КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет автоматизації і інформаційних технологій

Кафедра інформаційних технологій

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТР

на тему:

Інформаційні технології для налаштування систем розумового приватного будинку

Мусієнко Володимир Вікторович
(прізвище, ім’я та по батькові здобувача повністю)

Київ 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет автоматизації і інформаційних технологій

Кафедра інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
_____
„____”_____2024 року

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТР

Інформаційні технології для налаштування систем розумового приватного будинку

Виконав: Мусієнко Володимир Вікторович
(прізвище, ім’я та по батькові повністю)
122 «Комп`ютерні науки»
(спеціальність)
«Комп`ютерні науки»
(освітня програма)
Групи: КНм-23
Керівник: Вацкель В.Ю.
(прізвище та ініціали)
асистент
(вчене звання, науковий ступінь)

Ідентичність підтверджую

Київ 2024 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: Автоматизації і інформаційних технологій

Кафедра: інформаційних технологій

Освітній рівень: «магістр» за ОП «Комп'ютерні науки»

Спеціальність: 122 «Комп'ютерні науки»

Спеціалізація: Інформаційні управляючі системи і технології

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
Гончаренко Т.А.
"___"_____2024 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Мусієнко Володимир Вікторович

1. Тема роботи: Інформаційні технології для налаштування систем розумового приватного будинку, затверджена наказом ректора КНУБА № 2213/2 від «8» жовтня 2024р.
2. Керівник роботи: асистент Вацкель В.Ю.
3. Строк подання студентом роботи до захисту: «___» листопада 2024 р.
4. Зміст пояснювальної записки за розділами:
   - Р. 1. Аналіз предметної області
   - Р. 2. Розгляд архітектури проекту і порівняння існуючих рішень
   - Р. 3. Опис технологій проекту системи розумний будинок
   - Р. 4. Установка і конфігурування програмно-апаратного стека.
   - Р. 5. Економічний аналіз витрат та обслуговування
5. Інформаційні слайди
   - С.1. Титульний слайд
   - С.2. Вступ
   - С.3. Актуальність теми
   - С.4. Мета атестаційної випускної роботи
   - С.5. Клієнт-серверна архітектура
   - С.6. Структурна схема клієнт-серверної архітектура
   - С.7. Основна ідея клієнт-серверної архітектури

С.8. Мікрокомп’ютер Raspberry Pi 3B+

С.9. Операційна система

С.10. Apache

С.11. PHP

С.12. MySQL

С.13. VSFTPD, phpMyAdmin

С.14. NextCloud

С.15. Фото діючого пристрою

С.16. Висновки

С.17. Вихідний слайд

6. Календарний план виконання кваліфікаційної випускної роботи

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз предметної області	Вересень 2024 р.
Р. 2 Розгляд архітектури проекту і порівняння існуючих рішень	Вересень 2024 р.
Р. 3. Опис технологій проекту системи розумний будинок	Жовтень 2024 р.
Р.4. Установка і конфігурування програмно-апаратного стека.	Жовтень 2024 р.
Р.5. Економічний аналіз витрат та обслуговування	Листопад 2024 р.
Остаточне оформлення роботи	Листопад 2024 р.
Направлення роботи на рецензування	Грудень 2024 р.
Попередній захист роботи на кафедрі	Грудень 2024 р.

7. Дата видачі завдання: «__»_____2024 р.

Заф.кафедри Гончаренко Т.А.

(підпис) (прізвище та ініціали)

Керівник Вацкель В.Ю.

(підпис) (прізвище та ініціали)

Студент Мусієнко В.В.

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Обсяг роботи 109 сторінок, 37 ілюстрацій, 4 таблиці, 38 джерел посилань

Метою випускної атестаційної роботи є надання розробленого рішення у вигляді сконфігурованого програмно-апаратного комплексу для використання в екосистемі розумного будинку

Випускна атестаційна робота надає прямі вказівки на реалізацію даного завдання, має вступну описову частину концепції IoT, аналіз клієнт-серверної архітектури, порівняння схожих рішень та покрокову інструкцію по налаштуванню сервера та мережевої хмари дані про апаратне та програмне забезпечення.

Актуальність і сучасність теми дипломного проекту обумовлена тим фактором, що на сьогоднішній день будь-кому потрібна своя захищена мережа даних. Результатом виконання роботи є налаштований сервер з локальною мережевою хмарою. Даний проект буде цікавим як для студентів, так і для практикуючих спеціалістів.

ANNOTATION

The work consists of 109 pages, 37 illustrations, 4 tables, 38 references

The purpose of the final certification work is to provide the developed solution in the form of a configured hardware and software complex for use in the smart home ecosystem

The graduation thesis provides direct instructions for the implementation of this task, has an introductory descriptive part of the IoT concept, an analysis of the client-server architecture, a comparison of similar solutions, and step-by-step instructions for setting up the server and network cloud, as well as hardware and software data.

The relevance and modernity of the topic of the diploma project is due to the fact that today everyone needs their own secure data network. The result of the work is a configured server with a local network cloud. This project will be of interest to both students and practitioners.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

HTTP (Hyper Text Transfer) - Протокол передачі даних, що використовується в комп'ютерних мережах;

PHP (Personal Home Page) – скриптова мова програмування;

VSFTPD (Very Secure FTP Daemon) – FTP сервер з підтримкою IPv6 та SSL;

IP (Internet Protocol address) – це ідентифікатор мережевого рівня;

URL (Uniform Resource Locator) – стандартизована адреса певного ресурсу;

ARM (Advanced RISC Machines) – архітектура процесора, яку розробила компанія ARM;

HDMI (High Definition Multimedia Interface) – інтерфейс та кабель для передачі цифрових відео та аудіо даних;

OS (Operating system) це базовий комплекс програм, що виконує управління апаратною складовою комп'ютера або віртуальної машини;

CSS (Cascading Style Sheets) – спеціальна мова, що використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітками даних;

HTML (Hypertext Markup Language) – стандартна мова розмітки для створення веб-сторінок і веб-додатків;

Perl (Practical Extraction) – високорівнева, інтерпретована, динамічна мова програмування загального призначення;

API (Application Programming Interface) – це набір готових класів, процедур, функцій, структур і констант, що надаються для використання в зовнишніх програмних продуктах;

СУБД (Система управління базами даних) – це комплекс програмних і мовних засобів, необхідних для створення баз даних, підтримання їх в актуальному стані та організації пошуку в них необхідної інформації;

Solaris – пропрієтарна комп'ютерна операційна система родини UNIX;

LAN (Local Area Network) – являє собою об'єднання певного числа комп'ютерів відносно невеликій території;

FTP (File Transfer Protocol) – дає можливість абоненту обмінюватися двійковими і текстовими файлами з будь-яким комп’ютером мережі, що підтримує протокол FTP;

TCP (Transmission Control Protocol) – разом із протоколом IP є стрижневимпротоколом інтернету, який дав назву моделі TCP/IP.

ВСТУП

У сучасну цифрову епоху компанії інвестують значні кошти в комплекси обладнання для забезпечення безперебійної роботи різних інтернет-ресурсів, таких як веб-сайти, веб-додатки та інтернет-сервіси. Однак обладнання часто включає надлишкові або рідко використовувані ресурси, що призводить до марної трати ресурсів і фінансів.

Окрім інтернету, поява інтернету речей (IoT) та технології "розумного дому" призвела до необхідності створення недорогих і доступних рішень, які могли б задовольнити потреби приватних осіб. Попит на пристрої Інтернету речей, такі як розумні датчики, дверні замки, термостати та інші пристрої для розумного дому, стрімко зростає. Однак розробка та розгортання цих пристроїв вимагає надійного та ефективного веб-сервера, який може впоратися з трафіком та обробкою даних.

Тому існує нагальна потреба у розробці та створенні недорогого веб-сервера, який може забезпечити повну функціональність та простоту використання для малих підприємств та організацій. Мікрокомп'ютер Raspberry Pi 3B+ пропонує ідеальне рішення для цієї проблеми. Це рішення може виступати в якості тестової платформи для IoT-пристроїв або повнофункціонального веб-сервера для розумного будинку.

Таким чином, метою даної атестаційної випускної роботи є надання доступного рішення у вигляді конфігурованого програмно-апаратного комплексу на базі мікрокомп'ютера Raspberry Pi 3B+. Це рішення спрямоване на задоволення потреб приватних осіб які конфігурують свій розумний будинок шляхом надання надійного та ефективного веб-сервера для їх інтернет-ресурсів та пристроїв Інтернету речей. Завдяки цьому проекту вони можуть зменшити свої витрати, але при цьому мати можливість користуватись надійним рішення.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка задачі та актуальність

Тема дослідження:
"Технології систем розумного приватного будинку".

Актуальність теми

Сучасний світ стрімко рухається в бік цифровізації, а технології Інтернету речей (IoT) займають ключове місце в цьому процесі. Концепція розумного будинку (Smart Home) об'єднує в єдину систему автоматизоване управління енергетичними ресурсами, безпекою, комунікаціями та побутовими пристроями, сприяючи оптимізації повсякденного життя. У контексті глобальної екологічної кризи та зростаючої урбанізації такі рішення стають не просто зручністю, а необхідністю.

Однак сучасні комерційні системи часто є надмірно дорогими та складними для впровадження в приватних будинках середнього класу. Це створює попит на доступні, надійні та легко інтегровані рішення. Інтеграція IoT-технологій на основі мікрокомп’ютерів, таких як Raspberry Pi, надає можливість створення економічно ефективних рішень, які відповідають сучасним стандартам автоматизації.

Мета дослідження

Розробити універсальний програмно-апаратний комплекс для розумного будинку, що базується на мікрокомп’ютері Raspberry Pi, з інтеграцією IoT-пристроїв, хмарних технологій та клієнт-серверної архітектури для забезпечення ефективного управління та зберігання даних.

Завдання дослідження

1. Теоретичне обґрунтування:

– Провести систематичний огляд існуючих технологій у сфері розумного будинку, включаючи IoT-платформи, протоколи передачі даних, мікроконтролери, операційні системи, програмні стеки та хмарні сервіси.

– Визначити архітектурні принципи побудови клієнт-серверних систем для IoT-рішень.

– Дослідити проблеми конфіденційності, безпеки та енергоефективності в IoT-середовищі.

2. Проектування системи:

– Розробити архітектуру системи розумного будинку, що включає:

– Розробити програмні модулі для інтеграції IoT-пристроїв за допомогою протоколів передачі даних (MQTT, HTTP, FTP тощо).

3. Реалізація програмно-апаратного комплексу:

– Виконати вибір та налаштування операційної системи для мікрокомп'ютера Raspberry Pi.

– Використати клієнтське програмне забезпечення для управління системою з мобільних пристроїв та ПК.

4. Тестування та верифікація:

– Провести модульне тестування компонентів системи, зокрема веб-сервера, IoT-пристроїв та хмарного сховища.

– Оцінити ефективність роботи системи в умовах тестового сценарію реального використання.

5. Економічна та енергетична оцінка:

– Оцінити витрати на впровадження системи.

– Провести аналіз енергоспоживання мікрокомп'ютера та IoT-пристроїв у системі.

Наукова новизна

У роботі пропонується інноваційний підхід до розробки розумного будинку на основі мікрокомп'ютера Raspberry Pi. Замість традиційних комерційних рішень, акцент зроблено на використання відкритих програмних рішень, що забезпечують масштабованість і адаптивність до потреб користувача. Досліджено способи мінімізації витрат і енергоспоживання системи, а також підвищення її безпеки та надійності.

Практична значущість

Результати дослідження мають широку сферу застосування:

1. Для кінцевих користувачів: Надання доступного рішення для автоматизації приватного житла.

2. Для малого бізнесу: Можливість комерційного впровадження системи як частини послуг з автоматизації будинків.

3. Для освіти: Використання системи як платформи для навчання основ IoT, мережевих технологій та розробки програмно-апаратних комплексів.

4. Для досліджень: База для подальшого розвитку в області IoT, автоматизації та енергозбереження.

Очікувані результати:

1. Реалізація прототипу системи розумного будинку, що включає веб-сервер, хмарне сховище та IoT-пристрої.

2. Документована інструкція з розгортання та налаштування програмно-апаратного комплексу.

Експериментальні результати, що підтверджують ефективність та функціональність системи в умовах реального використання.

1.2 IoT як спосіб взаємодії між пристроями

Інтернет речей (IoT) визначається як екосистема підключених пристроїв, які обмінюються даними через бездротову або дротову мережу. Ці пристрої можуть бути різного типу, включаючи смартфони, ноутбуки, розумні побутові прилади, офісне обладнання та будь-які пристрої, оснащені датчиками. Зібрані дані передаються на сервери, які можуть бути розташовані як у хмарі, так і локально. На цих серверах дані обробляються для отримання інформації, що допомагає в прийнятті рішень. IoT може існувати не лише на рівні окремих будинків або офісів, але й у більших масштабах, таких як університетські кампуси, міста чи навіть промислові комплекси.

