієнтом і Nginx, таким чином шифруючи весь трафік від клієнта до Nginx. Далі Nginx розпаковує цей HTTPS-трафік і переадресує його на внутрішній порт до контейнера Nextcloud у вигляді HTTP. Тобто він виступає "зовнішньою точкою входу" (entrypoint) для клієнтів.

2.2.8. Провайдер та біла IP адреса

Для забезпечення доступу до локального серверу з зовнішньої мережі було розглянуто декілька варіантів, але все ж таки важливу роль в забезпеченні доступу грає провайдер та те, як будується його мережа. У даному випадку довелося придбати у провайдера білу IP адресу для подальших налаштувань.

Для реалізації локального серверу було використано провайдера “Best”. “Товариство з додатковою відповідальністю Компанія «Бест» – регіональний телекомунікаційний оператор, що надає телекомунікаційні послуги на території м. Києва, Київської та Житомирської областей через власну оптоволоконну мережу та безпроводні канали зв’язку” [11].

Місячна плата за послуги провайдера за пакетом “Престижний” становить 350 гривень на місяць. Пакет включає в себе швидкість до 1 гігабіт на секунду та виділену IP адресу з мережі провайдера. Так як для забезпечення доступу необхідно мати білу IP адресу, дана послуга коштувала 50 гривень на місяць. Отже, загальні витрати на послуги провайдера для забезпечення доступу склали 400 гривень на місяць.

2.2.9. WireGuard

WireGuard - це сучасний VPN-протокол і одночасно реалізація VPN-тунелю.

WireGuard забезпечує швидке, безпечне та просте у налаштуванні та розгортанні VPN з’єднання. Він працює за рахунок публічного та приватного ключа, які потрібні для аутентифікації та шифрування з’єднання між клієнтом і

сервером. Спочатку відбувається процес обміну публічними ключами між клієнтом і сервером для встановлення зашифрованого тунелю. Приватний ключ в свою чергу зберігається локально та використовується для підпису та дешифрування трафіку. Таким чином, це забезпечує конфіденційність та цілісність з’єднання.

Також наявне використання криптографічний алгоритмів, таких як: ChaCha20 (для шифрування), Poly1305 (для футентифікації), Curve25519 для обміну ключами. Використання даних алгоритмів робить WireGuard швидким та безпечним [12].

Важливо зазначити, що WireGuard це відкрите програмне забезпечення, яке має відкриртий вихідний код, що дозволяє його вільно змінювати, використовувати та підлаштовувати під власні потреби та проекти [13].

Його особливістю також є кросплатформленість, що дозволяє його використовувати на будь-якій операційній системі (Windows, Linux, macOS, Android, IOS, тощо) [14].

WireGuard має зручний інтерфейс, він кросплатформений, легкий в налаштуванні, керуванні та впровадженні, з відкритим кодом, а також надійними алгоритмами шифрування. Саме це і робить його ідеальним рішенням для реалізації локальної серверної інфраструктури.

Висновки

1. У другому розділі було спроектовано модель сервера на віртуальній машині з віддаленим доступом до нього через VPN тунель, а також описано методи та утиліти які будуть використані для реалізації.

А саме було розглянуто віртуальне середовище та операційна система яка буде розгорнута, а також ресурси які необхідні для розгортання. Далі було змодельовано яким чином буде організовано віддалений доступ до сервера який розгорнутий на віртуальній машині, а також були розглянуті принципи безпеки підключення, а саме їх реалізація у вигляді підключення по протоколу HTTPS. Також коротко було описано інструменти та утиліти які будуть використані при

реалізації моделі.

2. Загалом, Розділ 2 теоретично описує модель для подальшої реалізації, а також інструменти які потрібні для цього. Реалізовувати дану модель, можна не тільки шляхом розгортання віртуальної машини, якщо є наявне серверне обладнання, то підхід з розгортання необхідного програмного забезпечення та реалізації VPN тунелю, які описаний в Розділі 2, можна застосувати і до серверного обладнання відповідно. З цього випливає, що дана модель може бути реалізована не тільки в домашніх умовах, а й наприклад в офісі компанії, де співробітникам потрібен віддалений та безпечний доступ до спільного файлового сховища.

РОЗДІЛ 3
МЕТОДИКА РОЗГОРТАННЯ ФАЙЛОВОГО СХОВИЩА З ВИКОРИСТАННЯМ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ NEXTCLOUD

3.1. Розгортання файлового сховища в локальній мережі

3.1.1. Створення віртуальної машини

З урахуванням того, що серверного обладнання, у нас нема в розпорядженні, будемо використовувати наявні ресурси. А саме наш домашній ПК з наявною пам’яттю, на якому буде розгорнута віртуальна машина.

Розглянемо процес розгортання. Як було зазначено раніше будемо використовувати програму віртуалізації Oracle VM VirtualBox.

В головному меню VirtualBox натискаємо кнопку “Створити”, та бачимо вікно (Рис 3.1), де вказуємо назву віртуальної машини, папку головної ОС де вона буде зберігатися та вказуємо шлях до iso файлу з операційною системою яку ми будемо використовувати на сервері. Як було зазначено раніше, було обрано Ubuntu server 24.04.1. Після заповнення необхідних полів переходимо до наступного етапу натиснувши “Next”.

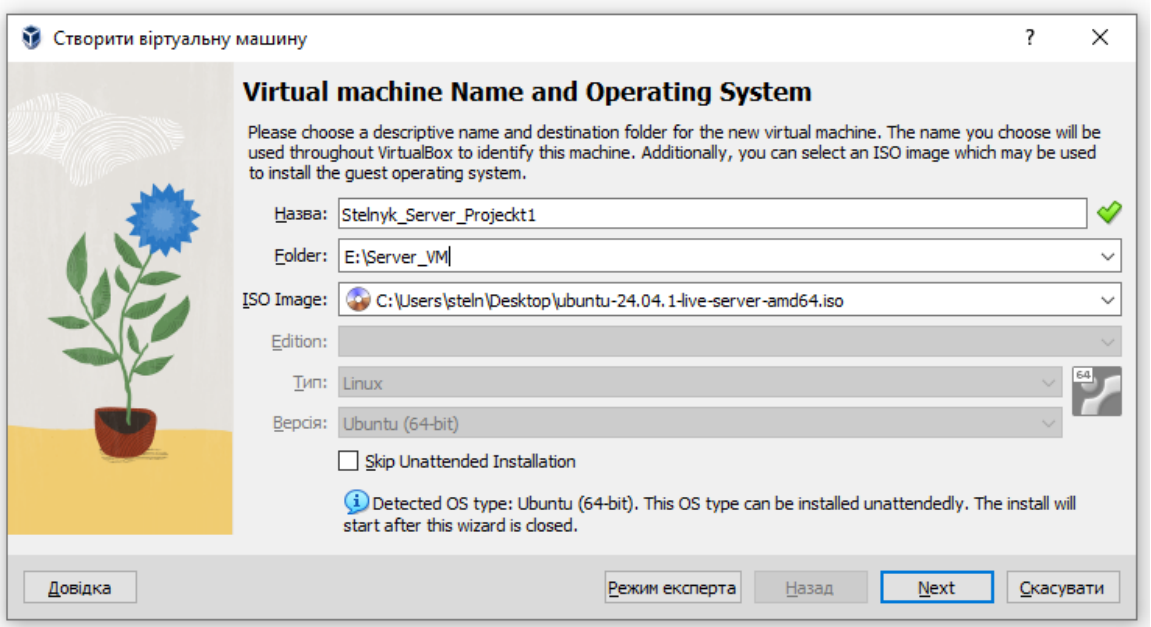

Рис. 3.1 Вікно створення віртуальної машини Ubuntu server

В наступному вікні (Рис 3.2) вказуємо ім’я користувача для системи, пароль, ім’я хоста та доменне ім’я.

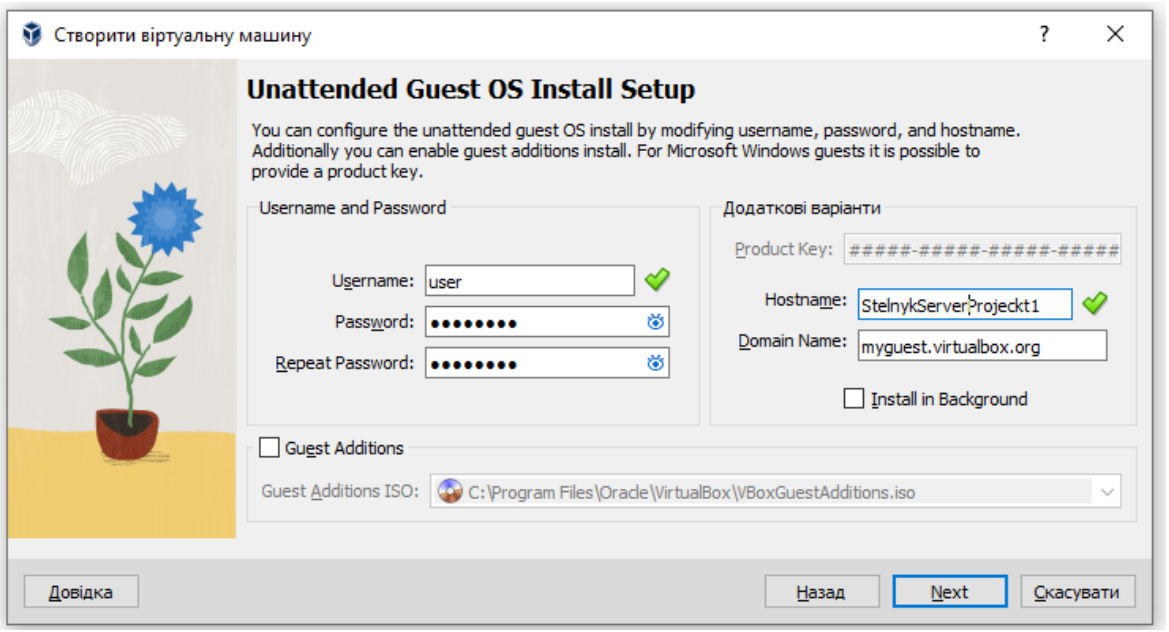

Рис. 3.2 Вікно налаштування користувача віртуальної машини Ubuntu server

Наступним кроком ми задаємо обсяг ресурсів які будуть виділені під віртуальну машину з операційної системи хоста, а саме обсяг оперативної пам’яті, кількість ядер та найголовніше в нашому випадку обсяг фізичної пам’яті (Рис 3.3, 3.4).

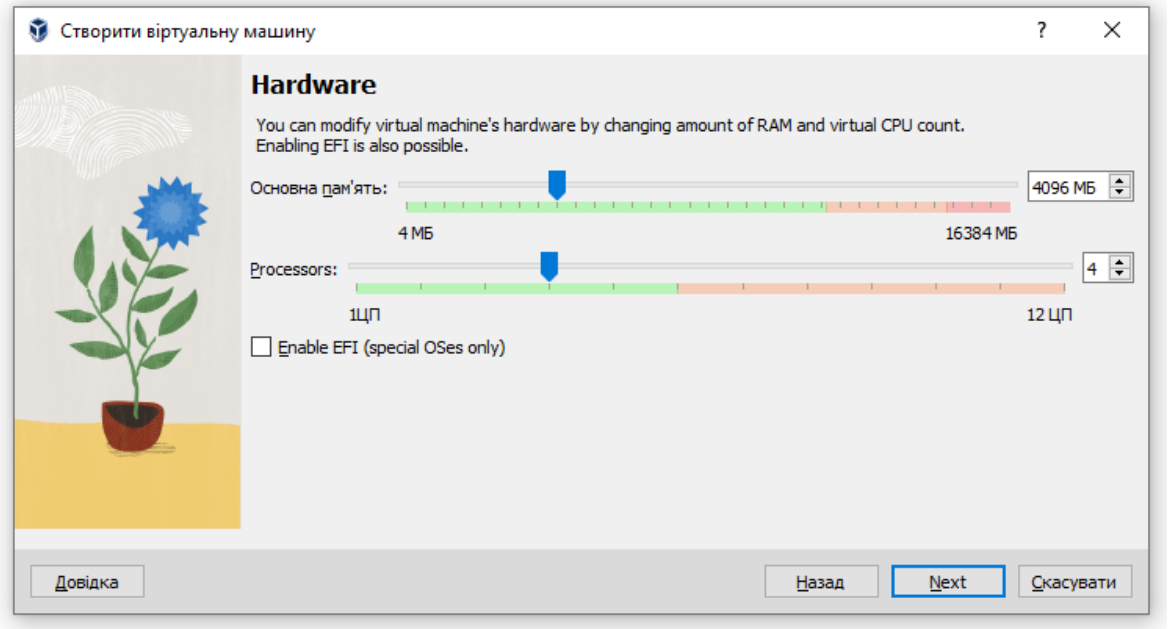

Рис. 3.3 Вікно налаштування оперативної пам’яті та кількості ядер для віртуальної машини Ubuntu server

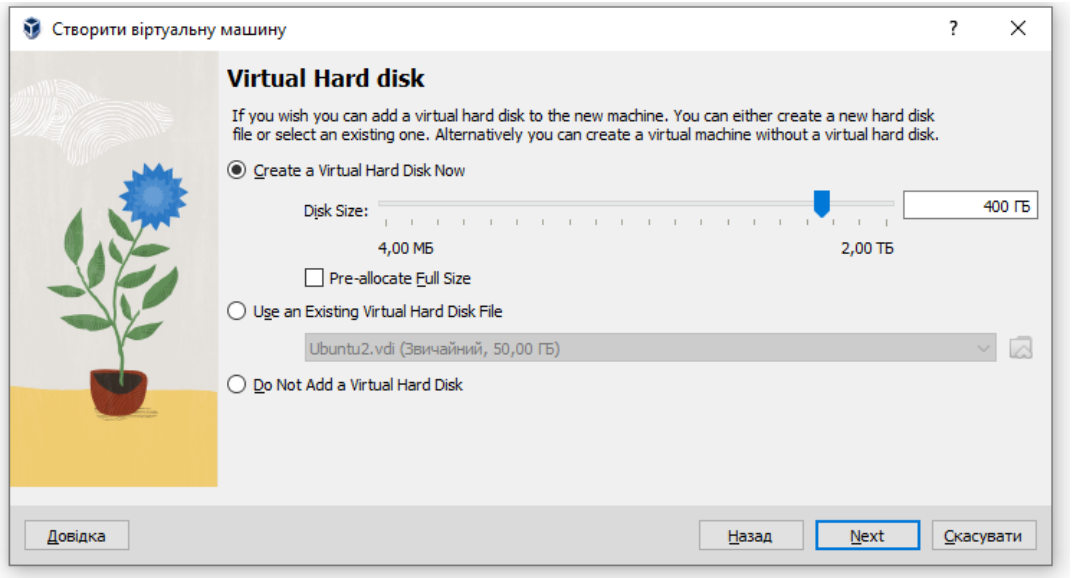

Рис. 3.4 Вікно налаштування обсягу фізичної пам’яті віртуальної машини Ubuntu server

Обсяг ресурсів, що ми виділяємо визначається за рахунок, наявних ресурсів на операційній системі хоста та мінімальних вимог гостьової операційної системи, які були наведені вище.

Так як першочергово ми збираємося використовувати данний сервер для зберігання інформації, то обсяг віртуального жорсткого диска є ледь не найважливішим ресурсом. Тому було прийняте рішення виділити аж 400 гігабайт пам’яті.

По завершенню вибору ресурсів, завершуємо створення віртуальної машини, після чого вона з’явиться в загальному списку віртуальних машин, де при її виборі, будуть коротко відображені характеристики (Рис. 3.5).

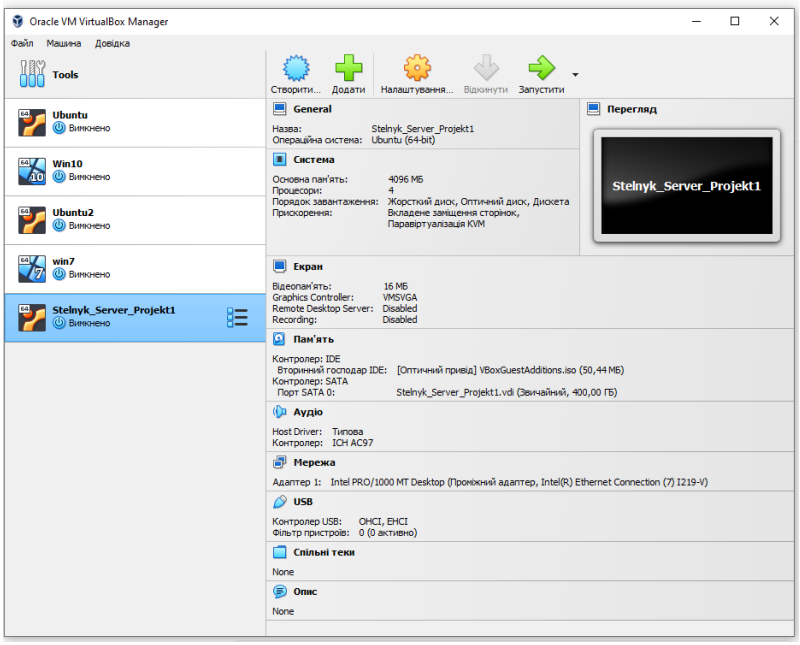

Рис. 3.5 Вікно з загальними характеристиками віртуальної машини Ubuntu server

Також важливим кроком в створення віртуальної машини, це налаштування мережевого адаптера. Тому, вибравши нашу віртуалку, переходимо у вкладку “налаштування”, далі “мережа” де у полі “під’єднанний до” обираємо “проміжний адаптер” (рис 3.6).

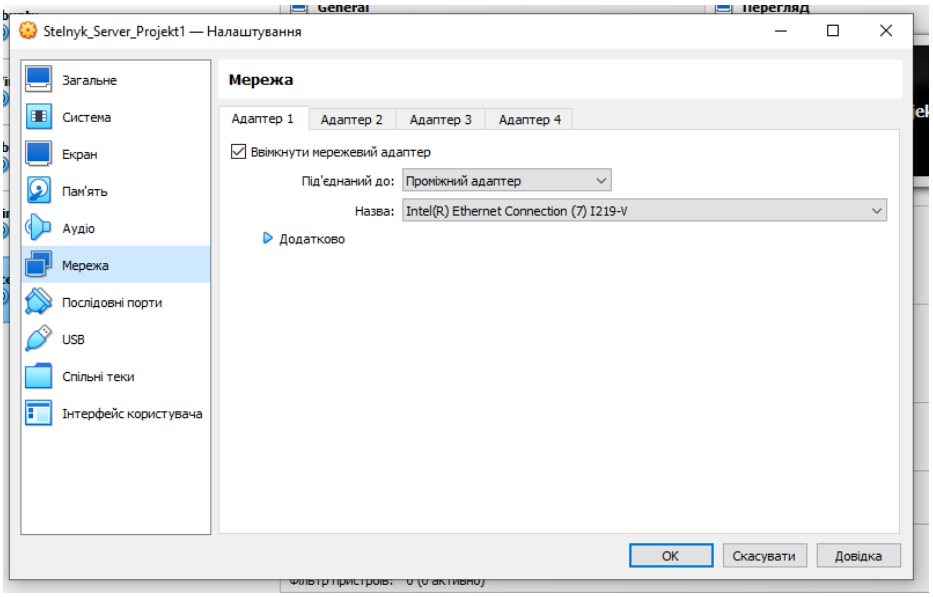

Рис. 3.6 Вікно налаштування мережевого адаптера віртуальної машини Ubuntu server

За рахунок даного налаштування, віртуальна машина буде отримувати IP адресу від домашнього роутера, так само як і інші члени локальної мережі, що є

дуже важливою частиною для наших подальших налаштувань.

Наступним кроком запускаємо віртуальну машину, де буде запущенно процес встановлення операційної системи Ubuntu server 24.04.1, образ якої ми вказували раніше (Рис 3.1). З важливих параметрів під час встановлення є те, що буде задано логін та пароль для системи, а також пароль для root користувача.

В результаті завершення встановлення операційної системи та її перезапуску ми отримаємо наступне вікно (Рис 3.7), де можна буде одразу побачити IP адресу яку отримав сервер від домашнього роутера, а саме 192.168.1.10.

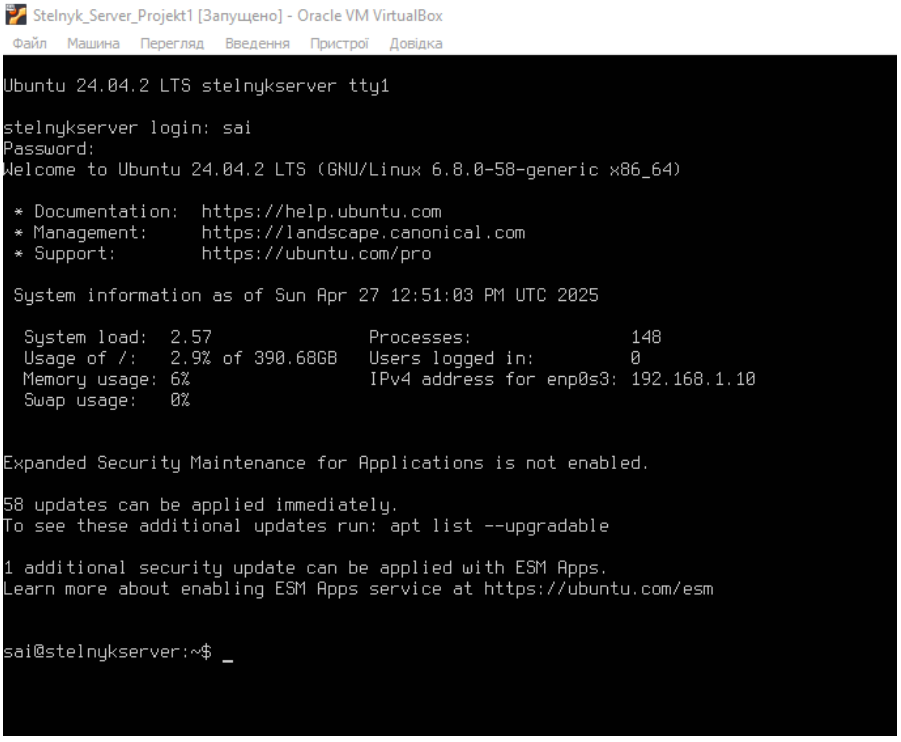

Рис. 3.7 Вікно запущенної віртуальної машини Ubuntu server

Для того щоб було зручніше користуватися інтерфейсом сервера, виконаємо вхід через програму PuTTY, чим підтвердимо доступність з локальної мережі. Для цього в відповідному полі вводимо IP адресу сервера в локальній мережі (Рис 3.8)

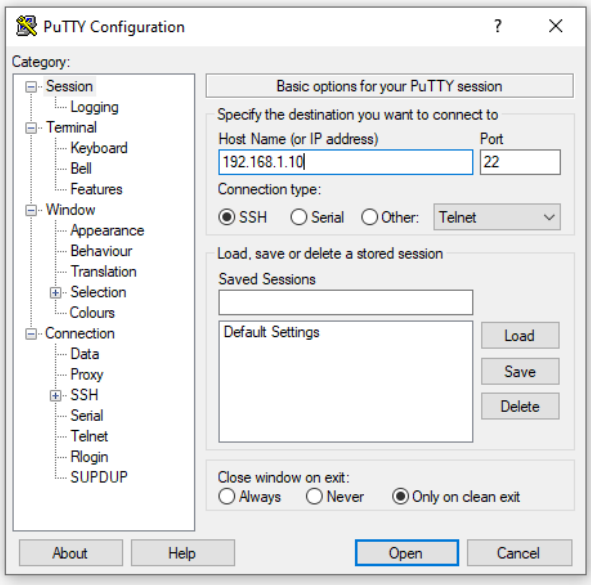

Рис. 3.8 Вікно входу до віртуальної машини Ubuntu server через програму PuTTY

Вхід звісно виконуємо по протоколу SSH для того щоб наші дані, а саме команди, передавалися в шифрованому вигляді, а також унеможливити перехват вводимих логіна і різноманітних паролів, наприклад.

Далі відкриваємо наше з’єднання, і бачимо що на сервер нас пускає, а отже сервер доступний в локальній мережі. Виконуємо логін, і можемо побачити, що кількість залогінених користувачів тепер 1, що вказує на наш вхід в систему для віддаленого управління. (Рис. 3.9)

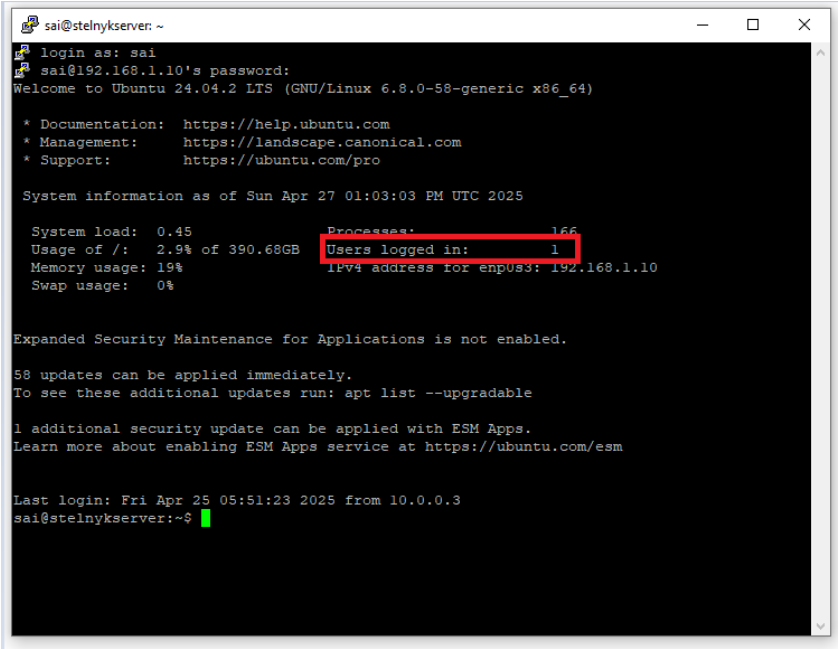

Рис. 3.9 Відображення входу в систему для віддаленого управління

На цьому етап створення віртуальної машини завершенно, переходимо до наступного етапу, розгортання сервісів за допомогою Docker контейнерів.