Інтернет речей відкриває широкі можливості для інтеграції фізичних пристроїв у цифрове середовище. Завдяки цьому стало можливим створення

більш автоматизованих і ефективних рішень у різних сферах, включаючи охорону здоров'я, сільське господарство, транспорт і розумні будинки. Розумні пристрої взаємодіють між собою, стаючи невід'ємною частиною повсякденного життя.

На рівні окремих користувачів це дає змогу створювати інтелектуальні системи для управління домом, такі як розумні термостати, охоронні системи або автоматичне регулювання освітлення. У масштабах бізнесу IoT забезпечує контроль виробничих процесів, збирання аналітичних даних і управління логістикою.

Важливою особливістю IoT є його здатність працювати в реальному часі, надаючи миттєвий доступ до інформації. Ця властивість робить технологію цінною для застосування в критично важливих системах, наприклад, в управлінні інфраструктурою міст (Smart Cities), контролі за здоров'ям пацієнтів або моніторингу екологічного стану довкілля.

Для досягнення таких можливостей IoT вимагає надійних мережевих з’єднань і різноманітних протоколів обміну даними, таких як MQTT, CoAP, HTTP, FTP та інші. Кожен протокол адаптований під специфічні вимоги – від передачі невеликих обсягів даних до забезпечення максимальної швидкості або надійності передачі. Водночас безпека даних у IoT є критичним питанням, оскільки велика кількість підключених пристроїв створює потенційні загрози для конфіденційності.

Інтернет речей продовжує розвиватися, інтегруючи нові технології, такі як 5G, штучний інтелект та машинне навчання. Це дозволяє створювати ще більш потужні системи, здатні до автономного функціонування, адаптації до умов навколишнього середовища та самонавчання. Очікується, що IoT матиме ще більше застосувань у майбутньому, пов'язаних із побудовою "розумних" інфраструктур, підвищенням ефективності виробництва та поліпшенням якості життя людей.

Інтернет речей забезпечує фундамент для створення нового рівня взаємодії між людиною, технологіями та навколишнім світом. Його роль у повсякденному

житті та промислових масштабах буде тільки зростати, формуючи нові можливості для інновацій і розвитку.[1]

1.3 Історія IoT

Історія Інтернету речей (IoT) починається з ARPANET, першої підключеної мережі, яка стала прародиною сучасного Інтернету. Пройдемося по важливих етапах еволюції IoT. У 1982 році студент Університету Карнегі-Меллона Девід Ніколс спробував зрозуміти, чи є в автоматі з продажу кока-коли холодна газована вода. За допомогою коду, розробленого разом зі своїми колегами, він зміг відстежувати наявність та стан пляшок в автоматі через мережу ARPANET. У 1989 році Тім Бернерс-Лі, пропонуючи структуру всесвітньої мережі, заклав основи Інтернету, що стало фундаментом для подальшого розвитку IoT. У 1990 році Джон Ромкі створив тостер, який можна було управляти через Інтернет. Це був перший пристрій IoT, який підключався до комп'ютера через кабель, оскільки тоді ще не було бездротового зв'язку Wi-Fi. У 1993 році в Кембриджському університеті Квентін Стаффорд-Фрейзер і Пол Джардецький створили "Trojan Room Coffee Pot" - кавоварку, зображення з якої транслювалося онлайн через мережу.

У 1999 році Кевін Ештон ввів термін "Інтернет речей" і розповів про зв'язок RFID з Інтернетом у своїй презентації в компанії Procter and Gamble. У 2003-2004 роках термін IoT почав активно використовуватися в основних виданнях, таких як The Guardian і Scientific American. Також в цей період RFID технологія стала використовуватися Міністерством оборони США та Walmart. В 2005 році Міжнародний союз телекомунікацій ООН визнав вплив IoT у своїй доповіді, передбачивши створення нової динамічної мережі мереж. У 2008 році в Цюріху відбулася перша конференція з присвячена IoT, що сприяла обміну знаннями між дослідниками та практиками. Того ж року Рада національної розвідки США включила IoT до шести руйнівних цивільних технологій. У 2011 році компанія Cisco випустила офіційний документ, в якому стверджувалося, що IoT народився між 2008 і 2009 роками, коли кількість підключених до Інтернету речей

перевищила кількість людей, що користуються Інтернетом. У 2011 році IoT було включено до циклу Hype для нових технологій, що були на підйомі, за даними компанії Gartner. У 2013 році IDC прогнозувала, що ринок IoT зросте на 7,9% і досягне 8,9 трильйона доларів США до 2020 року. Ці важливі моменти в історії IoT свідчать про поступовий розвиток технологій і підключення різних пристроїв до Інтернету, що відкриває безліч можливостей для сполучення фізичного та цифрового світу.[2]

1.4 Архітектура IoT

Архітектура Інтернету речей (IoT) включає три основних рівні: фізичний рівень, рівень периферійних обчислень і прикладний рівень. Ця багаторівнева модель забезпечує ефективний обмін даними, обробку інформації та інтеграцію пристроїв у єдину систему.

1. Фізичний рівень складається з пристроїв, датчиків та контролерів, які генерують дані. Ці пристрої можуть бути різного типу, включаючи смартфони, планшети, ноутбуки, розумні побутові прилади, пристрої з мікрочіпами або RFID-мітками. Цей рівень охоплює всі апаратні компоненти IoT, які здійснюють збір даних, зокрема температурні датчики, камери відеоспостереження, датчики руху, GPS-модулі тощо.

2. Рівень периферійних обчислень забезпечує зберігання та обробку даних неподалік від пристроїв, де вони генеруються. Це включає мережі та протоколи зв'язку, які забезпечують підключення пристроїв, а також можливості периферійних обчислень. Збір, первинна обробка та відправлення даних на сервери або у хмару відбуваються саме на цьому рівні. В IoT це часто досягається за допомогою шлюзів, які виступають посередниками між фізичними пристроями та центральними обчислювальними ресурсами.

3. Прикладний рівень знаходиться за межами рівня обчислень і використовує хмарні сервіси. На цьому рівні надаються інтегровані послуги IoT, які дозволяють отримувати корисну інформацію та аналізувати зібрані дані. Зібрані дані очищуються, зберігаються на хмарних серверах і підлягають

подальшій обробці для створення звітів, статистичних даних або прийняття рішень. Цей рівень забезпечує інтеграцію IoT із зовнішніми системами, такими як мобільні додатки, веб-інтерфейси або API для сторонніх сервісів.

Для підключення різних типів пристроїв до мережі IoT використовуються різноманітні протоколи. Деякі з них включають ZigBee, Z-wave, Wi-Fi, Li-Fi, NFC, 5G, BLE, IETF 6LoWPAN, IETF's CoAP тощо. Ці протоколи забезпечують передачу даних між пристроями та системами IoT. Наприклад, протоколи MQTT та HTTP використовуються для передачі даних між датчиками і хмарними сервісами, а ZigBee та Z-wave ефективні для створення локальних мереж розумного будинку.

У перспективі архітектура IoT все більше тяжіє до декомпозиції рівнів, що дозволяє знижувати навантаження на центральні сервери. Така тенденція обумовлена появою технологій "edge computing" (периферійні обчислення) і "fog computing" (обчислення у тумані). Вони дають змогу обробляти частину даних на пристроях або найближчих шлюзах, що зменшує затримки і підвищує ефективність використання мережевих ресурсів.

Додатковою перевагою цієї багаторівневої архітектури є можливість забезпечення надійності та безпеки даних. Наприклад, шифрування даних на рівні пристроїв мінімізує ризики перехоплення під час їх передачі, а централізоване управління доступом на прикладному рівні сприяє запобіганню несанкціонованого втручання.

Інтернет речей є однією з найбільш економічно перспективних технологій. За даними McKinsey, до 2025 року економічна цінність IoT може сягати від 3,9 трильйона до 11,2 трильйона доларів США. Це обумовлено широким спектром застосувань, що включають виробництво, охорону здоров'я, громадську безпеку, енергоменеджмент, транспорт і багато інших сфер.

Таким чином, багаторівнева архітектура IoT забезпечує фундамент для створення інноваційних рішень у різних галузях, дозволяючи підключати та інтегрувати пристрої, обробляти дані та надавати корисні сервіси кінцевим користувачам.[3]

1.5 Проблеми IoT

1. Незважаючи на те, що Інтернет речей (IoT) обіцяє змінити спосіб роботи бізнесу, впровадження цієї технології стикається з рядом проблем, які потребують вирішення.

2. Апаратні платформи

Протягом останніх років апаратні платформи IoT значно покращилися, забезпечуючи широкий спектр застосувань. Однак усе ще існують технічні обмеження, зокрема низька енергоефективність, обмежені можливості зберігання даних і недостатня обчислювальна потужність для виконання складних задач. Наприклад, малі датчики часто потребують мініатюрних батарей із тривалим терміном служби, але сучасні технології не завжди дозволяють цього досягти.

3. Планування даних

IoT генерує величезні обсяги даних, що вимагає ретельного стратегічного підходу до їх збирання, зберігання, обробки та аналізу. Неконтрольований потік даних може перевантажити систему та призвести до втрати важливої інформації. Вирішення цієї проблеми включає оптимізацію алгоритмів обробки та зберігання, а також використання технологій "edge computing" для первинного аналізу даних на місці їх збору.

4. Конфіденційність і безпека

IoT-системи часто обробляють чутливу інформацію, включаючи персональні дані користувачів і корпоративну інформацію. Це створює ризик несанкціонованого доступу, кібератак і витоку даних. Для вирішення цієї проблеми необхідно впроваджувати багаторівневу систему безпеки, яка включає шифрування даних, автентифікацію пристроїв, регулярні оновлення програмного забезпечення та моніторинг мережі для виявлення загроз.

5. Стабільність і енергоспоживання

Пристрої IoT часто працюють у режимі 24/7, що вимагає значних енергетичних витрат. Це особливо актуально для систем із великою кількістю підключених

пристроїв, таких як розумні міста чи промислові підприємства. Вирішення цієї проблеми включає оптимізацію алгоритмів обробки даних, використання енергоефективних компонентів і впровадження технологій відновлюваної енергії.

6. Сумісність і стандартизація

На ринку існує багато різних платформ і протоколів IoT, що створює складнощі у взаємодії між пристроями різних виробників. Відсутність єдиних стандартів може призвести до фрагментації ринку та обмеження можливостей інтеграції. Рішенням цієї проблеми є створення відкритих стандартів, які дозволять забезпечити сумісність між різними системами.

7. Масштабованість

У міру зростання кількості пристроїв у мережі IoT збільшуються вимоги до інфраструктури, зокрема до мережевого обладнання, серверів і програмного забезпечення. Це може викликати затримки в обробці даних, зниження продуктивності системи та зростання витрат. Для подолання цієї проблеми необхідно впроваджувати хмарні обчислення та технології децентралізованого управління даними.

8. Правові та етичні питання

IoT-системи часто стикаються з труднощами у регулюванні використання даних, дотриманні прав на конфіденційність і відповідності локальним законам. Наприклад, у Європейському Союзі використання IoT має відповідати нормам GDPR, що додає складнощів до проєктування систем. Крім того, виникають етичні питання, пов'язані з використанням даних для автоматизованого прийняття рішень.

9. Екологічний вплив

Масштабне впровадження IoT підвищує попит на електронні компоненти, що може спричинити зростання кількості електронних відходів і вплив на навколишнє середовище. Необхідно розробляти більш екологічно чисті рішення, такі як переробка старих пристроїв, використання біорозкладних матеріалів і впровадження енергоефективних технологій.

10. Таким чином, розвиток IoT супроводжується численними викликами, які вимагають комплексного підходу до їх вирішення. Впровадження інноваційних рішень, розробка нових стандартів і вдосконалення апаратних та програмних компонентів сприятимуть подальшому прогресу цієї технології.[4]

1.6 Технології, які сприяють розвитку IoT

1. Розвиток Інтернету речей (IoT) багато в чому залежить від впровадження інноваційних технологій, які забезпечують його функціональність, ефективність і зручність у використанні. Основними технологіями, що сприяють розвитку IoT, є:

2. Хмарні обчислення

Хмарні технології забезпечують доступ до потужних обчислювальних ресурсів і необмеженого сховища даних. Це дозволяє системам IoT ефективно зберігати, аналізувати та передавати великі обсяги даних у реальному часі. Використання хмарних обчислень також знижує витрати на інфраструктуру, оскільки організації можуть орендувати необхідні ресурси замість того, щоб інвестувати у власні сервери.

3. Великі дані (Big Data)

IoT генерує величезну кількість даних, які потребують обробки для отримання корисної інформації. Технології Big Data забезпечують збирання, структурування, аналіз і візуалізацію цих даних, що дозволяє виявляти тренди, прогнозувати події та приймати обґрунтовані рішення. Завдяки аналітиці великих даних IoT-системи стають більш адаптивними та ефективними.

4. Штучний інтелект (AI) і машинне навчання (ML)

Штучний інтелект і алгоритми машинного навчання відіграють ключову роль у підвищенні автономності IoT-пристроїв. Завдяки цим технологіям пристрої можуть адаптуватися до змін у навколишньому середовищі, навчатися на основі зібраних даних і покращувати свою роботу. Наприклад, AI може використовуватися в розумних будинках для оптимізації використання

енергоресурсів або в системах безпеки для виявлення загроз у режимі реального часу.

5. 5G і інші технології зв’язку

Впровадження мереж п’ятого покоління забезпечує значно вищу швидкість передачі даних, знижені затримки та більшу пропускну здатність. Це дозволяє підключати до мережі більше пристроїв без зниження їхньої продуктивності. Окрім 5G, розвиток технологій зв’язку, таких як LoRaWAN, NB-IoT, ZigBee, Bluetooth Low Energy (BLE), також сприяє підключенню пристроїв у різних умовах, включаючи віддалені або важкодоступні місця.

6. Розширена реальність (AR) та віртуальна реальність (VR)

Інтеграція AR і VR у IoT дозволяє створювати інноваційні рішення для візуалізації даних, навчання персоналу та управління складними системами. Наприклад, у промислових IoT-системах використання AR може полегшити діагностику обладнання або забезпечити покрокові інструкції з ремонту.

7. Технології периферійних обчислень (Edge Computing)

Периферійні обчислення дозволяють виконувати обробку даних безпосередньо на пристроях або найближчих до них вузлах мережі, що зменшує затримки та навантаження на хмарні ресурси. Це особливо важливо для IoT-рішень, які вимагають роботи в реальному часі, наприклад, у системах автономного транспорту або медичного моніторингу.

8. Блокчейн

Блокчейн-технології забезпечують прозорість, безпеку та незмінність даних, що передаються між IoT-пристроями. Це особливо корисно в системах із високими вимогами до довіри, наприклад, у фінансових транзакціях, логістичних ланцюжках або моніторингу походження товарів.

9. Енергозберігаючі технології

Розробка енергоефективних компонентів, таких як мікропроцесори з низьким енергоспоживанням або відновлювані джерела енергії (сонячні батареї, енергія тепла або вітру), сприяє зменшенню витрат на підтримку IoT-пристроїв, особливо у віддалених місцях.

10. Кібербезпека

Безпека даних є однією з ключових умов успішного впровадження IoT. Технології кібербезпеки, такі як шифрування, автентифікація, контроль доступу та засоби виявлення загроз, забезпечують захист IoT-систем від несанкціонованого доступу та кібератак.

11. Модульність і стандартизація

Модульний підхід до розробки IoT-пристроїв забезпечує їхню гнучкість і адаптованість до різних потреб. Стандартизація протоколів і платформ сприяє сумісності пристроїв різних виробників, що спрощує інтеграцію IoT-рішень у єдину екосистему.

12. Таким чином, розвиток IoT безпосередньо залежить від вдосконалення супутніх технологій, які забезпечують його масштабованість, ефективність і надійність. Інновації в цих сферах дозволяють IoT системам ставати невід'ємною частиною сучасного життя та забезпечують їхню інтеграцію у все більше сфер людської діяльності.[5]

1.7 Урядові постанови по регулюванню IoT

Державне регулювання Інтернету речей відрізняється в різних країнах, оскільки кожна країна має свій власний підхід до вирішення проблем і можливостей, що надаються технологіями Інтернету речей. Нижче наведено загальний огляд державного регулювання Інтернету речей у різних країнах світу.

Сполучені Штати: У США регулювання Інтернету речей в першу чергу зосереджене на конфіденційності та безпеці. Федеральна торгова комісія (FTC) видала рекомендації для виробників пристроїв Інтернету речей, закликаючи їх надавати пріоритет безпеці, забезпечувати чітку і прозору політику конфіденційності та отримувати згоду користувачів на збір даних. Крім того, кілька штатів прийняли власні закони, що стосуються безпеки Інтернету речей та захисту даних.[6]

Європейський Союз: ЄС застосував комплексний підхід до регулювання Інтернету речей за допомогою Загального регламенту про захист даних (GDPR).

GDPR регулює збір, зберігання та обробку персональних даних, включаючи дані, зібрані пристроями Інтернету речей. Він наголошує на згоді користувача, захисті даних та праві бути забутим. ЄС також має спеціальні правила для таких секторів, як електронна охорона здоров'я та "розумний" транспорт.[7]

Китай: Китай активно розробляє нормативно-правові акти у сфері Інтернету речей, щоб сприяти його впровадженню, забезпечуючи при цьому безпеку даних. Закон про кібербезпеку Китаю вимагає від операторів Інтернету речей дотримуватися певних стандартів безпеки та захищати дані користувачів. Уряд Китаю також запровадив програми сертифікації пристроїв та виробників Інтернету речей, щоб забезпечити дотримання правил безпеки.[8]

Південна Корея: Південна Корея прийняла Закон про сприяння розвитку Інтернету речей, який має на меті сприяти зростанню технологій та галузей Інтернету речей. Закон містить положення щодо захисту приватності, безпеки даних та галузевих стандартів. Він також сприяє співпраці між державним і приватним секторами в розвитку IoT.[9]

Японія: Японія прийняла Закон про сприяння розвитку Інтернету речей, який спрямований на створення сприятливого середовища для інновацій у сфері Інтернету речей. Він заохочує розробку галузевих стандартів, заходів безпеки даних та захисту приватності. Уряд також заохочує державно-приватне партнерство для просування ініціатив у сфері Інтернету речей.[10]

Індія: Уряд Індії випустив Національну політику цифрових комунікацій, в якій розглядаються різні аспекти Інтернету речей, включаючи захист даних, конфіденційність і безпеку. Вона наголошує на розвитку інфраструктури Інтернету речей, інновацій та нормативно-правової бази для підтримки розгортання Інтернету речей.[11]

Австралія: Австралія прийняла Закон про безпеку критичної інфраструктури, який охоплює системи Інтернету речей в таких важливих секторах, як енергетика, водопостачання і транспорт. Закон вимагає від організацій забезпечувати безпеку та стійкість своїх систем Інтернету речей та повідомляти про будь-які значні інциденти кібербезпеки.[12]

1.8 Майбутнє IoT

За прогнозами Statista, до 2025 року кількість підключених пристроїв у світі сягне 75,4 мільярда. Це продемонстровано на рисунку 1.1 Це демонструє, що майбутнє IoT буде не лише інноваційним, але й революційним. Очікується, що нові тенденції у сфері IoT значно впливатимуть на розвиток технологій та їх інтеграцію у різні галузі.

Однією з ключових тенденцій стане впровадження технології 5G, яка значно збільшить швидкість передачі даних, знизить затримки та дозволить підключати більшу кількість пристроїв до мережі. Це стане основою для розвитку високонавантажених IoT-додатків, таких як автономні транспортні засоби, розумні міста та індустрія 4.0.

Ще однією важливою зміною буде інтеграція IoT з штучним інтелектом (AI) та машинним навчанням. Ці технології дозволять пристроям аналізувати великі обсяги даних, робити прогнози та приймати рішення в реальному часі. Наприклад, у медицині це дозволить створити інтелектуальні системи для моніторингу здоров'я пацієнтів та попередження про можливі загрози.

Інтернет речей також стає ключовим компонентом у створенні розумних міст (Smart Cities). IoT використовується для оптимізації управління транспортом, моніторингу якості повітря, управління енергоресурсами та забезпечення безпеки. У майбутньому очікується збільшення рівня автоматизації міських інфраструктур завдяки впровадженню IoT-рішень.

Промисловий IoT (Industrial IoT) стане невід'ємною частиною індустрії 4.0, забезпечуючи автоматизацію та оптимізацію виробничих процесів. IoT дозволить компаніям створювати "розумні" фабрики, які здатні самостійно виявляти проблеми, регулювати витрати ресурсів і прогнозувати необхідність технічного обслуговування обладнання.

У сільському господарстві IoT сприятиме розвитку точного землеробства, що дозволить оптимізувати використання води, добрив і пестицидів,

забезпечуючи при цьому підвищення врожайності та зменшення негативного впливу на довкілля.

Окрему увагу привертає сфера кібербезпеки IoT. Зростання кількості підключених пристроїв збільшує ризики кібератак. Тому одним із головних викликів майбутнього буде розробка більш надійних стандартів безпеки, протоколів шифрування та засобів виявлення загроз.

Крім того, очікується, що IoT сприятиме розвитку економіки замкнутого циклу, дозволяючи створювати ефективні системи моніторингу та управління відходами, повторного використання ресурсів і зменшення вуглецевого сліду.

Ще однією перспективною сферою є інтеграція IoT з блокчейном, що дозволить забезпечити прозорість і незмінність даних у мережах IoT. Це може бути використано у ланцюжках поставок, де блокчейн та IoT гарантуватимуть відстеження походження товарів і їхнього стану на кожному етапі.

Таким чином, майбутнє IoT характеризується стрімким розвитком технологій, їх інтеграцією у різні сфери діяльності людини та підвищенням рівня автоматизації. Ця еволюція має значний потенціал для створення нових можливостей, поліпшення якості життя та вирішення глобальних проблем, таких як урбанізація, екологічні виклики та енергетична ефективність.

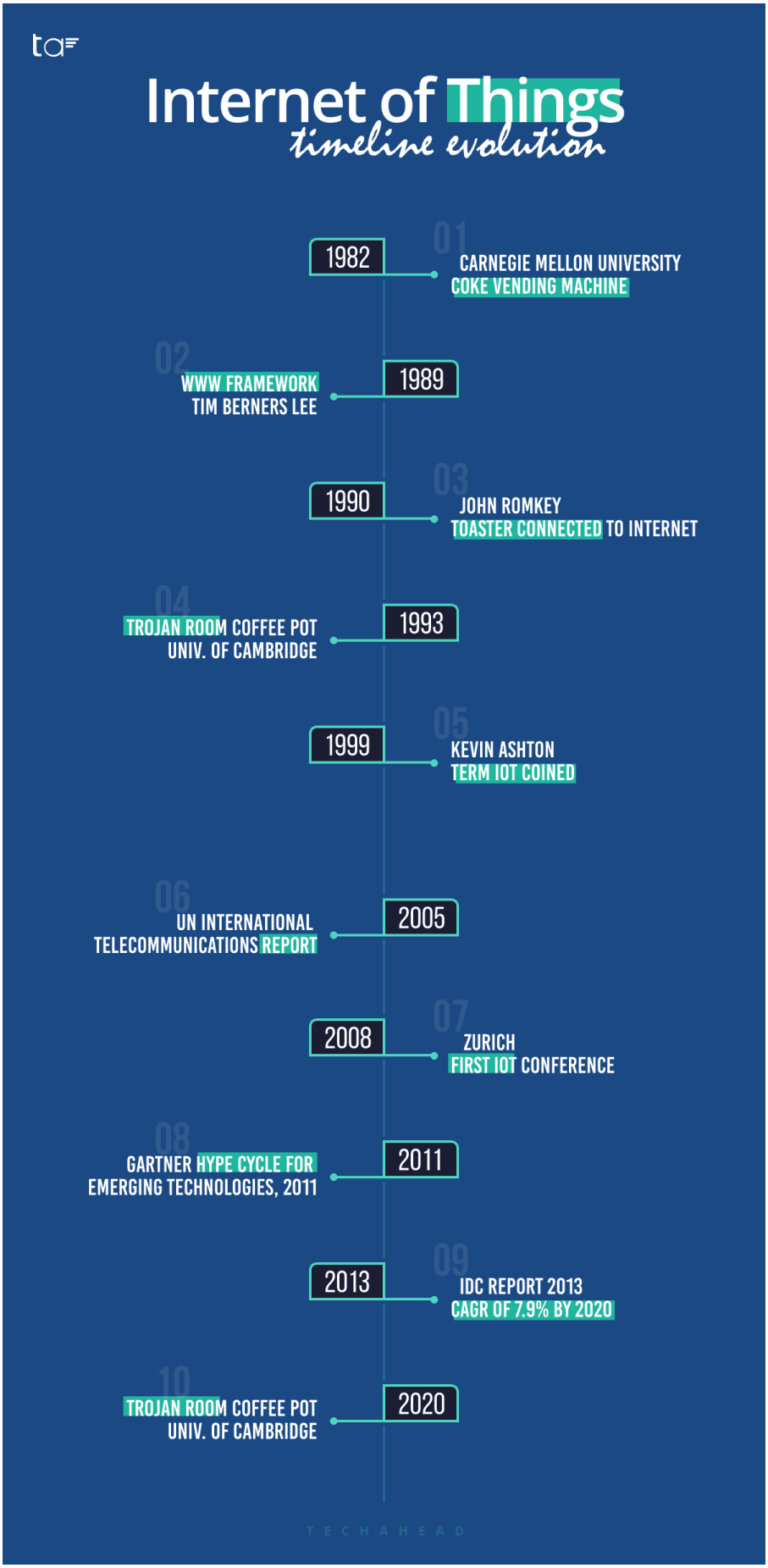


Рисунок 1.1 — Еволюція IoT

РОЗДІЛ 2 РОЗГЛЯД АРХІТЕКТУРИ ПРОЕКТУ І ПОРІВНЯННЯ ІСНУЮЧИХ РІШЕНЬ

2.1 Клієнт-серверна архітектура та ролі серверів

Клієнт-серверна архітектура є однією з найпоширеніших моделей взаємодії в інформаційних системах. Вона забезпечує централізовану обробку даних і управління сервісами, що робить її ключовим компонентом сучасних IoT-рішень, веб-додатків і корпоративних систем.