3.1.2. Розгортання та налаштування контейнерів

Перед початком роботи створимо директорію в які будемо працювати. Так як основним додатком для обробки файлів, буде некст клауд, назвемо папку іменем додатка. Отже створимо папку командою **mkdir nextcloud.**

Так як папку було створенно раніше, нам буде виведено повідомлення, що директорія вже існує. Тому для того щоб продемонструвати її, вводимо команду **ls** і бачимо нашу папку. (Рис. 3.10).

```
sai@stelnykserver:~$ mkdir nextcloud
mkdir: cannot create directory ‘nextcloud’: File exists
sai@stelnykserver:~$ ls
duckdns.log  nextcloud
```

Рис. 3.10 Створення папки з проектом

Для роботи з контейнерами встановимо Docker, який було згадано вище. Зробити це можна за допомогою команди:

sudo apt update && sudo apt install docker.io docker-compose

Так як докер було встановлено раніше, вивід команди вказує на те що докер встановленно, та він готовий до роботи, також в виводі є повідомлення що можна оновити 63 пакети, але ми цього не робитимемо, так як нема в тому необхідності (Рис. 3.11).

```
sai@stelnykserver:~/nextcloud$ sudo apt update && sudo apt install docker.io docker-compo
se
Hit:1 http://ua.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ua.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ua.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.6 kB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B
]
Get:7 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.3 kB
]
Get:8 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B
]
Fetched 200 kB in 1s (326 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
63 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker.io is already the newest version (26.1.3-0ubuntu1~24.04.1).
docker-compose is already the newest version (1.29.2-6ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 63 not upgraded.
sai@stelnykserver:~/nextcloud$
```

Рис. 3.11 Оновлення пакетів та втсановлення Docker

Далі для розгортання необхідних нам додатків ми скористаємося docker-compose для зручнішого управління контейнерами, де docker-compose.yml – це файл з інструкцією, де описано як запустити декілька контейнерів разом. Замість того щоб скачувати та запускати всі контейнери окремо великою кількістю команд, ми це все описуємо в одному місці, за допомогою цього файлу і запустимо все однією командою.

Для початку створимо docker-compose.yml файл за допомогою команди sudo nano docker-compose.yml (Рис. 3.12)

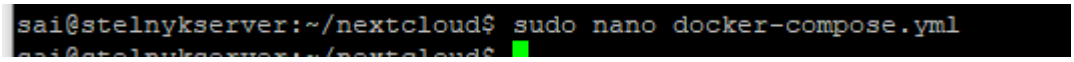

Рис. 3.12 Створення файлу docker-compose.yml

Відкриється текстовий редактор nano, в який нам необхідно прописати наступний код який вказаний в додатку А, для розгортання необхідних додатків.

В цілях безпеки логіни і паролі були замінені символом X. Також важливо зауважити що даний код дуже чутливий до пробілів, якщо бути не уважним і випадково десь поставити пробіл, код не буде працювати. Отже, розглянемо цей код детальніше. Даним файлом ми описуємо 4 контейнери.

Перший – база даних MySQL. Вона нам потрібна як основна база даних для збереження інформації про файли та систему, а саме: інформації про користувачів(логіни, паролі, email-и), права доступу, метаданні файлів(імена файлів, дати створення, зміни), налаштування некстклауд.

Якщо розбирати сам код:

services: - данною командою визначаємо перелік сервісів що будуть розгортатися;

db: - назва сервісу з базою даних;

image: mysql:8 – образ MySQL 8 що буде використовуватися;

container_name: nextcloud-db – назва контейнера;

restart: always – перезавантаження завжи;

command: --transaction-isolation=READ-COMMITTED --binlog-format=ROW - Спеціальна команда для налаштування поведінки транзакцій;

Через **environment** задаєм важливі змінні для MySQL;

volumes:

- db_data:/var/lib/mysql - Зберігає дані БД в Docker volume;

networks:

- nextcloud_network - Підключається до спільної мережі;

Другий контейнер, що розгортається – це Redis, кеш сервер. Він потрібен нам для кешування та прискорення роботи. Він зберігає тимчасові дані в оперативній пам'яті, тож наш основний додаток, не буде звертатися зайвий раз в базу даних. Також redis зберігає інформацію про активні сессії.

Отже, тут з нового:

redis: - назва сервісу;

image: redis:latest – назва образу контейнеру;

container_name: nextcloud-redis – назва контейнеру в системі;

Далі найважливіше, наш основний сервіс для зберігання та обробки файлів -Nextcloud, який детальніше був описаний вище.

app: - назва сервісу з основним додатком;

image: nextcloud:latest – назва образу, що буде використовуватися;

container_name: nextcloud-app – назва контейнеру в системі;

expose: - використовує внутрішній порт для інших контейнерів, не відкриваючи його ззовні, в даному випадку порт 80;

environment: - перелік змінних важливих для роботи сервісу;

MYSQL_DATABASE: nextcloud – змінна яка вказує яку базу даних використовувати;

REDIS_HOST: redis – змінна яка вказує використовувати Redis;

volumes:

- nextcloud_data:/var/www/html – вказуємо шлях де будуть зберігатися данні Nextcloud;

depends_on:

- db

- redis - має залежності від таких сервісів, тобто буде запускатися після

них.

Четвертий контейнер – вебсервер nginx. В нашому випадку використовується як reverse proxy для організації запитів по захищенному протоколу HTTPS, який шифрує з’єднання.

nginx: - назва сервісу з веб сервером.

image: nginx:alpine – назва образу вебсервера, що буде завантажена.

container_name: nginx-ssl – назва контейнера.

ports:

- "80:80"

- "443:443"- відкриває порти ззовні.

depends_on:

- app – залежить від основного сервісу та запускається після нього.

volumes: - створює файли де:

- ./nginx/default.conf:/etc/nginx/conf.d/default.conf - конфігурація самого вебсерверу.

- ./certs:/etc/nginx/certs:ro – сертифікати для SSL у режимі читання.

Остання частина команд це

volumes:

db_data:

nextcloud_data: - визначаємо значення для даних MySQL та Nextcloud.

networks:

nextcloud_network: - Всі сервіси спілкуються через окрему docker-мережу nextcloud_network, щоб бачити одне одного за іменами контейнерів (наприклад, db, redis, app), її ми і вказували в командах для кожного контейнера.

Також, важливо зазначити що всі дані будуть зберігатися в Docker volumes, тому не пропадуть навіть після перезапуску контейнерів.

Отже, коли docker-compose файл, було написано, можемо переходити до запуску контейнерів. Зробити це можна за допомогою команди **docker-compose up -d,** де -d запускає все у фоновому режимі. (Рис 3.13)

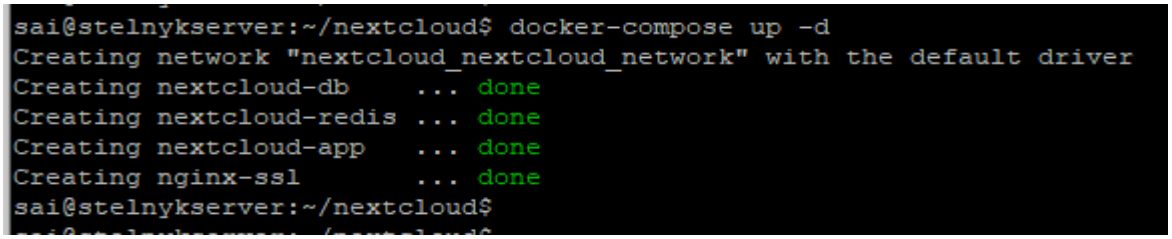

Рис. 3.13 запуск docker-compose.yml

Так як ці контейнери вже були попередньо мною створені, то весь процес відбувся доволі швидко, і вивід команди показує що контейнера запущені. Якщо робити це вперше, то через команди що були вказані в docker-compose файлі, Docker почне витягувати з інтернета образи цих контейнерів, та завантажувати їх в систему, через що, перший запус зазвичай відбувається довше.

Наступним кроком перевіряємо чи запущені наші контейнери за допомогою команди **docker ps** (рис 3.14)

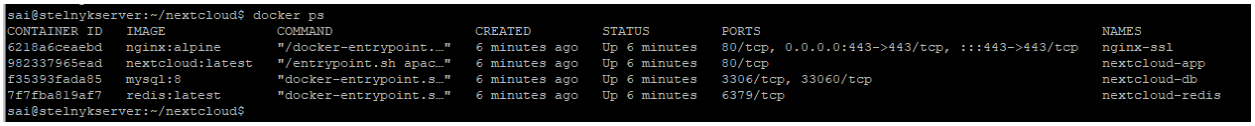

Рис. 3.14 запущені контейнери

Тут ми можемо побачити важливу інформацію, про використовувані контейнери, таку як: ID контейнерів, назву образу що було використано, команда, що використовується в середині контейнера, час створення контейнеру, його статус, тобто скільки він вже запущений, порти, та назви контейнерів.

Наступним кроком нам необхідно провести налаштування в самих контейнерах, перш за все, в нашому основному, тобто контейнер Nextcloud, а також контейнер nginx, де нам необхідно провести налаштування з його конфігураційним файлом. Так як ми вказали всі необхідні залежності та змінні для MySQL та Redis, проводити додаткові налаштування цих контейнерів не потрібно.

Перейдемо до налаштувань Nextcloud. Для того щоб зайти в середину контейнера, необхідно використати команду **docker exec -it nextcloud-app bash** (Рис 3.15)

```
sai@stelnykserver:~/nextcloud$ docker exec -it nextcloud-app bash
root@982337965ead:/var/www/html#
```

Рис. 3.15 Вхід в контейнер Nextcloud

Як видно з попереднього рисунку, після введення команди ми опиняємося в середині нової віртуальної системи, а саме контейнера. Так як контейнер це цілком інша система, для того щоб відкрити конфігураційний файл нам необхідно встановити текстовий редактора nano, який відсутній тут за замовчуванням. Робимо ми це за допомогою команди **apt update && apt install nano –y**, після чого у нас починається завантаження. (Рис 3.16)

```
root@982337965ead:/var/www/html/config# apt update && apt install nano -y
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8792 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [512 B]
Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [257 kB]
Fetched 9305 kB in 2s (5181 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
21 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  hunspell
The following NEW packages will be installed:
  nano
0 upgraded, 1 newly installed, 0 to remove and 21 not upgraded.
Need to get 690 kB of archives.
After this operation, 2871 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bookworm/main amd64 nano amd64 7.2-1+deb12u1 [690 kB]
Fetched 690 kB in 0s (3526 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package nano.
(Reading database ... 17302 files and directories currently installed.)
Preparing to unpack .../nano_7.2-1+deb12u1_amd64.deb ...
Unpacking nano (7.2-1+deb12u1) ...
Setting up nano (7.2-1+deb12u1) ...
update-alternatives: using /bin/nano to provide /usr/bin/editor (editor) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/editor.1.gz because associated file /usr/share/man/man1/nano.1.gz (of link group editor) doesn't exist
update-alternatives: using /bin/nano to provide /usr/bin/pico (pico) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/pico.1.gz because associated file /usr/share/man/man1/nano.1.gz (of link group pico) doesn't exist
root@982337965ead:/var/www/html/config#
root@982337965ead:/var/www/html/config#
```

Рис. 3.16 Встановлення nano в контейнері Nextcloud

Після завантаження можемо переходити у відповідну директорію і відкривати конфігураційний файл (Рис 3.17).

```
root@982337965ead:/var/www/html/config# nano config.php
```

Рис. 3.17 Відкриття конфігураційного файлу Nextcloud config.php

Далі відкриється безпосередньо сам конфігураційний файл Nextcloud, а саме config.php. Повний код, який потрібно прописати в цьому файлі, знаходиться в додатку Б.

Важливо зазначити, що при першому відкриті файл виглядатиме інакше, тут же в додатку знаходиться файл конфігурації Nextcloud в якому, вже проведенні певні налаштування. Також в цілях персональної безпеки, чутлива

інформація була замінена символом X.

Так як майже весь файл генерується автоматично при розгортанні контейнера, розглянемо лише ту частину, яку нам необхідно змінити.

Перш за все нам необхідно прописати '**trusted_domains**', а саме прописати IP адреса та домени з яких ми маємо доступ до нашого контейнера в мережі некстклауда. Так як '**trusted_domains**' представлений у вигляді списку з нового рядка ми просто прописуємо домени та IP адреси яким ми дозволяємо зайти на наш контейнер з Nextcloud. Це такий собі мережевий екран, а точніше айпі адреси яким ми довіряємо і дозволяємо доступ до контейнера.

'trusted_domains' =>

array (

0 => 'localhost', - домен локального хоста, щоб ми могли зайти через лупбек;

1 => 'nextcloud.local', - локальний домен некстклауда;

2 => '192.168.1.10', - IP адреса віртуальної машини де розташовано наш контейнер;

3 => '10.0.0.1' – IP адреса серверу в віртуальній приватній мережі WireGuard, про що детальніше буде описано пізніше;

Без вказання дозволених доменів та IP адрес, ми не зможемо попасти до Nextcloud. В браузері буде видавати помилку, що недостатньо прав.

Також варто зазначити рядки які ми додаємо до файлу для організації підключення по HTTPS.

'overwrite.cli.url' => 'https://10.0.0.1', - вказуємо для Nextcloud яку URL-адресу використовувати як основну для доступу.