Клієнт-серверну архітектуру можна описати як систему, у якій клієнтські пристрої надсилають запити до серверів, а сервери обробляють ці запити та повертають результати. У контексті IoT вона є особливо важливою для організації роботи пристроїв, які взаємодіють з центральним сервером для обміну даними.

Основні компоненти клієнт-серверної архітектури

1. Клієнт

Це програма або пристрій, який ініціює запит до сервера. Клієнти можуть бути як апаратними пристроями (сенсори, контролери), так і програмними інтерфейсами (веб-додатки, мобільні програми).

2. Сервер

Це центральний елемент, який обробляє запити клієнтів, зберігає та надає дані, виконує бізнес-логіку або управляє пристроями. Сервери забезпечують обробку великих обсягів даних і можуть функціонувати в різних ролях залежно від завдань.

3. Мережа

Забезпечує з’єднання між клієнтами та серверами. Мережа може бути локальною (LAN) або глобальною (WAN/Інтернет), а для IoT часто використовуються протоколи зв’язку, такі як MQTT, HTTP або CoAP.

Клієнт-серверна архітектура набула своєї популярності завдяки динамічному розвитку мережі Інтернет та зосередження значної частини інформації в базах даних на серверах.[14]

Відповідно до рисунку 2.1 клієнт-серверну архітектуру можна означити, як концепцію інформаційної мережі в якій основна частина її ресурсів зосереджена в серверах, обслуговуючих своїх клієнтів. Така архітектура визначає такі типи компонентів:

— набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

— набір клієнтів, які використовують сервіси, що надаються серверами;

— мережа, яка забезпечує взаємодію між клієнтами та серверами.

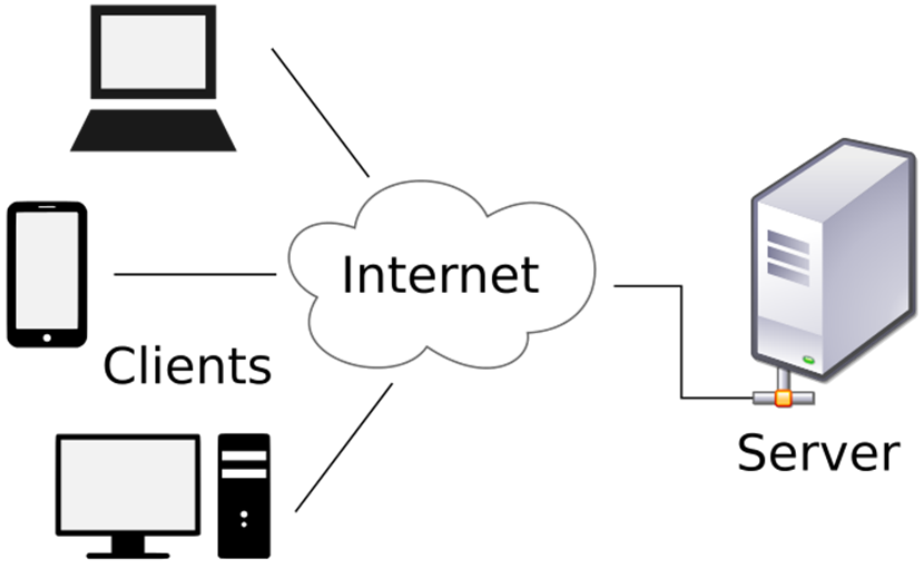


Рисунок 2.1 – Схема структурна роботи сервера

Правила взаємодії між клієнтом і сервером називаються протоколом обміну (протоколом взаємодії)

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов’язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

— рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

— прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації;

— рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дволанкова клієнт-серверна архітектура передбачає взаємодію двох програмних модулів — клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

— модель тонкого клієнта, в рамках якої вся логіка застосунку та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;

— модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Трьохланкова клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка застосунку. Програми проміжного рівня можуть функціювати під управлінням спеціальних серверів застосунків, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних. [15]

Дволанкова архітектура простіша, так як всі запити обслуговуються одним сервером, але саме через це вона менш надійна і висуває підвищені вимоги до продуктивності сервера.

Триланкового архітектура складніша, але завдяки тому, що функції розподілені між серверами другого і третього рівня, ця архітектура проявляє:

— високий ступінь гнучкості і масштабованості.

— високу безпеку (тому що захист можна визначити для кожного сервісу або рівня).

— високу продуктивність (тому що завдання розподілені між серверами).

Прикладом клієнт-серверної взаємодії є сервіс WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.[16]

Основна ідея архітектури «клієнт-сервер» полягає в поділі мережевого додатки на кілька компонентів, кожен з яких реалізує специфічний набір сервісів. Компоненти такого додатку можуть виконуватися на різних комп'ютерах, виконуючи серверні і/або клієнтські функції. Це дозволяє підвищити надійність, безпеку і продуктивність мережевих додатків і мережі в цілому.

Ролі серверів

Роль — це функція сервера (наприклад поштовий, контролер домена тощо). Один сервер може відігравати як одну так і декілька ролей одночасно.

В залежності від ролі, сервісу який надається, розрізняють такі сервери:

Веб-сервер (Web-Server)

Сервер, що приймає HTTP-запити від клієнтів, зазвичай веб-браузерів, видає їм HTTP-відповіді, зазвичай разом з HTML-сторінкою, зображенням, файлом, медіа-потоком або іншими даними. Веб-сервер — основа Всесвітньої павутини.

Веб-сервером називають як програмне забезпечення, що виконує функції веб-сервера, так і комп'ютер, на якому це програмне забезпечення працює.

Клієнти дістаються веб-сервера за URL-адресою потрібної їм веб-сторінки або іншого ресурсу.

Сервер застосунків (Application Server)

Сервер, що виконує деякі прикладні програми. Термін також відноситься і до програмного забезпечення, що встановлено на такому сервері і забезпечує виконання прикладного ПЗ.

Сервери баз даних

Сервери баз даних використовуються для опрацювання користувацьких запитів мовою SQL. При цьому СУБД знаходиться на сервері, до якого і підключаються клієнтські додатки.

Файловий сервер (File Server)

Сервер що зберігає інформацію у вигляді файлів і представляє користувачам доступ до неї. Як правило файл-сервер забезпечує і певний рівень захисту від несакціонірованного доступу.

Сервер друку (Print Server)

Сервери друку використовуються для надання та управління доступом до принтерів. Роль сервера друку дозволяє управляти принтерами через веб-оглядач, друкувати через URL принтера, використовуючи протокол IPP, а також підключати принтери, використовуючи Point and Print.

Поштовий сервер (Mail Server)

Дозволяє обслуговувати базові поштові скриньки ваших користувачів і дозволяє приймати і відправляти пошту з сервера. Вхідна пошта може зберігатися на сервері, а потім забиратися користувачем по протоколу POP3. Для ролі поштового сервера ви повинні мати: Активне з’єднання з інтернет Зареєстроване доменне ім’я Запис MX у провайдера для вашого поштового домен

Термінальний сервер (Terminal Server)

Сервер, що надає клієнтам обчислювальні ресурси (процесорний час, пам’ять, дисковий простір) для вирішення завдань. Технічно термінальний сервер — надпотужний комп’ютер (або кластер), підключений до мережі з термінальними клієнтами — у котрих є, як правило, малопотужні або застарілі робочі станції або спец рішення для доступу до термінального сервера. Термінальний сервер служить для віддаленого обслуговування користувача з наданням робочого столу.

Remote Access/VPN Server

Сервери віддаленого доступу і VPN надають точку входу в вашу мережу для віддалених користувачів. Використовуючи роль Remote Access / VPN Server, ви можете реалізувати протоколи маршрутизації для середовищ LAN і WAN. Ця роль підтримує модемні з’єднання і VPN через інтернет.

DNS Server

Служба DNS дозволяє перетворювати доменні імена (FQDN) в адреси IP.

DHCP Server

Сервер DHCP дозволяє клієнтам отримувати свій IP за потребою. Сервер DHCP також надає додаткову інформацію для конфігурації мережі — адреса серверів DNS, WINS і т.п.

Streaming Media Server

Використовуються для управління і доставки мультимедійного контенту

Ігровий сервер (Game server)

Сервер, що забезпечує зв’язок між різними клієнтами, надаючи їм можливість комунікації один з одним в рамках програмної оболонки конкретної гри. [14]

2.2 Протоколи передачі даних та FTP-сервер VSFTPD

Комп'ютери в Інтернеті можуть безперешкодно спілкуватися один з одним, незалежно від платформи або операційної системи, що використовується. Це стає можливим завдяки дотриманню набору правил передачі даних, відомих як протокол TCP/IP. Назва походить від двох ключових протоколів, що входять до цього набору: TCP та IP. TCP забезпечує надійний зв'язок між комп'ютерами і керує передачею даних, розбиваючи їх на пакети, кожному з яких присвоюється номер для забезпечення правильного відновлення після отримання. Протокол IP додає службову інформацію до кожного пакета, включаючи адреси відправників і одержувачів, гарантуючи, що всі пакети будуть доставлені адресату, навіть якщо вони проходять різними шляхами і прибувають не в повному обсязі. Потім протокол TCP впорядковує пакети і збирає їх у повне повідомлення. Цей набір

протоколів використовується всіма службами Інтернету, навіть якщо кожна служба має свій власний унікальний протокол. [17]

HTTP (Hyper Text Transfer Protocol) - це протокол, який використовується для доступу до гіпертекстових документів, зокрема веб-сторінок, створених за допомогою HTML.[18] FTP (File Transfer Protocol) - протокол для роботи з файлами. [19] SMTP (Simple Mail Transfer Protocol) використовується для передачі вихідних повідомлень електронної пошти [20], тоді як POP3 (Post Office Protocol 3) і IMAP (Internet Message Access Protocol) використовуються для отримання вхідних повідомлень електронної пошти. IMAP є більш досконалим, ніж POP3, і дозволяє користувачам працювати з сервером, як з власним жорстким диском, створювати папки, переміщати і копіювати повідомлення, а також копіювати їх на свій комп'ютер. [21]

Для завантаження файлів на сервер користувачі використовують FTP-клієнти, а сам сервер працює як FTP-сервер. Одним з таких серверів є VSFTPD, який пропонує поєднання функціональності, стабільності та безпеки. Про його ефективність свідчить те, що його включено до стандартного пакунку багатьох проектів, таких як ftp.debian.org, ftp.freebsd.org та ftp.gnu.org. Сервер може похвалитися широким спектром можливостей, які зазвичай не зустрічаються в інших FTP-серверах, таких як можливість призначати віртуальні IP-адреси, створювати віртуальних користувачів, працювати автономно без inetd/xinetd, контролювати швидкість пропускної здатності, конфігуруватися за IP-адресою, реалізувати підтримку IPv6, а також підтримувати SSL-шифрування даних.

У системах Інтернету речей (IoT) обмін даними є ключовим компонентом для забезпечення злагодженої роботи пристроїв. Протоколи передачі даних відіграють важливу роль у забезпеченні сумісності між пристроями, ефективної передачі інформації та безпеки. FTP-сервер VSFTPD (Very Secure FTP Daemon) є одним із найбільш надійних рішень для передачі файлів у таких системах.

Основні протоколи передачі даних у IoT

1. HTTP (HyperText Transfer Protocol):

   Протокол, який використовується для передачі гіпертекстових документів,

зокрема веб-сторінок. У контексті IoT HTTP застосовується для передачі даних між клієнтами та серверами, а також для інтеграції з веб-інтерфейсами.

2. FTP (File Transfer Protocol):

   FTP забезпечує можливість передачі файлів між клієнтом і сервером. Це один із найстаріших і найпоширеніших протоколів передачі даних, який підтримує як текстові, так і двійкові файли. У цьому проєкті FTP використовується для управління файлами на сервері.

3. MQTT (Message Queuing Telemetry Transport):

   Легковаговий протокол для передачі даних, який ідеально підходить для пристроїв із обмеженими ресурсами. Використовується для обміну повідомленнями між IoT-пристроями в режимі реального часу.

4. CoAP (Constrained Application Protocol):

   Протокол, розроблений спеціально для IoT-систем. Він оптимізований для роботи з обмеженими ресурсами та підтримує передачу даних через UDP.

5. SFTP (Secure File Transfer Protocol):

   Більш безпечний варіант FTP, який забезпечує шифрування передачі даних. Використовується в системах із підвищеними вимогами до безпеки.

6. WebSocket:

   Двонаправлений протокол передачі даних, який забезпечує постійне з’єднання між клієнтом і сервером. Він корисний для передачі даних у реальному часі, наприклад, у системах моніторингу.

7. Bluetooth і BLE (Bluetooth Low Energy):

   Протоколи для передачі даних на короткі відстані. BLE використовується для економії енергії, що робить його ідеальним для IoT-пристроїв із батарейним живленням.

Особливості FTP-сервера VSFTPD

VSFTPD є одним із найпопулярніших FTP-серверів завдяки своїй швидкості, стабільності та безпеці. Цей сервер використовується в багатьох відомих проєктах, включаючи ftp.debian.org, ftp.freebsd.org і ftp.gnu.org.

Основні переваги VSFTPD:

1. Безпека:
    - Підтримка SSL/TLS для шифрування даних під час передачі.
    - Захист від brute-force атак через механізми обмеження підключень.
    - Можливість використання віртуальних користувачів із обмеженим доступом.
2. Стабільність:

   Сервер може обробляти тисячі підключень одночасно, зберігаючи при цьому високу продуктивність.
3. Гнучкість налаштувань:
    - Підтримка індивідуальних конфігурацій для кожного клієнта.
    - Встановлення обмежень на швидкість передачі даних і кількість активних сесій.
    - Можливість призначення віртуальних IP-адрес.
4. Підтримка IPv6:

   VSFTPD забезпечує повну сумісність із сучасними мережевими протоколами.

Функціональність VSFTPD у проєкті

У межах цього проєкту VSFTPD використовується для:

- Управління файлами конфігурації IoT-пристроїв.
- Завантаження оновлень програмного забезпечення для IoT-компонентів.
- Резервного копіювання даних, зібраних із сенсорів та інших пристроїв.

Оптимізація роботи VSFTPD для Raspberry Pi

Оскільки Raspberry Pi має обмежені ресурси, конфігурація VSFTPD була адаптована для максимальної продуктивності:

- Обмежено кількість активних підключень одночасно (max_clients).
- Встановлено обмеження на швидкість передачі даних для кожного клієнта (local_max_rate).
- Увімкнено режим автономної роботи без inetd/xinetd, що зменшило використання ресурсів.

Роль протоколів у IoT

Протоколи передачі даних відіграють ключову роль у забезпеченні сумісності між пристроями та системами. Їх правильний вибір залежить від вимог до безпеки, швидкості передачі даних, енергоефективності та масштабованості. Використання VSFTPD у поєднанні з іншими протоколами, такими як MQTT чи CoAP, дозволяє створити надійну і функціональну інфраструктуру для IoT-систем.

Поєднання традиційних протоколів, таких як FTP, із сучасними технологіями передачі даних забезпечує ефективну роботу IoT-систем. FTP-сервер VSFTPD у цьому контексті залишається одним із ключових елементів для обміну файлами та управління даними, зберігаючи баланс між продуктивністю, безпекою та гнучкістю налаштувань.

2.3 Порівняння існуючих рішень

1. Комерційні системи розумного будинку (Amazon Alexa, Google Home, Apple HomeKit)

Переваги:

Інтуїтивно зрозумілий інтерфейс:

Комерційні системи створені з урахуванням масового споживача. Простий інтерфейс забезпечує легкість використання навіть для людей, які не мають технічного досвіду.

Приклади: голосове управління через Amazon Alexa або Google Assistant.

Широка підтримка пристроїв:

Виробники таких систем активно співпрацюють з іншими брендами, що дозволяє інтегрувати широкий спектр пристроїв — від розумних лампочок до термостатів.

Хмарна інфраструктура:

Використання хмарних обчислень забезпечує стабільність і можливість доступу до системи з будь-якого місця, де є Інтернет.

Недоліки:

Висока вартість:

Хоча базові пристрої можуть бути доступними, повна інтеграція системи (особливо для великих будинків) може обійтися дуже дорого.

Залежність від хмарних сервісів:

Без постійного Інтернет-з’єднання або у разі збою серверів функціональність системи може суттєво обмежитися.

Потенційні ризики конфіденційності:

Збирання даних користувачів (наприклад, запис голосових команд або даних про споживання енергії) може становити ризик для приватності.

Обґрунтування:

Комерційні рішення ідеально підходять для користувачів, які готові платити за простоту, стабільність та широку інтеграцію. Вони мають перевагу в доступності і широкому функціоналі, але є менш гнучкими і дорожчими, ніж DIY-рішення.

2. DIY-рішення на базі Raspberry Pi та інших одноплатних комп'ютерів

Переваги:

Доступність:

Raspberry Pi та аналогічні пристрої коштують значно дешевше, ніж комерційні системи. Вартість базового обладнання (Raspberry Pi, аксесуари) становить близько $35-50.

Повна кастомізація:

Користувач має можливість створити систему, яка повністю відповідає його потребам. Це стосується як програмного забезпечення, так і апаратного складу.

Локальне зберігання даних:

Дані зберігаються на локальному сервері, що забезпечує конфіденційність і незалежність від сторонніх хмарних сервісів.

Гнучкість у виборі програмного забезпечення:

Можливість використовувати різні операційні системи (наприклад, Raspbian) і платформи (Home Assistant, OpenHAB).

Недоліки:

Складність налаштування:

Для створення та підтримки такої системи потрібні базові знання програмування, мережевих технологій і роботи з Linux.

Обмежена підтримка пристроїв:

Підключення окремих пристроїв може вимагати спеціальних драйверів, додаткового налаштування або навіть самостійної розробки інтеграцій.

Обмежені апаратні ресурси:

Raspberry Pi має обмежену обчислювальну потужність, що може бути критично при використанні великої кількості датчиків або інтенсивних обчислень.

Обґрунтування:

DIY-рішення — це ідеальний вибір для ентузіастів і тих, хто хоче мінімізувати витрати, але готовий інвестувати час у налаштування системи. Вони дають повну свободу дій, але вимагають більше технічних знань. У вашому проєкті використання Raspberry Pi дозволяє реалізувати доступне та гнучке рішення для розумного будинку.

3. Open-source платформи (Home Assistant, OpenHAB)

Переваги:

Повна гнучкість у налаштуванні:

Open-source платформи дозволяють налаштовувати систему відповідно до потреб користувача. Можна змінювати інтерфейси, сценарії автоматизації та логіку роботи.

Підтримка великої кількості пристроїв і протоколів:

Home Assistant та OpenHAB підтримують десятки протоколів (MQTT, Z-Wave, Zigbee тощо) та інтегруються з популярними пристроями різних виробників.

Активна спільнота:

Користувачі цих платформ створюють численні інструкції, відеоуроки та доповнення, які допомагають у налаштуванні та розв’язанні проблем.

Безкоштовність:

Платформи відкритого коду безкоштовні, і їх можна використовувати без фінансових витрат на ліцензії.

Недоліки:

Залежність від компетенції користувача:

Для використання всіх можливостей потрібне розуміння принципів роботи IoT, мережевих протоколів і навички програмування.

Відсутність офіційної технічної підтримки:

У разі виникнення проблем доведеться звертатися до спільноти або розбиратися самостійно.

Менш стабільна робота порівняно з комерційними системами:

Через відкритий код можливі баги та проблеми з оновленнями, які іноді впливають на стабільність.

Обґрунтування:

Open-source платформи пропонують ідеальний баланс між доступністю та функціональністю. Вони дозволяють розробникам створювати інтегровані системи для розумного будинку без значних фінансових витрат. Використання таких платформ у вашому проєкті дозволяє забезпечити гнучкість і розширюваність, але потребує підготовки для оптимальної реалізації.

4. Індустріальні IoT-рішення (наприклад, Siemens IoT, Bosch IoT Suite)

Переваги:

Висока надійність:

Ці рішення створені для інтенсивного використання в промислових умовах і мають суворі стандарти якості, що гарантують їх стабільну роботу.

Масштабованість:

Продукти легко інтегруються у великі системи та можуть обробляти значний обсяг даних або кількість пристроїв.

Інтеграція з промисловими стандартами:

Вони підтримують стандарти безпеки, передачі даних і сумісність із широким спектром обладнання.

Професійна технічна підтримка:

Виробники пропонують консультації, технічну допомогу та навчання для клієнтів.

Недоліки:

Висока вартість:

Такі системи орієнтовані на бізнес і мають значно вищу ціну, ніж рішення для приватних будинків.

Складність впровадження:

Інтеграція потребує участі досвідчених спеціалістів, що збільшує загальну вартість реалізації.

Зайва функціональність для приватних будинків:

Багато функцій, таких як управління складними промисловими процесами, можуть бути зайвими для потреб звичайного користувача.

Обґрунтування:

Індустріальні IoT-рішення оптимальні для великих підприємств або комплексних проєктів, де потрібна висока надійність і підтримка. Для вашого проєкту вони, скоріш за все, не є доцільними через їхню вартість і складність, проте деякі ідеї або протоколи можна запозичити для підвищення ефективності системи розумного будинку.

5. Рішення на основі Raspberry Pi із власною конфігурацією

Переваги:

Доступність і низька вартість:

Raspberry Pi та супутні компоненти дозволяють створити функціональну систему за значно менший бюджет порівняно з комерційними або індустріальними рішеннями.

Незалежність від хмарних сервісів:

Усі дані обробляються і зберігаються локально, що усуває ризики, пов'язані з передачею даних стороннім сервісам.

Гнучкість і адаптивність:

Ви можете повністю адаптувати систему під конкретні потреби користувача, змінюючи програмне забезпечення, налаштування, функціонал.

Навчальна цінність:

Процес створення і налаштування системи є корисним для студентів, які вивчають IoT, програмування або системне адміністрування.

Модульність:

Можливість інтегрувати додаткові датчики та функції з мінімальними витратами.

Недоліки:

Необхідність технічних знань:

Налаштування і підтримка системи вимагають навичок роботи з Linux, мережами, IoT-протоколами.

Обмежена потужність:

Raspberry Pi може не впоратися з обробкою великої кількості даних або одночасною роботою складних додатків.

Відсутність технічної підтримки:

У разі проблем користувач повинен вирішувати їх самостійно або шукати допомогу у спільноті.

Менш розвинута екосистема:

Порівняно з комерційними системами, підтримка додаткових пристроїв і програмного забезпечення може бути менш зручною.

Обґрунтування:

Ваше рішення пропонує унікальне поєднання доступності, локального управління та адаптивності, що робить його ідеальним для ентузіастів та малобюджетних проєктів. Воно вирізняється своєю універсальністю, але може бути складним для непідготовлених користувачів. Такий підхід дозволяє створити систему, яка задовольняє базові потреби розумного будинку, одночасно забезпечуючи навчальну та дослідницьку платформу.

Я створив таблицю порівняльного аналізу різних рішень для розумних будинків, яка включає ключові критерії оцінки (вартість, простота використання, гнучкість, приватність, масштабованість, рівень необхідних технічних знань).

Критерії оцінки:

1. Вартість: витрати на реалізацію та обслуговування.
2. Простота використання: зручність роботи для кінцевого користувача.
3. Гнучкість: можливість адаптації під специфічні потреби.
4. Конфіденційність: рівень захисту даних і незалежності від сторонніх сервісів.
5. Масштабованість: здатність системи адаптуватися до збільшення кількості пристроїв.
6. Необхідні технічні знання: рівень знань і навичок для впровадження та налаштування системи.

Підходи:

1. Комерційні системи (наприклад, Amazon Alexa, Google Home):
    - Висока простота використання (5 балів).
    - Недоліки: висока вартість (4 бали) і низький рівень конфіденційності (2 бали).
2. Рішення "Зроби сам" (DIY на базі Raspberry Pi):
    - Високий рівень гнучкості і конфіденційності (по 5 балів).
    - Недоліки: потребує значних технічних знань (4 бали).
3. Платформи з відкритим вихідним кодом (наприклад, Home Assistant, OpenHAB):
    - Поєднує хорошу гнучкість (5 балів) і конфіденційність (4 бали).
    - Вимагає помірних технічних знань (4 бали).
4. Промислові IoT-рішення (Siemens IoT, Bosch IoT Suite):
    - Висока масштабованість (5 балів) і надійність.
    - Значна вартість і складність впровадження.
5. Запропоноване рішення:

- Гнучкість і конфіденційність на рівні DIY (по 5 балів).
- Низька вартість (1 бал), але потребує певного рівня технічної компетенції (4 бали).