'overwriteprotocol' => 'https', - вказує Nextcloud, що слід використовувати HTTPS

'trusted_proxies' => - список довірених проксі серверів

array (

0 => 'nginx-ssl' – вказуємо контейнер з njinx як довірений проксі сервер.

Після внесених змін, проводимо збереження конфігураційного файлу, та

можемо виходити з контейнера командою **exit.**

А тепер перейдемо до більш детальної організацію підключення HTTPS.

3.1.3. Налаштування HTTPS

Не дивлячись на те що наш канал із сервером і так зашифрований через VPN, додатковий захід захисту ніколи не буде зайвий. Також налаштування HTTPS дозволить в майбутньому легко маштабувати інфраструктуру і для інших сервісів які вимагають HTTPS та додатково захистити трафік між браузером та сервером.

HTTPS буде налаштовано за допомогою self-signed сертифіката та вебсерверу nginx який буде використовуватися як reverse proxy. Він буде приймати HTTPS запити, далі розшифровує їх за допомогою вищезазначеного self-signed сертифіката і передавати до Nextcloud в середині мережі Docker у вигляді HTTP запитів. Використовуючи Docker compose було проведено розгортання контейнеру вебсервера nginx. Зараз розглянемо процес його налаштування, а також процес створення self-signed сертифіката.

Для початку нам необхідно створити самопідписаний сертифікат. Це можна зробити за допомогою команди:

**openssl req -x509 -nodes -days 365 -newkey rsa:2048 **

**-keyout . /nextcloud/certs/selfsigned.key **

**-out /nextcloud/certs/selfsigned.crt **

Для того щоб виконати цю команду необхідно попередньо встановити утиліту openssl. А тепер розглянемо дану команду більш детально. **openssl** викликає попередньо встановлену утиліту. **req** – запускає генерацію запиту на сертифікат. **-x509** створює самопідписаний сертифікат. **-nodes** не шифрувати приватний ключ щоб Nginx міг його використовувати без запиту пароля. **-days 365** задаємо кількість днів скільки буде дійсний сертифікат. **-newkey rsa:2048** – створює приватний і публічний ключі довжиною в 2048 біт. **-keyout** шляг до місця зберігання приватного ключа. **-out** шлях до місця зберігання сертифікату.

Таким чином, в результаті виконання даної команди ми отримаємо два файли. (Рис 3.18)

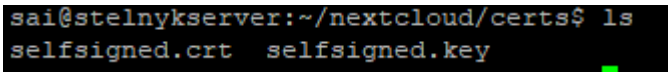

Рис. 3.18 Демонстрація створених файлів з сертифікатами

Далі необхідно провести налаштування вебсерверу nginx. Для цього заходимо в файл конфігурації яким ми вказували попередньо при створені контейнеру, а саме **./nginx/default.conf**. Ми також створили файл який автоматично підключений до контейнера, а саме **/etc/nginx/conf.d/default.conf**, тому ми можемо його відкрити та змінювати прямо з хоста. Переходимо до файлу конфігурації веб серверу nginx, default.conf та прописуємо там наступний код, який вказано в додатку В.

Розглянемо код детальніше. Перший блок **server** призначений для організаціїї перенаправлення HTTP в HTTPS. **listen 80** прослуховується порт HTTP, далі за допомогою **server_name _;** приймає усі домени, та завдяки **return 301 https://$host$request_uri;** перенаправляє на ту ж адресу але вже на HTTPS. Тобто завдяки даному блоку, ми перенапраляємо клієнта автоматично на HTTPS навіть якщо він намагається попасти на 80 порт, тобто HTTP.

Другий блок **server** призначений для організації онсовного HTTPS серверу. Командою **listen 443 ssl;** ми вказуємо слухати порт HTTPS, **server_name_;** також вказуємо приймати усі домени. **ssl_certificate /etc/nginx/certs/selfsigned.crt;** та **ssl_certificate_key /etc/nginx/certs/selfsigned.key;** вказують на шляхи до самопідписаного сертифіката та ключа які забезпечують шифрування HTTPS.

ssl_protocols TLSv1.2 TLSv1.3;

ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256';

ssl_prefer_server_ciphers on;

Даним блоком команд вказуємо параметри ssl.

В блоці **location** налаштовуємо вже проксі до Nextcloud, тобто

перенаправляємо всі запити до контейнера.

add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always; - вказуємо браузеру завжди використовувати HTTPS, навіть якщо користувач вводить HTTP.

Після проведених налаштувань, зберігаємо конфігураційний файл, перезапускаємо контейнери та можемо тестувати підключення до нашого серверу.

3.1.4. Перевірка та аналіз результатів налаштувань в LAN

Для перевірики проведених налаштувань, спробуємо зайти на сервер з локальної мережі. Для цього з будь-якого пристрою, що знаходиться в локальній мережі, в даному випадку - хостова система, необхідно перейти до браузеру та в полі пошуку вбити IP адресу віртуальної машини в локальній мережі. Як було зазначено раніше, в даній моделі, дана IP адреса це 192.168.1.10. Вбиваємо дану IP в пошукову строку, виконуємо вхід на серевер, та опиняємося в головному меню нашого файлового сховища (Рис 3.19)

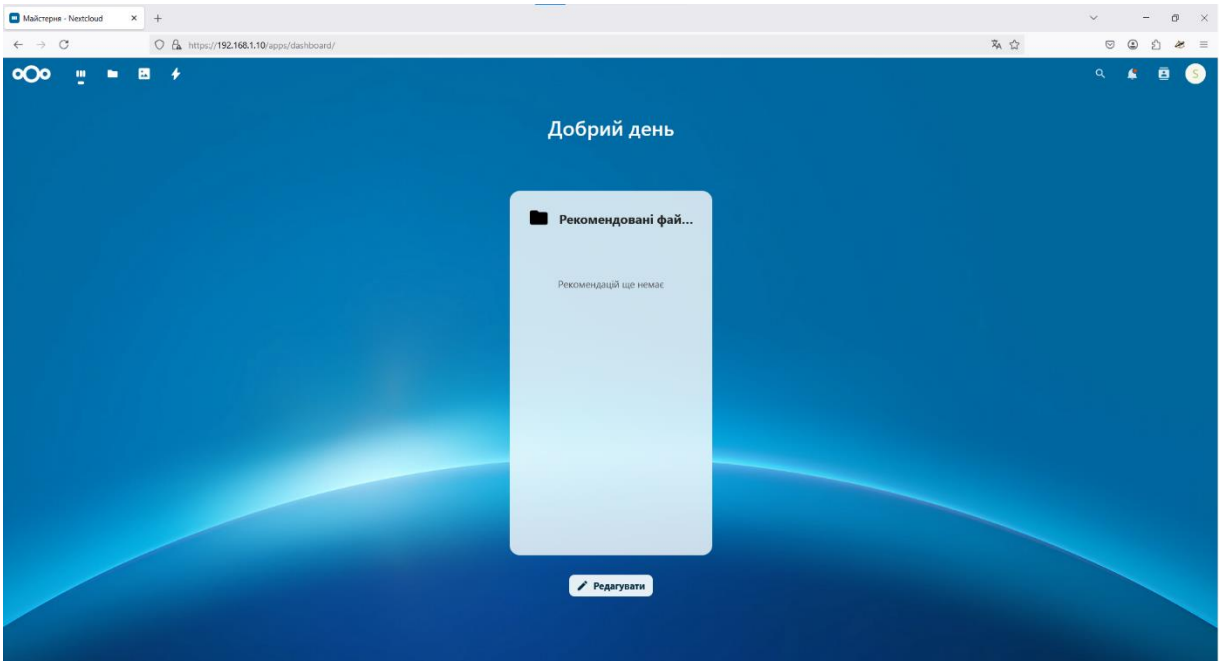

Рис. 3.19 Головне меню файлового сховища

Як можна зауважити вхід було виконано успішно. Тут вже можна додавати редагувати та проводити потрібні операції з файлами.

Окремо варто виділити пошукову строку, так як вона може багато чого

показати. Перш за все, було успішно виконано вхід з локальної мережі на локальну IP адресу, підтвердженням чого є вище зазначена локальна IP віртуальної машини (Рис 3.20). А також факт успішного входу підтверджує що нашаштування сервісу Nextcloud, були виконані правильно.

Також на рисунку 3.20 ми можемо побачити те, що вхід на сервер відбувся по протоколу HTTPS, що підтверджує правильність відповідних налаштуваннь.

За замовчуванням якщо в браузері ввести просто IP адресу, пошук відбуватиметься за протоколом HTTP, а так як у нас вхід було виконано за протоколом HTTPS, це підтверджує, що налаштування преадресації з HTTPS на HTTPS на веб сервері nginx, були виконані також правильно.

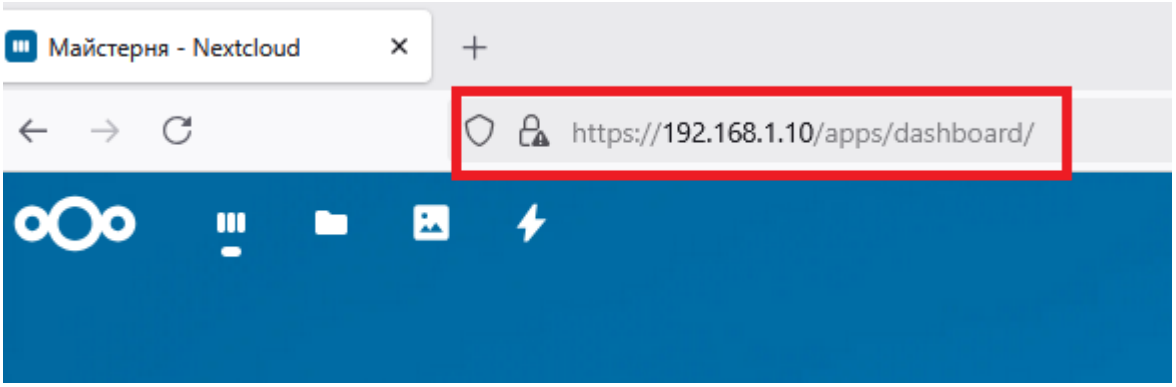

Рис. 3.20 Успішний вхід на сервер в середині LAN по протоколу HTTPS

3.2.Налаштування VPN для доступу до файлового сховища

3.2.1. Налаштування домашнього роутера та білої IP адреси

Як було зазначено раніше, для організації VPN підключеня ми будемо використовувати VPN-протокол WireGuard.

Перед початком налаштування WireGuard необхідно провести декілька попередніх налаштувань, а саме: налаштуванти проброс портів на роутері, також на роутері закріпити локальну IP адресу за віртуальною машиною та отримати білу IP адресу у провайдера. Це нам необхідно зробити для того щоб наш сервер отримав доступ в глобальну мережу та навпаки, щоб з глобальної мережі можна було отримати доступ до сервера.

Почнемо з проброса портів. Він нам потрібен для того щоб клієнти ззовні могли встановити з’єднання з сервером WireGuard. Без пробросу портів роутер

не знатиме, куди направити вхідний трафік, і VPN-з’єднання не буде встановлено. Це відбувається тому, що на будь-якому домашньому роутері, налаштований PAT, для загальної економії IP адрес, так як він дозволяє багатьом пристроям виходити в мережу з однієї зовнішньої IP адреси. Коли пристрій в середині LAN робить запит в інтернет, роутер змінює IP пристроя з якого надходить запит на свою зовнішню IP, додає унікальний номер порта та зберігає відповідність IP та порту в своїй таблиці. Відповідь приходить на цей IP та порт, і саме тому роутер знає кому передати IP пакети назад в внутрішній мережі.

Отже, коли пристрій ззовні хоче підключитися за зовнішньою IP адресою, він надсилає запит на цю зовнішню адресу та потрібний йому порт запит. Домашній роутер, якому належить зовнішня IP адреса, отримує цей запит і не знає кому з локальної мережі переслати цей запит, бо не зберігав жодного запиту в свої табличці у зворотньому напрямку і як результат, блокує всі вхідні з’єднання для безпеки. Саме тому налаштування пробросу портів є важливою частиною, тому перейдемо до самого налаштування.

Для початку необхідно дізнатися IP адресу домашнього роутера. Зазвичай за замовчуванням стоїть адреса 192.168.1.1, але для більшої впевненності зробимо коротку перевірку, так як бувають виключення. Для цього відкриємо консоль в хостовій системі, яка під’єднана до локальної мережі. За допомогою команди **tracert** яка допомагає відстежити маршрут пакетів до цільового хоста, ми дізнаємося IP шлюза, що і буде адресою нашого роутера, так як в маршруті пакетів, адреса шлюза має бути першою (Рис 3.21). Цільовим хостом було обрано IP адресу DNS серверу компанії GOOGLE, а саме 8.8.8.8.

```
C:\Users\steln>tracert 8.8.8.8

Tracing route to dns.google [8.8.8.8]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.1.1
  2     5 ms    <1 ms     1 ms  gateway.local [95.158.8.1]
  3     6 ms    <1 ms    <1 ms  95.158.0.206.gateway [95.158.0.206]
  4     6 ms     1 ms     1 ms  142.250.173.134
  5     6 ms     2 ms     2 ms  142.250.238.57
  6     5 ms     1 ms     1 ms  74.125.245.64
  7    21 ms    15 ms    15 ms  142.251.224.76
  8    19 ms    16 ms    16 ms  142.251.77.180
  9    19 ms    16 ms    16 ms  192.178.72.143
 10    19 ms    15 ms    15 ms  142.251.65.217
 11    18 ms    15 ms    14 ms  dns.google [8.8.8.8]

Trace complete.
```

Рис. 3.21 Визначення IP адреси домашнього роутера

Отже, базуючись на виводі команди, можемо зробити висновок, що як і припускалося IP адреса шлюза, тобто нашого домашнього роутеру стоїть така сама як і за замовчуванням, тобто 192.168.1.1.