Ця таблиця дозволяє зрозуміти, що запропоноване рішення базується на оптимальному співвідношенні ціни, функціональності та технічних можливостей, особливо для користувачів, які готові інвестувати час у навчання та налаштування системи

Таблиця 1 – Порівняльний аналіз різних рішень для розумних будинків

Рішення	Вартість	Простота використання	Гнучкість	Конфіденційність	Масштабованість	Технічні знання
Комерційні системи	4	5	3	2	4	1
Рішення «зроби сам	1	2	5	5	3	4
Платформи з відкритим вихідним кодом	2	3	5	4	4	4
Промисловий IoT	5	4	3	3	5	4
Запропоноване рішення	1	2	5	5	3	4

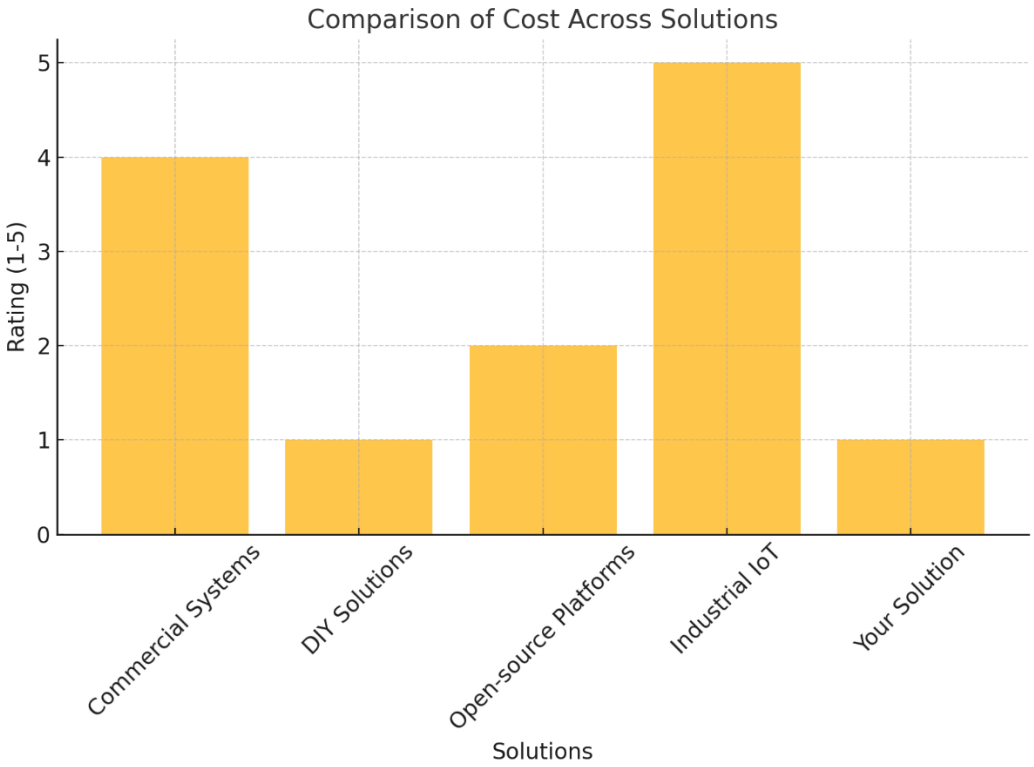


Рисунок 2.2 — Порівняльний аналіз за вартістю

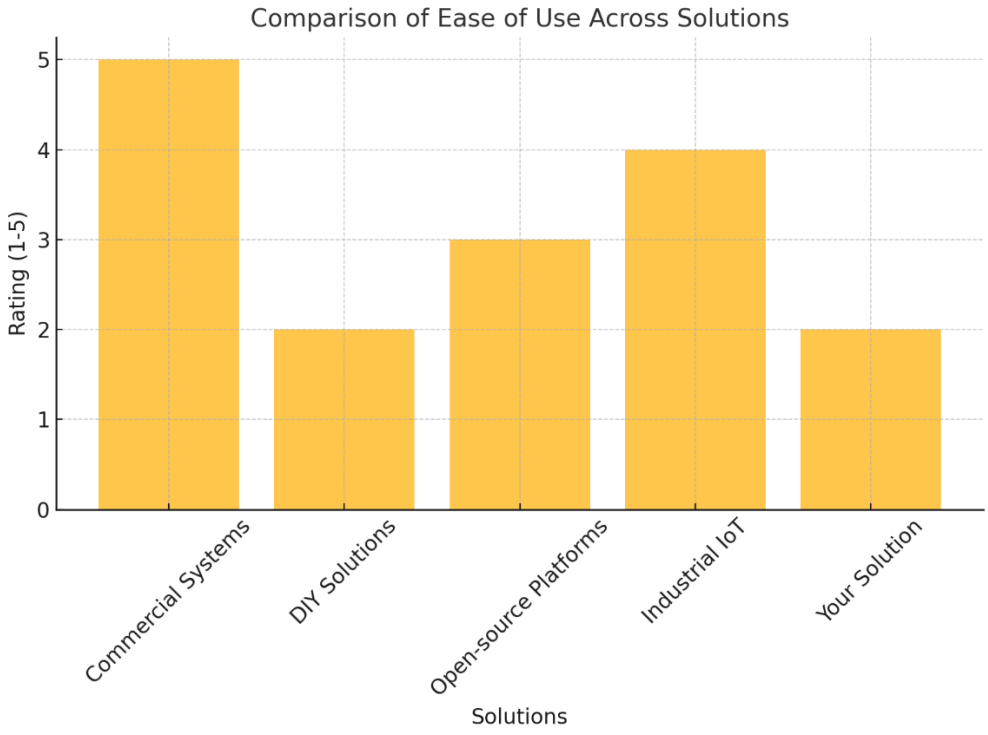


Рисунок 2.3 — Порівняльний аналіз за простотою використання

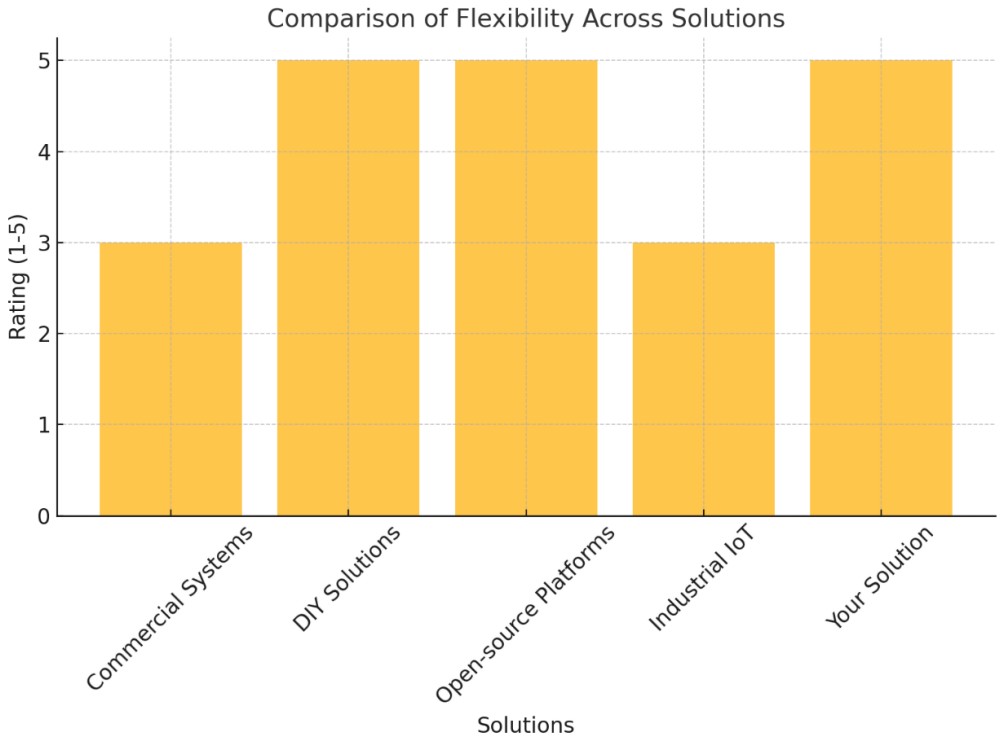


Рисунок 2.4 — Порівняльний аналіз за гнучкістю системи

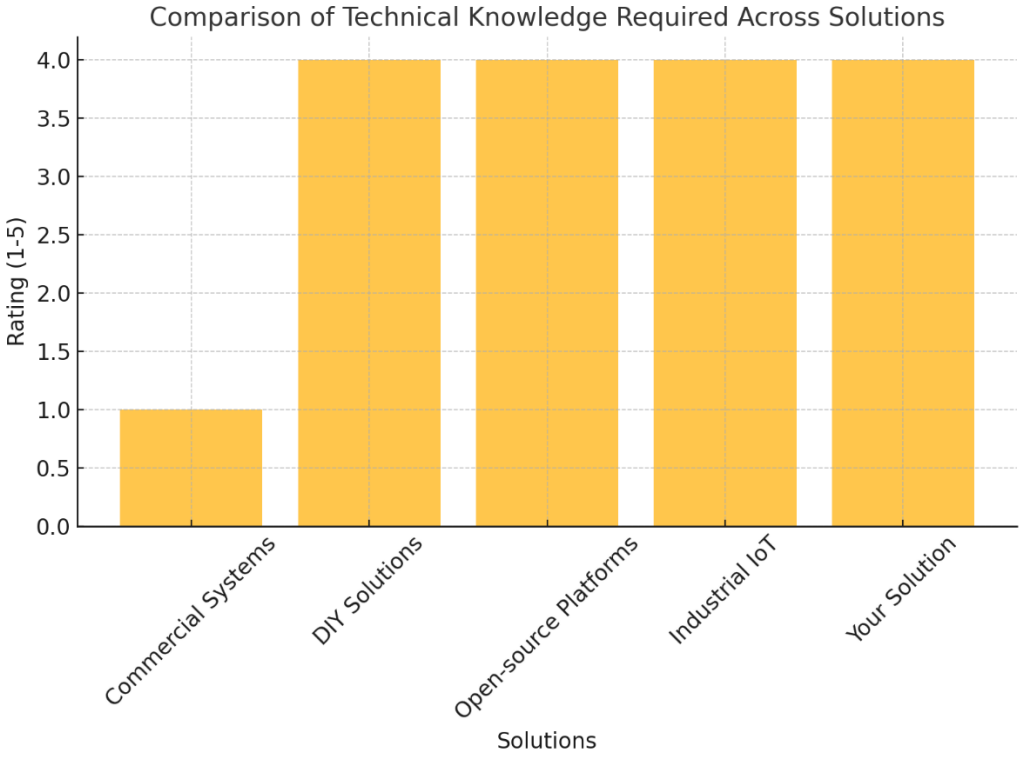


Рисунок 2.3 — Порівняльний аналіз за технічними знаннями

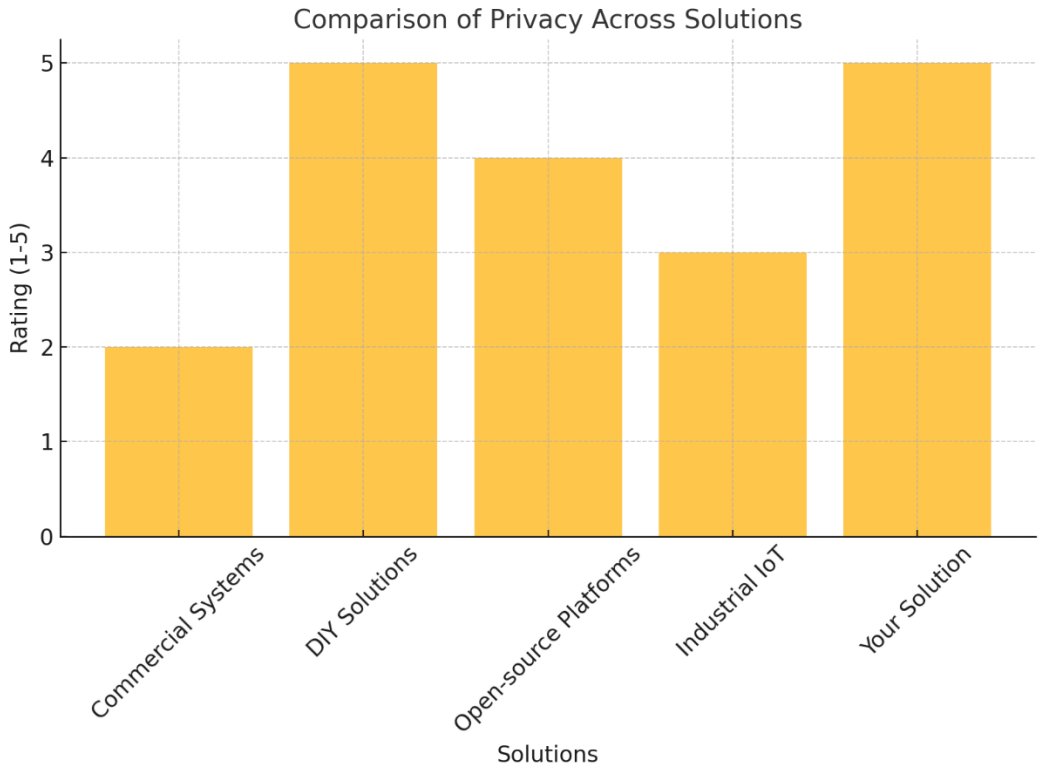


Рисунок 2.3 — Порівняльний аналіз за конфіденційністю системи

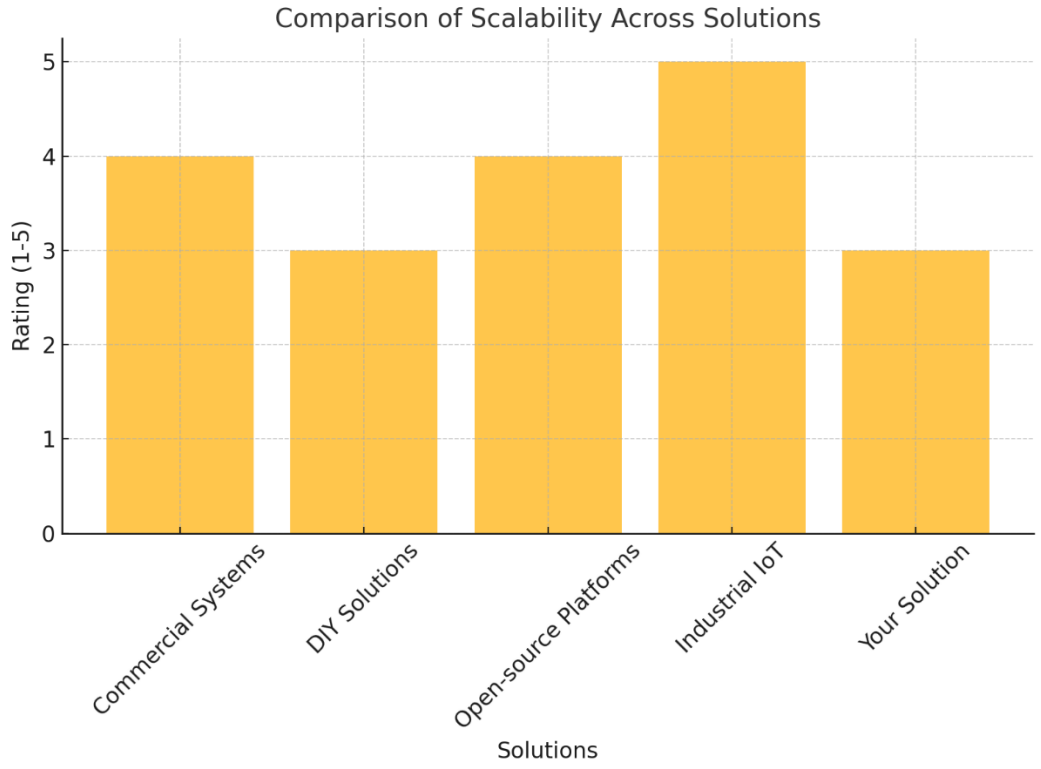


Рисунок 2.3 — Порівняльний аналіз за масштабованістю системи

РОЗДІЛ 3 ОПИС ТЕХНОЛОГІЙ ПРОЕКТУ СИСТЕМИ РОЗУМНИЙ БУДИНОК

3.1 Мікрокомп’ютер Raspberry Pi 3B+ та його операційна система

В якості платформи веб-сервера було використано мікрокомп’ютер Raspberry Pi Foundation, Raspberry PI 3B+. Цей мікрокомп’ютер випускається з 29 лютого 2016 року, і його ключовою відмінністю від попереднього покоління є використання 64-розрядного процесора ARM, а також додавання мікросхем WIFI і Bluetooth, розпаяних на платі. На сьогоднішній день це найсучасніший і найпотужніший мікрокомп’ютер, вироблений Raspberry Pi Foundation. На рисунку 1.4 показано зовнішній вигляд пристрою.

Рисунок 3.1 – Мікрокомп’ютер Raspberry Pi 3B+

Вибір платформи для веб-сервера базувався на кількох ключових факторах. По-перше, платформа мала бути популярною серед аматорів, і Raspberry Pi відповідала цьому критерію, адже було продано понад 12,5 мільйонів пристроїв. Крім того, спільнота Raspberry Pi є дуже активною, надаючи багато знань та підтримки. Raspberry Pi – це повнофункціональний комп’ютер, оснащений 4 портами USB, виходом HDMI і звуковим виходом. Чотирьохядерний 64-бітний

процесор, що використовується в Raspberry Pi 3 моделі B, на сьогоднішній день є найпотужнішим з усіх представлених рішень. При вартості близько 35 доларів США, Raspberry Pi є дуже доступним за ціною.

Однією з головних переваг Raspberry Pi є його універсальність, оскільки його можна використовувати як медіацентр, ігровий автомат або «мозок» робота. Розмір Raspberry Pi еквівалентний пластиковій картці, що робить його дуже портативним. Пристрій також може похвалитися високою кастомізацією, що дозволяє користувачам зібрати кластер мікрокомп'ютерів або вибрати будь-який корпус, який їм подобається. Raspberry Pi має 40 універсальних контактів вводу/виводу GPIO, що дозволяє йому взаємодіяти з широким спектром пристроїв.

Raspberry Pi є абсолютно безшумним пристроєм, оскільки використовує пасивне охолодження, що робить його ідеальним для чутливих до шуму додатків. Він також має низьке енергоспоживання.

Raspberry Pi 3 Model B оснащений 64-бітним чотирьохядерним процесором ARM Cortex-A53 на базі однокристальної мікросхеми Broadcom BCM2837, з тактовою частотою процесора 1,2 ГГц. Чіп має приріст продуктивності близько 60% у порівнянні з попереднім поколінням Raspberry Pi 2 Model B і десятикратний приріст продуктивності у порівнянні з першим поколінням Raspberry Pi. Мікрокомп'ютер має 1 ГБ пам'яті LPDDR2 SDRAM, яка використовується спільно з графічним чіпом. В Raspberry Pi використовується двоядерний графічний процесор VideoCore IV, який підтримує стандарти OpenGL ES 2.0, OpenVG, MPEG-2, VC-1. Можливості чіпа вражають: він здатний кодувати, декодувати і виводити відео у форматі Full HD (1080p, 30 кадрів в секунду, H.264 High-Profile).

Мікрокомп'ютер Raspberry Pi 3 Model B оснащений різними периферійними пристроями для полегшення його використання. Для підключення дисплея на платі є роз'єм HDMI, який підтримує роздільну здатність від 640 x 350 до 1920 x 1200. Аудіороз'єм 3,5 мм можна використовувати для підключення колонок або навушників, а також передавати

звук через HDMI. Мікрокомп’ютер має 4 USB-порти, які з’єднані через загальний концентратор і можуть використовуватися для підключення клавіатури, миші та інших периферійних пристроїв. До слотів CSI-2 та DSI мікрокомп’ютера можна підключати стандартні модулі для раціонального використання ресурсів процесора.

Мікрокомп’ютер має 40 універсальних портів вводу/виводу, які можна конфігурувати і використовувати для зв’язку між компонентами системи, такими як мікропроцесор і периферійні пристрої. Ці порти GPIO використовуються для різних цілей, таких як отримання інформації від датчиків, керування двигунами постійного струму, аудіо, відео, РК-дисплеями, світлодіодами, перемикачами, реле, джойстиками та фотодатчиками.

Для зв’язку мікрокомп’ютер має різні інтерфейси, такі як 10/100 Мбіт Ethernet, Wi-Fi 802.11n та Bluetooth 4.1, які забезпечуються мікросхемою Broadcom BCM43438. На платі також є UART (послідовний), I2C/TWI, SPI, а також виводи для підключення 3,3 В, 5 В і заземлення.

Мікрокомп’ютер можна живити за допомогою 5-вольтового адаптера, підключеного до роз’єму micro-USB або до контактів живлення на GPIO-панелі. Для роботи потрібен струм мінімум 2 А, який необхідний для живлення USB-портів. Raspberry Pi 3 Model B споживає від 2 до 3 Вт під час роботи, 4 Вт при максимальному навантаженні пристрою і 1 Вт при зупинці.

Мікрокомп’ютер Raspberry Pi 3B+ має розміри 85x54 мм, як стандартна банківська картка, і використовує флеш-карти MicroSD для зберігання інформації. Він може підтримувати карти ємністю 4 ГБ і більше, а для передачі файлів використовується стандарт USB 2.0, швидкість залежить від характеристик карти. Завантажувач та операційна система встановлюються на карту MicroSD, а для запуску декількох операційних систем можна використовувати декілька карт. [24]

Операційною системою за замовчуванням для Raspberry Pi 3B+ є Linux, але також можна встановити Windows 10 IOT. Цей мікрокомп’ютер має найбільшу кількість підтримуваних операційних систем, як офіційних, так і неофіційних.

До офіційних належать Raspbian, Pidora, OpenELEC, OSMC, RISC OS та Windows 10 IOT. Неофіційні: A2, FreeBSD, OpenWrt, HypriotOS, Kali Linux, Archlinux ARM, Raspbian Server Edition, RasPBX, IPFire, Raspberry Pi Thin Client, Parrot Security OS, Ubuntu та Wtware. Деякі операційні системи на стадії розробки включають Android, Chromium OS та Puppy Linux. [25]

Raspberry Pi 3B+ є чудовим вибором для розгорнутого веб-сервера завдяки своєму чотирьохядерному процесору та низькому енергоспоживанню. Операційна система Raspbian є найбільш оптимізованою для Raspberry Pi і розповсюджується безкоштовно. Raspbian базується на Debian GNU/Linux, який має найбільший репозиторій пакунків серед дистрибутивів Linux і сувору політику щодо пакунків, що забезпечує високу якість випусків. Debian GNU/Linux також посіла перше місце в рейтингу серверних операційних систем за версією журналу Tagline, складеному на основі опитування 180+ цифрових агентств з серпня 2014 по квітень 2016 року.

3.2 Опис веб-сервера Apache

В якості основного HTTP-сервера був обраний Apache HTTP-сервер. Apache належить до рангу кросплатформенного, який вільно поширює програмне забезпечення і підтримує такі операційні системи як: Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS. Розробкою і підтримкою сервера займається відкрите співтовариство розробників під контролем компанії Apache Software Foundation. Apache HTTP Server входить до складу багатьох програмних продуктів, таких як: СУБД Oracle і IBM Netcraft. На основі статистичних даних компанії Netcraft, Apache з 1996 і по теперішній час є самим частіше використовуваним HTTP-сервером, а також найпопулярніший сервер з часткою ринку 65%. [26]

Головними особливостями Apache є те, що він надає підключення зовнішніх модулів для представлення даних, використання СУБД для аутенфікації і авторизації користувачів, модифікувати повідомлення про помилки. Є підтримка Ipv6. Ядро Apache володіє наступними можливостями:

– обробка конфігураційних файлів;

– протокол HTTP;

– система завантаження модулів.

Ядро розробляється саме компанією Apache Software Foundation, у сторонніх програмістів доступ до розробки закритий. Ядро написано повністю на мові C.

Конфігурація Apache заснована на використанні текстових конфігураційних файлів.

Є три рівні конфігурації:

– безпосередньо конфігурація сервера (httpd.conf);

– конфігурація віртуального хоста (httpd.conf с версії 2.2, extra/httpd-vhosts.conf);

– конфігурація рівня директорії (.htaccess).

Присутня власна мова конфігурації, яка заснована на блоках директив. Всі параметри ядра можуть бути змінені через конфігураційні файли, навіть управління MPM (мультипроцесорні моделі). Незалежно від цього, параметри можна передати через ключі командного рядка.

Існує безліч моделей MPM, наприклад:

– Worker. Це гібридна мультипроцесорна – мультипоточна модель. Вона здатна зберігати стабільність мультипроцесорних варіантів виконання і дозволяє величезну кількість клієнтів з мінімізованим використанням ресурсів; [27]

– Pre-fork. Мультипроцесорна модель, основним фактором в роботі моделі є те, що йде генерація окремих процесів, без використання механізму threads; [28]

– Perchild. Мультипроцесорна модель, яка використовує фіксовану модель кількості процесів; [29]

– Netware. Мультипоточна модель, створена і оптимізована для середовища NetWare; [30]

– Winnt. Мультипоточна модель, створена і оптимізована для операційної системи Windows; [31]

– Apache-ITK. Мультипроцесорна модель, яка в свою чергу взяла за основу модель pre-fork, вона дозволяє запуск кожного віртуального хоста під окремим uid і gid; [28]

– Peruser. Мультипроцесорна модель, яка в свою чергу взяла за основу модель perchild. Дозволяє запуск кожного віртуального хоста під окремими uid і gid. Не використовує потоки. [29]

Як було сказано раніше Apache HTTP server підтримує систему модулів. На даний момент існує понад 500 модулів створені як самою компанією Apache Software Foundation, так і спільнотою розробників. Система модулів має досить гнучку структуру, дозволяючи додавати модулі під час компіляції сервера і завантажувати їх динамічно через директиви конфігураційних файлів.