Наступним кроком перейдемо, до безпосереднього налаштування маршрутизатора. Для цього в пошуковій строці браузера вбиваємо визначену IP адресу. Після цього нам відкриється стандартне вікно налаштування домашнього роутера (рис 3.22), в нашому випадку це маршрутизатор netis WF2780.

Рис. 3.22 Вікно налаштування домашнього роутера

З міркувань персональної безпеки SSID та паролі було приховано. Далі переходимо у вкладку розширених налаштувань (Advanced) та переходимо до пункту “Переадресація”, де обираємо владку “Віртуальний сервер”. В данній вкладці заповнюємо формочку відповідними даними. Вписуємо опис,

промисуємо IP адресу яку отримала віртуальна машина, обираємо протокл UDP, бо саме він потрібен для WireGuard, та вписуємо зовнішній і внутрішній порт для WireGuard, а саме 51820. Після чого нажимаємо “Додати” і проброс портів буде налаштовано (Рис 3.23)

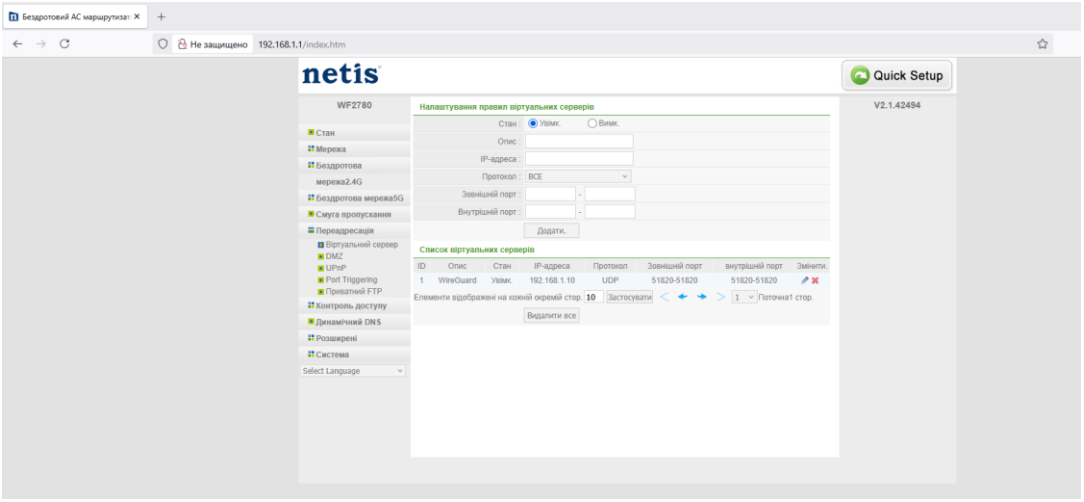

Рис. 3.23 Налаштування пробросу портів на домашньому роутері

Наступним кроком необхідно закріпити за нашою віртуальною машиною IP адресу яку вона отримала від маршрутизатора. Якщо ми цього не зробимо, при перезавантажені віртуальної машини, або роутера, або хостової системи, за рахунок DHCP відбудеться перерозподіл IP адрес в локальній мережі, і ту IP адресу яка була у віртуальної машини, може отримати інший пристій в середині LAN. Якщо таке відбудеться то вище налаштований проброс порту буде діяти вже до нового пристрою, а не до нашого серверу, за рахунок чого буде втрачено доступ. Отже, для уникнення такої ситуації закріпимо айпі адресу за віртуальною машиною. Для цього на роутері в меню розширених налаштувань, переходимо до вкладки “Мережа” і до пункту “LAN”. Відкриється список клієнтів DHCP де буде відображено всі пристрої в локальній мережі та IP адреса які було їм видано. Серед них знаходимо сервер, та нажимаємо на значок ланки ланцюга, чим і закріпимо IP за сервером. Після цієї дії в вкладці “Зарезервовано” з’явиться прапорець “так” (Рис 3.24)

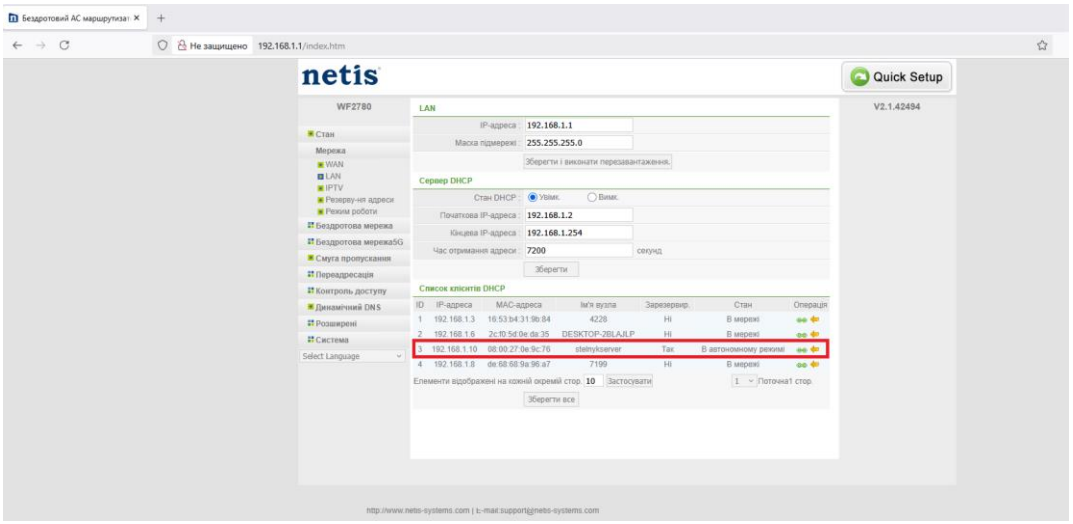

Рис. 3.24 Призначення статичної IP адреси для віртуальної машини

Після цього, як було описано раніше, нам потрібна біла IP адреса. Її можна отримати виключно шляхом звернення до свого провайдера. Для цього було сконтактовано з представниками компанії Бест, та оновлено пакет послуг за яким сплачується щомісячна оплата за отримання білої IP адреси. Факт отримання білої IP адреси можна перевірити на самому роутері в розширених налаштуваннях у вкладці “Стан” (рис 3.25) де, як ми бачимо маршрутизатор дійсно отримує білої IP адресу.

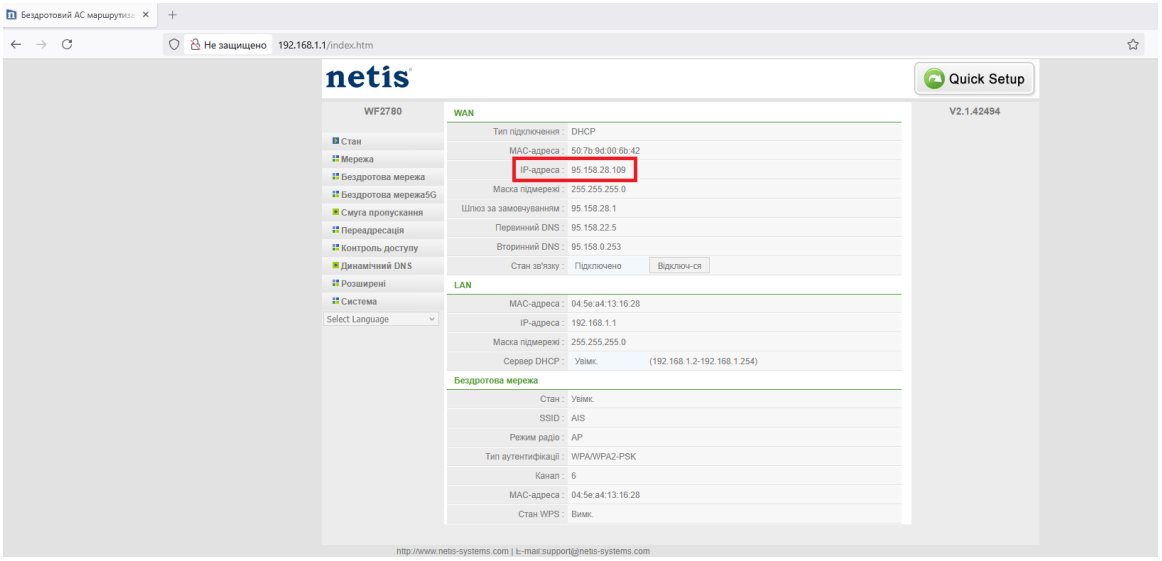

Рис. 3.25 Перевірка білої IP адреси на маршрутизаторі

3.2.2. Розгортання WireGuard

Коли підготовчі налаштування зроблено, розглянемо процес налаштування WireGuard. На сервері прописуємо команди **sudo apt update** для оновлення

списку доступних пакетів з репозиторіїв і після неї **sudo apt install wireguard** яка вже безпосередньо встановлює WireGuard (рис 3.26). Так як WireGuard вже був попередньо встановлений, вивід другої команди повідомляє нас про те, що найновіша версія цієї програми вже встановленна.

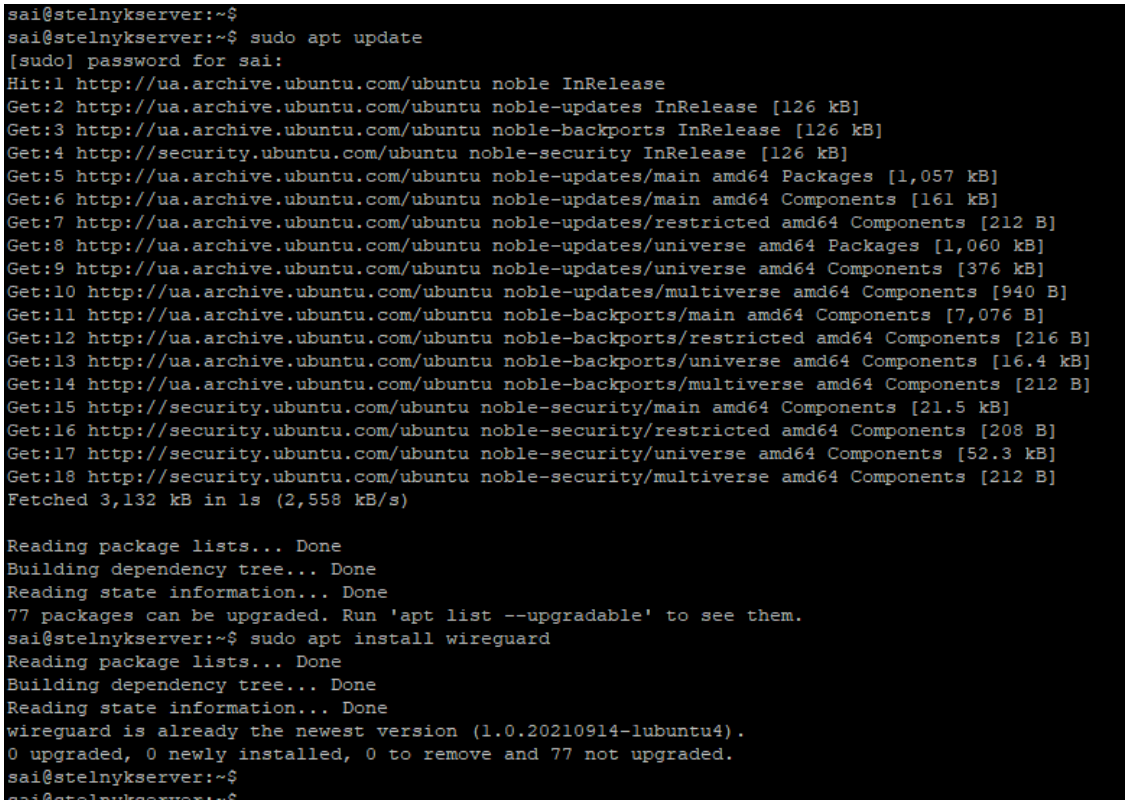

Рис. 3.26 Розгортання WireGuard на сервері

Далі за допомогою команди **wg genkey | tee privatekey | wg pubkey > publickey** генеруємо ключі шифрування для WireGuard. Після цього створюємо конфігураційний файл для WireGuard за допомогою команди **sudo nano /etc/wireguard/wg0.conf** і у нас відкриється текстовий редактор з порожнім конфігураційним файлом wg0.conf, де ми прописуємо наступну конфігурацію:

```
    [Interface]
Address = 10.0.0.1/24
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -
A FORWARD -o wg0 -j ACCEPT; iptables -t nat -A
POSTROUTING -o enp0s3 -j MASQUERADE
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables
-D FORWARD -o wg0 -j ACCEPT; iptables -t nat -D
```

```
POSTROUTING -o enp0s3 -j MASQUERADE
ListenPort = 51820
PrivateKey = XXXXXXX

[Peer]
PublicKey = 1ml5y3ipU4zrW6MHKg2RFGVUP28p7pB470yhLkJ8BEU=
AllowedIPs = 10.0.0.2/32

[Peer]
PublicKey = O+udwSrn+hvbaMQYXdpVE5/2Q+tC9y3BmPsZxn5w2Ss=
AllowedIPs = 10.0.0.3/32
```

З міркувань персональної безпеки приватний ключ шифрування було замінено на символи X. Отже, розберемо код детальніше.

[Interface] – налаштування локального інтерфейсу;

Address = 10.0.0.1/24 – адреса нашого серверу в мережі WireGuard та маска мережі;

PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -A FORWARD -o wg0 -j ACCEPT; iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE

PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -D FORWARD -o wg0 -j ACCEPT; iptables -t nat -D POSTROUTING -o enp0s3 -j MASQUERADE – дані два рядки вказують на те, що відбувається коли ми вводимо команди для підняття або перед зупинкою інтерфейсу VPN, а саме **wg0**. В першому рядку, після введення команди для підняття інтерфейсу VPN, додається правило в ланцюг **FORWARD** для прийому трафіку, що надходить та виходить через інтерфейс **wg0**, що дозволяє забезпечити двосторонню передачу даних між VPN клієнтами та внутрішньою мережею. Далі додаємо правило в таблицю **nat**, ланцюг **POSTROUTING**. Це правило застосовує маскарадинг, тобто підміну IP адреси джерела, до всього трафіку, що виходить через фізичний

інтерфейс **enp0s3.** Це дозволяє пристроям, які підключені до Wireguard виходити в інтернет через сервер на якому і відбувається розгортання.

Відповідно, другим рядком відбувається видалення тих правил, що були додані при запуску.

ListenPort = 51820 – вказуємо порт який використовує WireGuard, попередньо ми його прокидували на маршрутизаторі;

PrivateKey = - вказуємо приватний ключ шифрування з попередньо створеного файлу privatekey;

[Peer] – перше підключення

PublicKey = - вказуємо публічний ключ першого підключення.

AllowedIPs = 10.0.0.2/32 - адреса та маска другого клієнта в мережі WireGuard;

[Peer] – друге підключення;

PublicKey = - вказуємо публічний ключ другого підключення;

AllowedIPs = 10.0.0.3/32 - адреса та маска другого клієнта в мережі WireGuard.

Маска в підключенні відповідає не за підмережу, а саме за конкретну IP адресу, для того щоб одна адреса належала лише одному конкретному пристрою.

Після завершення налаштувань, зберігаємо та закриваємо файл з конфігурацією. Наступним кроком необхідно провести заходи захисту файлів де лежать наші ключі шифрування, а також сам файл конфігурації Wireguard. Це треба зробити за допомогою команди **chmod 600 /etc/wireguard/wg0.conf** для файлу конфігурації та команди **chmod 600 privatekey** для файлу з приватним ключем. Таким чином ми змінюємо права доступу до цих файлів, що лише власник може їх читати та змінювати, інші ж користувачі чи групи не мають жодног доступу.

3.2.3. Налаштування клієнтів

Тепер необхідно під’єднати клієнтів до VPN мережі. Почнемо з конфігурації мобільного пристрою.

В презентованій моделі було використано пристрій, з операційною системою IOS 16.5.1, тому процес конфігурації буде описаний саме під дану операційну систему. Отже, нам необхідно встановити застосунок WireGuard. Встановлення застосунку здійснюється шляхом його завантаження з офіційного магазину App Store через введення назви у пошук та ініціацію інсталяції. По завершенню інсталяції, відкриваємо застосунок та в лівому верхньому кутку нажимаємо на плюс. (Рис 3.27)

Рис. 3.27 Розгортання WireGuard на клієнті з операційною системою IOS

В результаті чого відкриється меню налаштування клієнта, де необхідно заповнити відповідні поля. В полі **Name** вказуємо назву нашого підключення. В секції нижче натискаємо кнопку "**Generate keypair**", тим самим ми згенеруємо ключі шифрування для клієнту, де згенерований **Public key** вставляємо в конфігураційний файл WireGuard на сервері, а саме в поле **PublicKey** першого підключення **[Peer]**. В полі **Addresses** вказуємо IP адресу клієнта яку ми вказували на сервері для клієнта в тому ж таки першому підключенні та маску мережі в яку він входить, для того щоб він міг "бачити" інших клієнтів в мережі VPN. У даній конфігурації параметри ListenPort та MTU можуть залишатися з автоматичними значеннями, оскільки їхня конкретна фіксація не є необхідною та не впливає на VPN підключення в нашому випадку. В полі DNS servers, вказуємо DNS сервер GOOGLE, а саме 8.8.8.8 . (Рис 3.28)

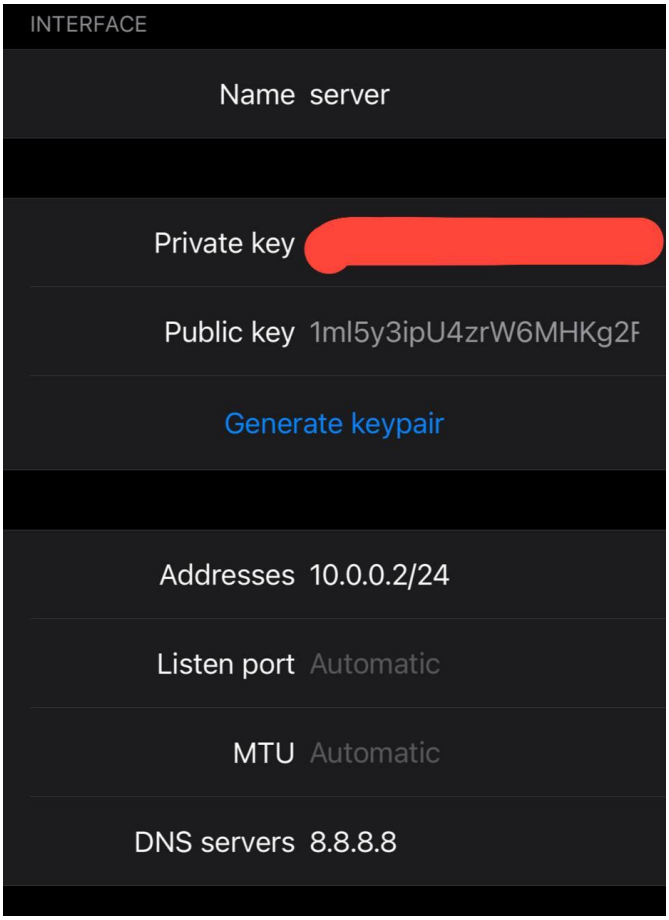

Рис. 3.28 Налаштування інтерфейсу Wireguard на клієнті

В наступному розділі налаштування **Peer**, в полі **Public key**, ми вказуємо публічний ключ сервера, до якого і відбувається підключення. В полі **Preshared key** залишаємо Optional. В полі **Endpoint** нам необхідно вказати IP адресу яку клієнту необхідно шукати в глобальній мережі, а саме нашу білу IP адресу яка в моделі надається від провайдера до домашнього маршрутизатора, та порт, проброс якого було зроблено попередньо. Так як сервер схований за PAT, то якщо вказати адресу саме сервера, клієнт просто не зможе підключитися. В наступне поле **Allowed IPs** вносимо значення 0.0.0.0/0 . За рахунок цього клієнт буде намагатися перенаправити весь свій трафік через сервер, і вийти в глобальну мережу вже від імені сервера. Всі інші поля залишаємо без змін. (Рис. 3.29)

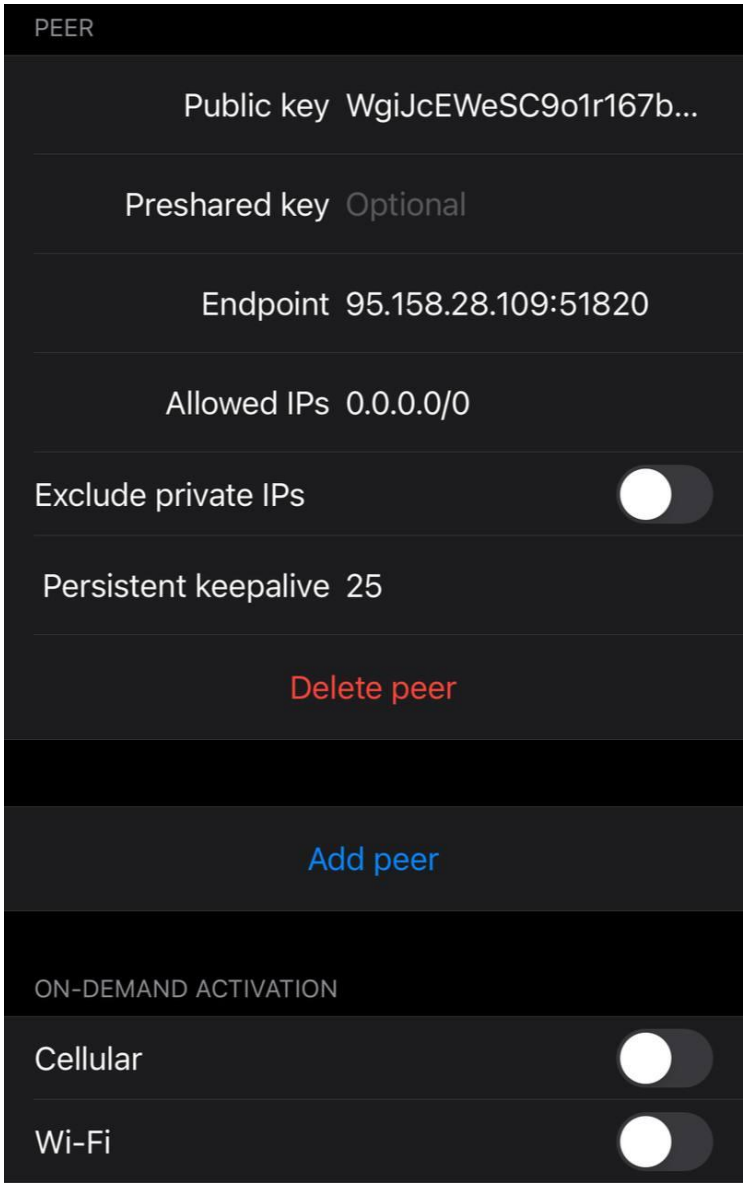

Рис. 3.29 Налаштування підключення до серверу на клієнті

Наступним кроком підєднаємо другого клієнта, до VPN мережі. Це буде ноутбук на операційній системі Windows. Для цього з офіційного сайту WireGuard [14], завантажуємо відповідне програмне забезпечення. Після розгортання застосунку на пристрої, відкриваємо його, та в лівому нижньому кутку, наводимо на стрілочку та натискаємо “Add empty tunel…”. (Рис 3.30.)

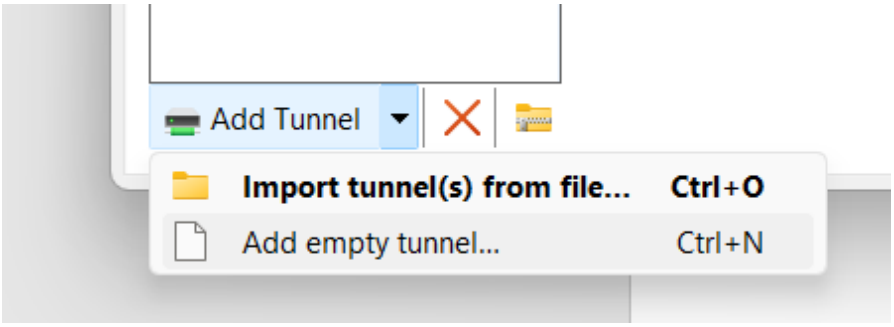

Рис. 3.30 Створення тунелю на клієнті з операційною системою Windows

Після цього в нас відкриється вікно з конфігураційним файлом клієнта. В даному файлі автоматично буде згенеровано публічний та приватний ключі, одразу після відкриття цього файлу, а також в полі **Назва** вводимо назву нашого підключення. Для налаштування підключення прописуємо наступний код (Рис 3.31)

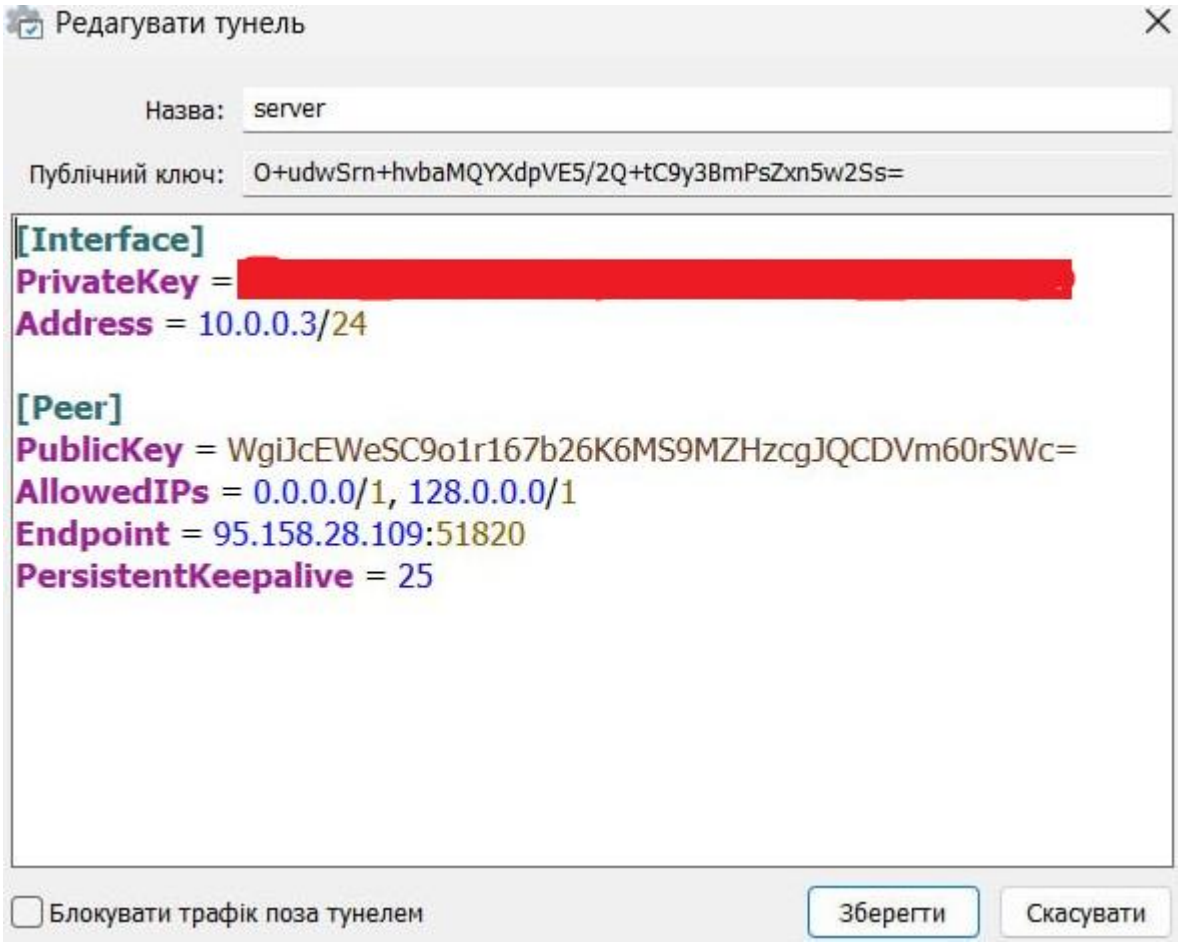

Рис. 3.31 Налаштування тунелю на клієнті з операційною системою Windows

Данні налаштування практично ідентичні до тих, що ми проводили для попереднього підключення. У нас є **[Interface]** – налаштування локального

інтерфейсу, де у нас автоматично було згенеровано **PrivateKey**, для персональної безпеки його було замальовано. Наступний рядок в налаштуванні локального інтерфейсу це **Address**, тобто IP адресса клієнта в мережі VPN. Вказуємо IP адресу клієнта яку ми вказували на сервері для клієнта в другому підключені, тобто 10.0.0.3 з маскою 24, для того щоб помістити його в одну мережу з іншими клієнтами, та надати можливість при необхідності обмінюватися інформацією з ними. В налаштуванні підключення **[Peer]**, проводимо такі самі налаштування як і на попередньому клієнті. Тобто, для змінної **PublicKey** вказуємо публічний ключ шифрування серевера. Для змінної **Allowed IPs** ставимо значення 0.0.0.0/0 для того щоб і цей клієнт направляв весь трафік через сервер. Змінній **Endpoint** задаємо кінцеву білу IP адресу та порт до яких трба підключитися для встановлення з'єднання з сервером. **PersistentKeepalive** залишаємо без змін. Важливим кроком налаштування також є те що в даному вікні налаштування, в лівому нижньому кутку треба прибрати галочку біля напису "Блокувати трафік поза тунелем", тому, що якщо не зробити цього, при підключенні до сервера через VPN ми не зможемо вийти в глобальну мережу, а точніше не зможемо отримати відповідь. Якщо не прибрати дану галочку комунікація відбуватиметься виключно з сервером.

Коли налаштування на клієнтах було проведено можна перевіряти з'єднання. Для цього вмикаємо VPN на клієнті з операційною системою IOS та IP адресою в середині VPN 10.0.0.2 (рис 3.32) а також вмикаємо VPN на клієнті з операційною системою Windows IP адресою в середині VPN 10.0.0.3 (рис 3.33). Одразу можна побачити хороший знак того, що налаштування були проведені правильно, так як при включенному з'єднанні, в головному меню застосунку Wireguard, на обох клієнтах показано кількість трафіку яка передається та була отримана по з'єднанню. Проведемо перевірку з'єднання на самому сервері.

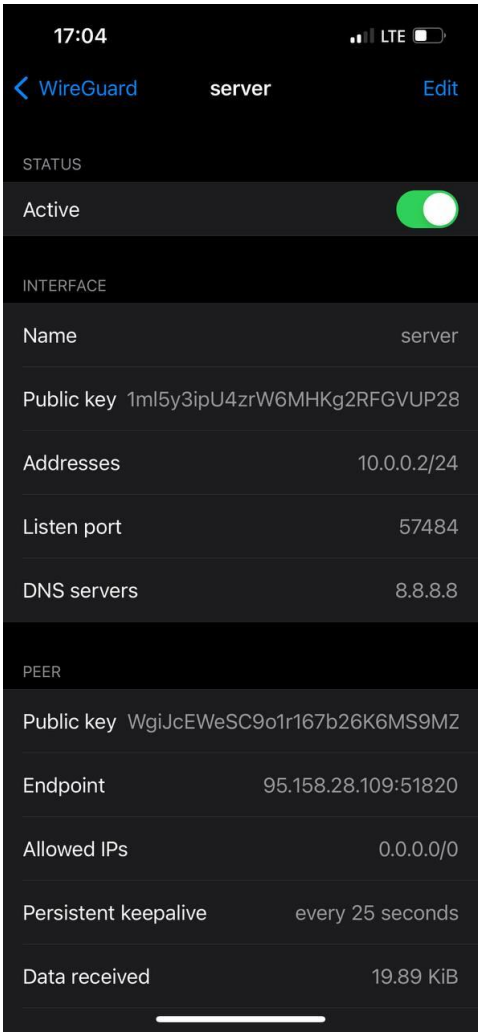

Рис. 3.32 Підключення до VPN на IOS клієнті

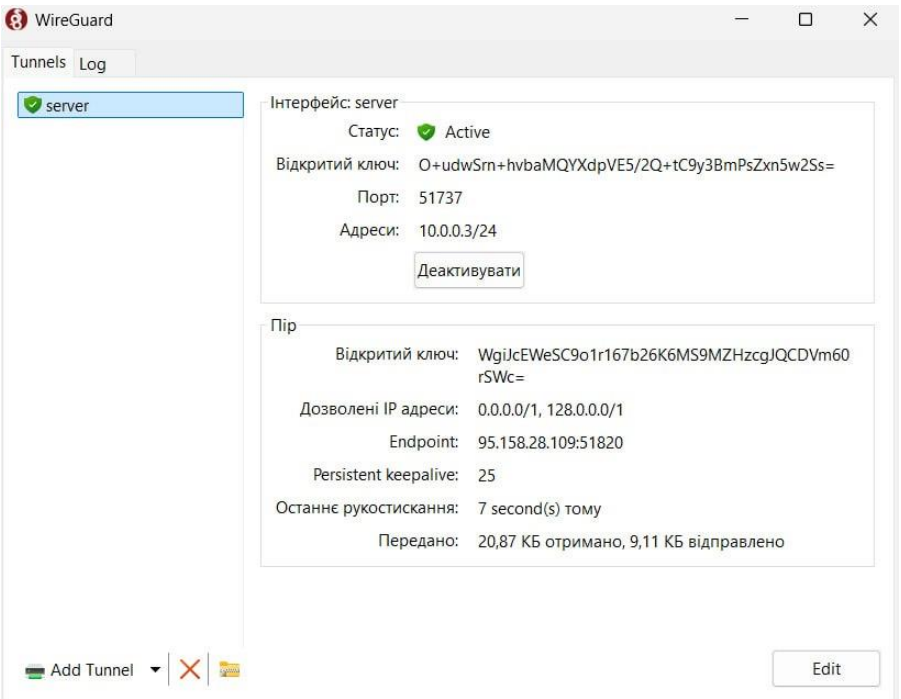

Рис. 3.33 Підключення до VPN на Windows клієнті

Для того щоб перевірити чи активні підключення на сервері, необхідно ввести команду **sudo wg show**, яка виведе інформацію про клієнтів. (Рис 3.34)

Як ми можемо побачити на рисунку 3.34 ми маємо 2 підключення **peer,** які ідентифікуються по публічним ключам ключам клієнтів. Також завдяки команді **sudo wg show** буде виведено справжню IP адресу клієнта (**endpoint**), IP адресу клієнта в мережі VPN(**allowed ips**), час коли було встановленно з'єднання (**latest handshake**), та об'єм трафіку який було отримано та надіслано до від клієнта до серверу (**transfer**). Вивід даної команди доводить що підключення активне і працює. Якщо ж станеться якась помилка і підключення буде не активне, то данна команда виведе лише ідентифікатор клієнта та його IP адресу в мережі VPN.

```
sai@stelnykserver:~$ sudo wg show
[sudo] password for sai:
interface: wg0
  public key: WgiJcEWeSC9olrl67b26K6MS9MZHzcgJQCDVm60rSWc=
  private key: (hidden)
  listening port: 51820

peer: lml5y3ipU4zrW6MHKg2RFGVUP28p7pB470yhLkJ8BEU=
  endpoint: 46.133.170.249:8082
  allowed ips: 10.0.0.2/32
  latest handshake: 47 seconds ago
  transfer: 139.28 KiB received, 362.84 KiB sent

peer: O+udwSrn+hvbaMQYXdpVE5/2Q+tC9y3BmPsZxn5w2Ss=
  endpoint: 46.133.170.249:8091
  allowed ips: 10.0.0.3/32
  latest handshake: 1 minute, 32 seconds ago
  transfer: 2.12 MiB received, 37.87 MiB sent
sai@stelnykserver:~$
```

Рис. 3.34 Інформація про підключених клієнтів на сервері

Також для перевірки підключення можна скористатися протоколом ICMP та з його допомогою виконати exo запит на клієнта застосувавши команду **ping** та вказавши IP адресу клієнта (Рис 3.35, Рис 3.36). Як ми можемо побачити на рисунках 3.35 та 3.36 ми отримуємо відповідь від клієнтів, а отже вони підключені. Важливо зазначити, що перевірка підключення за допомогою протокола ICMP з сервера на клієнта може видати що клієнт недоступний, хоча насправді підключення буде активне. Це відбувається в наслідок того що на клієнті може бути міжмережевий екран за замовчуванням, який буде відхиляти

ехо запити.

```
sai@stelnykserver:~$ ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=47.9 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=57.2 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=73.4 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=122 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3009ms
rtt min/avg/max/mdev = 47.919/75.198/122.302/28.681 ms
sai@stelnykserver:~$ ping 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
^C
--- 10.0.0.3 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4172ms

sai@stelnykserver:~$
```

Рис. 3.35 Відповідь від IOS клієнта

```
sai@stelnykserver:~$ ping 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=128 time=3.66 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=128 time=17.5 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=128 time=4.12 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=128 time=21.6 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=128 time=3.65 ms

--- 10.0.0.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4015ms
rtt min/avg/max/mdev = 3.652/10.115/21.647/7.834 ms
sai@stelnykserver:~$
```

Рис. 3.36 Відповідь від Windows клієнта

3.2.4. Перевірка та аналіз результатів налаштування VPN

В результаті налаштувань сервісу Wireguard, було організовано VPN тунель для того щоб мати віддалений доступ до серверу з будь-якої мережі, а не тільки локальної. Також VPN з’єднання дозволяє безпечно предавати файли на сервер, так як весь трафік шифрується VPN протоколом Wireguard.

Доступність файлового сховища на сервері через VPN перевірялася з IOS клієнта, підключеного до мобільної мережі, що вказано цифрою 1 Рис 3.37, а також бачимо IP адресу сервера в середині VPN, та символ замку, що вказує на те що використовується протокол HTTPS цифра 2 (Рис 3.37).

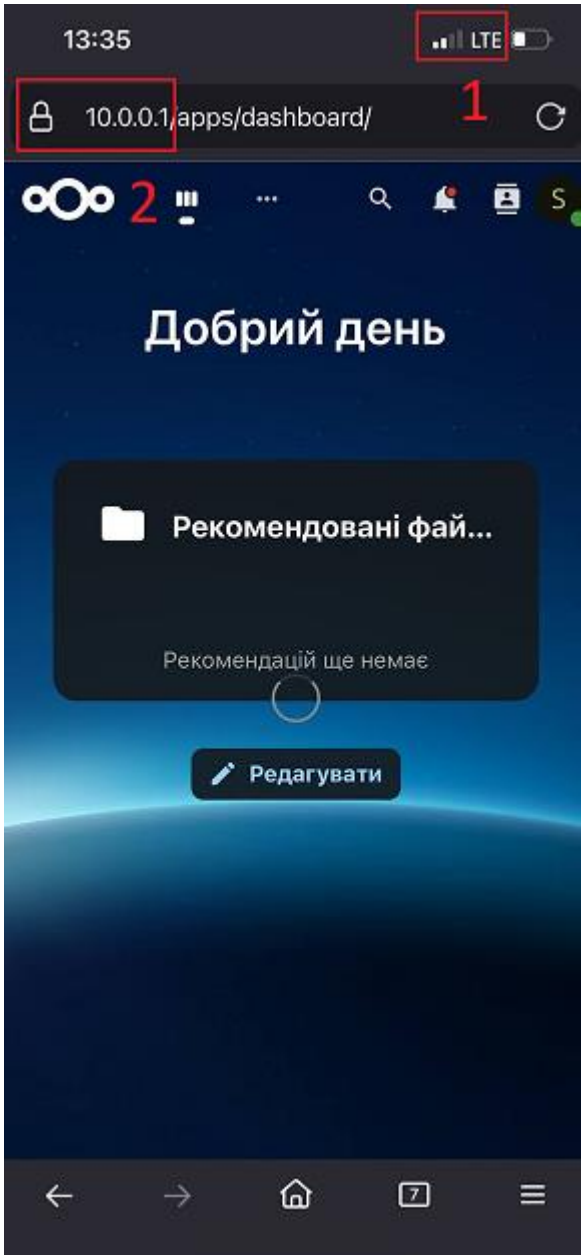

Рис. 3.37 Доступність файлового сховища через VPN тунель

Окрім того, було налаштовано VPN клієнта так, щоб весь трафік пропускався через сервер, таким чином перетворюючи наш сервер в VPN сервер, за рахунок чого, в разі потреби, ми можемо виходити в світ під домашньою IP адрессою сервера, приховуючи нашу справжню IP адресу клієнта та шифруючи весь трафік до самого серверу. За допомогою онлайн сервісу 2ip.ua [15], який показує яка наша поточна IP адреса, продемонструємо переадресацію нашої реальної IP адреси в IP адресу серверу. На рисунку 3.38 у який показує стан речей

до переадресації. Номером 1 помічено індикатор того, що ми підключені до мобільної мережі, номером 2 позначена наша поточна IP адреса, і номером 3 провайдер до якого ми підключені. На Рис 3.39 показано стан речей після переадресації, де, як і на попередньому малюнку номером 1 помічено індикатор того, що ми підключені до мобільної мережі, номером 2 позначена наша оновлена IP адреса, і номером 3 провайдер до якого ми підключені. Як ми можемо побачити не дивлячись не те, що ми досі підключені до мобільної мережі, наша IP адреса та провайдер змінилися. В свою чергу це означає що в глобальну мережу ми можемо виходити під новою адресою, маскуючи свою справжню.

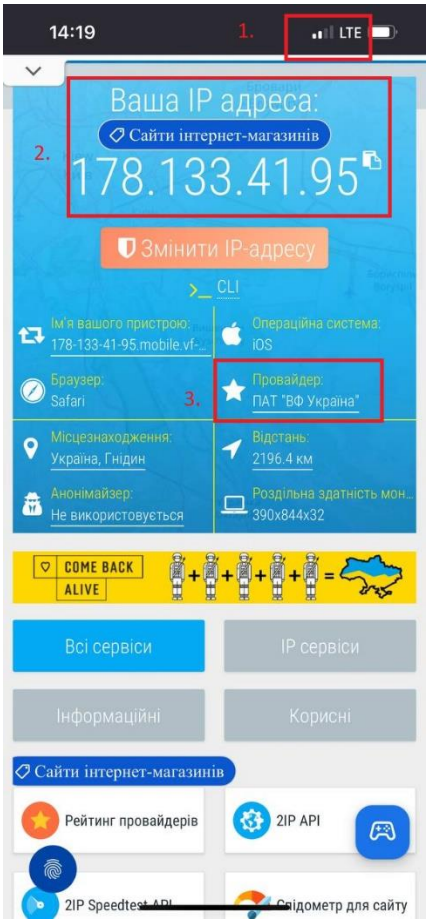

Рис 3.38 IP адреса на клієнті до переадресації

Рис 3.39 IP адреса на клієнті після переадресації

Висновки

1. У 3 розділі було реалізовано методику створення локальної серверної інфраструктури, відповідно до неї було реалізовано файловий сервер за допомогою сервісу Nextcloud, який було розгорнуто локально. Доступ до сервера було утворено за допомогою програмного забезпечення з відкритим вихідним кодом WireGuard, що створює VPN тунель до нашої локальної мережі та шифрує трафік. Також з його допомогою було реалізовано вихід в світ з білої IP адреси самого серверу, що робить наше рішення не тільки файловим сховищем з віддаленим доступом, а ще й VPN сервером, що унеможливлює перехоплення трафіку в публічних мережах, через шифрування тунелю.

2. Запропоноване рішення можна реалізувати для побутового використання із збереженням персональних файлів, з можливістю синхронізації та підключенням інших користувачів до сховища (наприклад, членів сім'ї для переглядання спільних альбомів). Також дане рішення надає функцію захисту трафіку в публічних мережах за рахунок реалізованого VPN підключення.

3. Розроблена інфраструктура може бути адаптована і для комерційного використання, забезпечуючи віддалений доступ для декількох співробітників об’єднуючи їх в одну мережу та надаючи доступ до спільного файлового сховища, що дозволяє оптимізувати бізнес процеси за рахунок мінімальних вкладень та не використовуючи дорого вартісні готові комерційні рішення, де ми не маємо безпосереднього контролю над нашими даними.

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

1. У ході виконання роботи було здійснено аналіз існуючих видів систем зберігання даних, зокрема DAS, NAS та SAN. На основі проведеного огляду, було обґрунтовано доцільність використання та реалізації NAS системи, а також наведені основні положення з реалізації NAS системи як гібридної яка поєднує в собі можливість отримати доступ до файлів як з LAN, так і з WAN.

2. Було спроектовано та реалізовано модель гібридної системи зберігання даних, що поєднує в собі локальну серверну інфраструктуру з можливістю віддаленого доступу через VPN тунель. Модель було реалізовано на базі віртуальної машини з операційною системою Ubuntu Server. За допомогою інструментів Docker було розгорнуто спеціальне програмне забезпечення Nextcloud як веб-орієнтоване середовище для організації файлообміну й зберігання та адміністрування файлів. Також було розгорнуто допоміжні сервіси, а саме MySQL, Redis та Nginx.

3. На базі WireGuard було організовано доступ до сервера через VPN тунелі, де тунель між сервером та клієнтом шифрується. Такий підхід дозволяє отримати безпечний канал зв’язку від користувача до сервера незалежно від розташування користувача чи провайдера до якого підключений користувач чи сервер.

4. Окрім шифрованого VPN тунелю, для захисту обробляємої та зберігаємої інформації було реалізовано підтримку протоколу HTTPS, що дозволяє уникнути передачі даних відкритим текстом при підключені до сереверу, як з локальної мережі так і з глобальної.

5. Проведене дослідження доводить, що для того щоб створити власну систему зберігання даних, з достатнім рівнем захищеності та доступності не обов’язково звертатися до комерційних або централізованих рішень і задача по створенню такої системи є досить реальною навіть за умови обмежених ресурсів. Такий підхід дозволяє не лише знизити залежність від великих компаній, але й налаштувати систему під себе та взяти під контроль свої дані.

6. На практиці таку систему можна застосувати як і при домашньому

користуванні так і на невеликих підприємствах, або ж як компонент більшої системи (SAN), у великих компаніях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ISO/IEC 2382:2015
   URL: https://www.iso.org/obp/ui/en/#iso:std:iso-iec:2382:ed-1:v2:en
2. Moore, G. E. (1965). "Cramming more components onto integrated circuits." Electronics Magazine, April 19, 1965.
3. URL: https://www.virtualbox.org/
4. URL: https://ubuntu.com/download/server#system-requirements-lts
5. URL: https://www.docker.com/company/
6. URL: https://www.docker.com/resources/what-container/
7. URL: https://nextcloud.com/about/
8. URL: https://redis.io/about/
9. URL: https://www.mysql.com/about/
10. URL: https://nginx.org/en/
11. URL: https://best.net.ua/pro-kompaniu/
12. Donenfeld, Jason A. "WireGuard: Next Generation Kernel Network Tunnel." NDSS. 2017.
13. URS: https://github.com/wireguard
14. URL: https://www.wireguard.com/
15. URL: https://2ip.ua/ua/
16. URL: https://www.netacad.com/cisco-packet-tracer
17. Poulton, Nigel. Data Storage Networking. N.p.: Sybex, 2014. – pp. 145 – 156.
18. Sacks, D. (2001). Demystifying storage networking das, san, nas, nas gateways, fibre channel, and iscsi. *IBM storage networking*, 3-11
19. Журавльов, П. В., & Медведський, А. М. (2016). Проблема вибору типу сховища даних підприємства. *Міжнародний науковий журнал*, (6 (1)), 25-27.
20. Shu, J. (2024). *Data Storage Architectures and Technologies*. Springer.
21. Грайворонський, М. В., & Новіков, О. М. (2009). Безпека інформаційно-комунікаційних систем.
22. Satyanarayanan, M., Howard, J. H., Nichols, D. A., Sidebotham, R. N.,

Spector, A. Z., & West, M. J. (1985). The ITC distributed file system: Principles and design. *ACM SIGOPS Operating Systems Review*, *19*(5), 35-50

23. URL: https://docs.docker.com/compose/

Додаток А

Файл docker-compose.yml

```
    version:'3.9'

services:
  db:
    image:mysql:8
    container_name:nextcloud-db
    restart:always
    command:--transaction-isolation=READ-COMMITTED -
binlog-format=ROW
    environment:
      MYSQL_ROOT_PASSWORD:XXXXXXXXX
      MYSQL_DATABASE:nextcloud
      MYSQL_USER:XXXX
      MYSQL_PASSWORD:XXXXXXXXXXX
    volumes:
      -db_data:/var/lib/mysql
    networks:
      -nextcloud_network

  redis:
    image:redis:latest
    container_name:nextcloud-redis
    restart:always
    networks:
      -nextcloud_network

  app:
    image:nextcloud:latest
```

```
    container_name:nextcloud-app
    restart:always
    expose:
      -"80"
    environment:
      NEXTCLOUD_ADMIN_USER:XXX
      NEXTCLOUD_ADMIN_PASSWORD:XXXXXXXXXX
      NEXTCLOUD_TRUSTED_DOMAINS:nextcloud.local
      MYSQL_HOST:db
      MYSQL_DATABASE:nextcloud
      MYSQL_USER:XXXXXX
      MYSQL_PASSWORD:XXXXXXXX
      REDIS_HOST:redis
    volumes:
      -nextcloud_data:/var/www/html
    networks:
      -nextcloud_network
    depends_on:
      -db
      -redis

  nginx:
    image:nginx:alpine
    container_name:nginx-ssl
    restart:always
    ports:
      -"80:80"
      -"443:443"
    depends_on:
      -app
```

```
    volumes:
      -
./nginx/default.conf:/etc/nginx/conf.d/default.conf
      -./certs:/etc/nginx/certs:ro
    networks:
      -nextcloud_network

volumes:
  db_data:
  nextcloud_data:
networks:
  nextcloud_network:
```

Додаток Б

Конфігураційний файл Nextcloud config.php

```
    <?php
$CONFIG=array(
  'htaccess.RewriteBase'=>'/',
  'memcache.local'=>'\\OC\\Memcache\\APCu',
  'apps_paths'=>
  array(
    0=>
    array(
      'path'=>'/var/www/html/apps',
      'url'=>'/apps',
      'writable'=>false,
    ),
    1=>
    array (
      'path'=>'/var/www/html/custom_apps',
      'url'=>'/custom_apps',
      'writable'=>true,
    ),
  ),
  'memcache.distributed'=>'\\OC\\Memcache\\Redis',
  'memcache.locking'=>'\\OC\\Memcache\\Redis',
  'redis'=>
  array(
    'host'=>'redis',
    'password'=>'',
    'port'=>6379,
  ),
  'upgrade.disable-web'=>true,
```

```
  'passwordsalt'=>'XXXXXXX',
  'secret'=>'XXXXXXXX',
  'trusted_domains'=>
  array(
    0=>'localhost',
    1=>'nextcloud.local',
    2=>'192.168.1.10',
    3=>'10.0.0.1'
  ),
  'overwrite.cli.url'=>'https://10.0.0.1',
  'overwriteprotocol'=>'https',
  'trusted_proxies'=>
  array(
    0=>'nginx-ssl'
  ),
  'datadirectory'=>'/var/www/html/data',
  'dbtype'=>'mysql',
  'version'=>'30.0.5.1',
  'dbname'=>'nextcloud',
  'dbhost'=>'db',
  'dbport'=>'',
  'dbtableprefix'=>'oc_',
  'mysql.utf8mb4'=>true,
  'dbuser'=>'XXXXXX',
  'dbpassword'=>'XXXXXXXX',
  'installed'=>true,
  'instanceid'=>'ocdtwtq3hggp',
);
```

Додаток В

Файл конфігурації веб серверу nginx, default.conf

```
server{
    listen 80;
    server_name 10.0.0.1;

    return301https://$host$request_uri;
}
server{
    listen443ssl;
    server_name10.0.0.1;

    ssl_certificate /etc/nginx/certs/selfsigned.crt;
    ssl_certificate_key /etc/nginx/certs/selfsigned.key;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-
AES128-GCM-SHA256';
    ssl_prefer_server_ciphers on;

    location / {
        proxy_pass http://app:80;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;
    }
    add_header Strict-Transport-Security "max-
age=31536000; includeSubDomains" always;
}
```