Apache підтримує технологію віртуальних хостів. Це дозволяє виконувати на одній єдиній IP-адресі безліч доменних імен, відображаючи для кожного з них свій власний вміст. Для кожного віртуального хоста є можливість вказати власні налаштування ядра і модулів, обмежити доступ до домену або окремого файлу, дозволити вести облік і обмеження ресурсів сервера (CPU, RAM, трафік).

До всього, Apache підтримує 2 дуже важливі механізми: CGI і FastCGI, що забезпечує виконання програми на всіх мовах програмування, в тому числі на C, C ++, Lua, sh, Java.

Apache містить безліч можливостей по забезпеченню безпеки і розмежування доступу до даних, до них відносяться:

– обмеження доступу до теки та папок;

– авторизація користувачів, для надання доступу до тек на підставі HTTP-аутенфікаціі (modJauthJbasic) і digest-аутенфікаціі (modJauthJdigest);

– за тією чи іншою IP-адресою користувачів, можна обмежити доступ до певних тек або всьому серверу;

– обмеження доступу до певних типів файлів для визначених користувачів або всіх користувачів;

– підтримка авторизації через СУБД або PAM.

Деякі MPM-модулі дозволяють запускати кожен процес Apache з відповідністю кожному користувачеві або групі користувачів, використовуючи різні uid і gid. Шифрування даних відбувається шляхом SSL, через бібліотеку OpenSSL. Для перевірки справжності веб-сервера використовується сертифікати X.509. Так само присутні зовнішні засоби, наприклад, модуль modJsecurity.

Apache має можливість визначення сервером локалі клієнта. Повідомлення про помилки та події, що відправляються браузеру, надаються на декількох мовах і використовують технологи. SSI. Apache підтримує безліч кодувань, в їх числі Unicod, що забезпечує використання сторінок, створених на інших кодуваннях і іншими мовами.

Є можливість установки власних сторінок і обробників для всіх HTTP помилок і подій, на різних мовах, таких як 404 (Not Found) та 403 (Forbidden).

Це механізм, який дозволяє здійснювати динамічне формування HTML-документів на стороні сервера. Модуль modJinclude управляє механізмом SSI і є базовим в пакеті Apache. Gzip представлений модулем mod gzip потрібно включити, що забезпечить стиснення трафіку, файли на зразок CSS, HTML і JavaScript які, наприклад, ніколи не змінюють- ся або змінюються дуже рідко, тепер повинні стискатися і далі модуль кешує заархівовані копії. Необхідно встановити модуль mod_rewrite, який дозволяє динамічно змінювати запитувані дані. Модуль є програмним модулем Apache і не виконується під іншими веб-серверами. Зберігання часто запитуваних даних і контенту забезпечується включенням модуля mod_cache, на відміну від інших способів кешування модуль може обробляти мінливі дані в реальному часі, для цього можливо задавати терміни дії кеша. Кеш зберігається, як на HDD диску, так і в пам’яті, для вилучення і зберігання даних використовуються ключі URL.

3.3 Опис мови PHP

Цей компонент Apache, використовується для обробки коду по відображенню динамічного контенту і додатків. Модуль використовується для запуску скриптів, підключається до баз даних MySQL і передачі обробленого

контенту в веб-сервер для візуалізації. На платформі Raspberry, PHP міститься в репозиторіях Debian представлений у версії 5.6.30 + deb8u1.

Даний компонент виконується в Apache як prefork. У PHP аналогічне управління модулями, як і у Apache. Разом з ним необхідно встановити наступні модулі, щоб розширити функціональність PHP:

– php-curl. Бібліотека для отримання файлів з FTP, Gopher і HTTP-сервера;

– php-gd. Бібліотека для роботи з графікою зі сценаріїв PHP. Підтримуються формати PNG, JPEG, XPM, а також шрифти freetype / ttf;

– php-cgi. Бібліотека являє собою інтерпретатор CGI зроблений для Apache з модулем modJactions або будь-яким іншим інтерпретатором CGI;

– php-cli. Пакет, який являє собою командний інтерпретатор для тестування сценаріїв PHP з консолі або виконання сценаріїв загального призначення;

– php-common. Пакет включає документацію і приклади для пакетів PHP;

– php-dbg. Пакет являє символи налагодження для правильного налагодження помилок PHP;

– php-dev. Бібліотека, що включає файли джерел необхідні до розробки модулів. [32]

3.4 Опис баз даних MySQL

Система управління базами даних (далі СУБД), вільно поширювана реляційна СУБД від компанії Oracle. СУБД MySQL дуже гнучка у використанні, тим, що існує підтримка величезної кількості типів таблиць: користувач може здійснити вибір таблиці типу MyISAM, у неї є повно- текстовий пошук, такі таблиці типу InnoDB, підтримуюча транзакції на рівні окремих записів. Крім цього СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, де демонструється принцип створення нових типів таблиць. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СУБД MySQL йде постійна поява нових типів таблиць. MySQL портована на велику кількість платформ:

– FreeBSD;

– HP-UX;

– Linux;

– Mac OS X;

– NetBSD;

– OpenBSD;

– SGI IRIX;

– Solaris;

– сімейство Windows.[33]

СУБД надає API для мов Delphi, C, C ++, Ейфель, Java, Лисп, Perl, PHP, Python, Ruby, Smalltalk, Компонентний Паскаль і Tcl, бібліотеки для мов платформи .NET, а також забезпечує підтримку для ODBC за допомогою ODBC-драйвера MyODBC. Максимальний розмір таблиць нічим не обмежений, він обмежений лише її типом. У разі InnoDB зберігання таблиці може бути розбите на кілька файлів, складаючи при цьому єдиний табличний простір. Хоча при цьому є обмеження на кількість стовпців, яке можна додати в одну сторінку. Розмір сторінки пам'яті за замовчуванням дорівнює 16 кілобайтам, під дані при цьому відведено 8123 байт.

Розмір покажчика на динамічне поле дорівнює 20 байт. Таким чином у разі використання динамічного формату рядка ROW_FORMAT = DYNAMIC, одна таблиця може вмістити по максимуму 409 стовпців типу blob або text. [34]

На даний момент актуальна версія MySQL - 5.5. Що значить:

– використання за замовчуванням движка InnoDB;

– є підтримка напівсинхронного (Semi-synchronous) механізму реплікації, заснованого на патчах до InnoDB від компанії Google;

– вироблено поліпшення функцій по секціонуванню даних;

– розширений синтаксис для розбиття великих сторінок на кілька частин, розміщених в файлових системах (partitioning);

– додані операції RANGE, LIST і метод оптимізації «partition pruning »;

– новий механізм оптимізації вкладених запитів і JOIN-операцій;

– перероблена система внутрішніх блокувань;

інтегрований патч Google з оптимізацією роботи InnoDB на процесорах з великою кількістю ядер.

3.5 Середовище адміністрування PhpMyAdmin

Ця програма написана на PHP, має відкритий вихідний код і являє собою веб-інтерфейс для управління базами даних СУБД. MySQL через інтернет-браузер і не тільки. PhpMyAdmin виконує адміністрування сервера MySQL, надає можливість перегляду таблиць і вмісту баз даних, запускає команди і запити SQL, додає нових користувачів, запускає команди і запити SQL.[35] Додаток активно використовують веб-розробники, так як розробники постійно враховують зміни в СУБД MySQL, яке переведено на 62 мови. Основними можливостями програми є:

– веб-інтерфейс;

– управління базами даних MySQL і MariaDB;

– імпорт даних з CSV і SQL;

– експорт даних в різні формати: CSV, SQL, XML, PDF (через бібліотеку TCPDF), ISO/IEC 26300 - текст і таблицю OpenDocument, Word, Excel, LaTeX;

– адміністрування декількох серверів;

– створення PDF-графіків баз даних;

– створення складних запитів з використанням Query-by-Example (QBE);

– пошук по всьому світу в базі даних;

– перетворення збережених даних в будь-який формат з використанням набору визначених функцій, таких як відображення BLOB-даних в якості зображення або посилань;

– живі графіки для моніторингу активності сервера MySQL, такі як з'єднання, процеси, використання ЦП/пам'яті тощо;

– робота з різними операційними системами.[36]

Висновок щодо вибору програмного стека один - система проста і має повноцінний функціонал, сервер цілком сучасний, немає нічого зайвого.

3.6 Опис мережевої хмари NextCloud

NextCloud - це потужне та універсальне мережеве хмарне рішення, яке надає користувачам повний контроль над їхніми файлами та даними. Це безкоштовна платформа з відкритим вихідним кодом, що означає, що її функції та можливості можуть бути налаштовані відповідно до потреб користувача. Однією з ключових переваг NextCloud є те, що обсяг його сховища обмежений лише розміром диска, який використовується для зберігання файлів, а це означає, що користувачі можуть розширювати своє сховище в міру зростання своїх потреб. [22]

Доступ до NextCloud простий і зручний через веб-інтерфейс, який дозволяє користувачам переглядати, редагувати та ділитися своїми файлами з будь-якого пристрою, підключеного до Інтернету. Він також пропонує спеціальні додатки для Windows, Linux, Mac OS, Android та IOS, що дозволяють користувачам отримувати доступ до своїх файлів на ходу. На рисунку 3.2 показано схему структурну взаємодії зі сховищем

Рисунок 3.2 – Схема структурна роботи мережевого сховища

На додаток до надійних функцій управління файлами, NextCloud пропонує кілька вбудованих додатків, які покращують користувацький досвід. Він включає систему управління календарем і контактами, що дозволяє легше залишатися організованим і бути в курсі важливих подій. Він також має

вбудований аудіо- та відеопрогравач, який дозволяє користувачам відтворювати свої медіафайли без потреби в додатковому програмному забезпеченні.

Крім того, NextCloud має вбудований блокнот, який дозволяє користувачам робити нотатки, складати списки справ і відстежувати важливу інформацію. Це корисна функція для окремих осіб або команд, які працюють над проектами, оскільки вона допомагає тримати всіх на одній сторінці.

NextCloud також забезпечує шифрування даних, що гарантує безпеку даних користувачів під час їх зберігання в хмарі. Це дуже важлива функція, яка захищає конфіденційну інформацію від несанкціонованого доступу або крадіжки.

NextCloud - це універсальна платформа, яку можна використовувати як у локальній мережі, так і в глобальній мережі Інтернет. Це означає, що користувачі можуть налаштувати власний приватний хмарний сервер для зберігання та управління своїми файлами, або ж вони можуть використовувати загальнодоступні сервери NextCloud, доступні в Інтернеті. Вона є чудовим вибором для приватних осіб, підприємств чи організацій, які потребують надійного та доступного зберігання даних [23]

Основні характеристики NextCloud

1. Відкрите програмне забезпечення:

NextCloud є повністю відкритим і доступним для модифікації, що дозволяє розробникам адаптувати його до своїх потреб. Це також знижує витрати, оскільки немає потреби в ліцензуванні.

2. Шифрування та безпека:

NextCloud підтримує шифрування даних як під час їх передачі, так і в процесі зберігання. Для підвищення безпеки пропонується інтеграція з двофакторною аутентифікацією (2FA), SSL/TLS і керування правами доступу.

3. Файловий менеджмент:

NextCloud забезпечує зручне керування файлами: створення папок, перегляд, завантаження та видалення файлів. Підтримується збереження історії змін і версій файлів, що дозволяє легко відновлювати попередні версії.

4. Синхронізація даних:

Завдяки спеціалізованим клієнтам для Windows, macOS, Linux, Android та iOS користувачі можуть синхронізувати дані між своїми пристроями.

5. Інтеграція з іншими сервісами:

NextCloud інтегрується з численними сторонніми програмами та службами, такими як Office 365, Google Drive, Dropbox та багато інших.

6. Розширення функціональності:

Через систему додатків NextCloud дозволяє додавати функції, зокрема календар, пошту, відеоконференції, запис нотаток, відстеження завдань тощо.

Використання NextCloud у проєкті

У цьому проєкті NextCloud використовується для:

1. Зберігання даних IoT:

Дані, зібрані пристроями IoT, передаються до хмарного сховища NextCloud для подальшого аналізу та доступу користувачів.

2. Резервне копіювання:

Хмарна інфраструктура дозволяє зберігати резервні копії критично важливих файлів, забезпечуючи їх доступність у разі збоїв локального обладнання.

3. Спільний доступ до файлів:

Користувачі можуть обмінюватися файлами або папками через захищені посилання, а також налаштовувати права доступу (читання, редагування тощо).

4. Моніторинг та управління:

Адміністратори можуть легко відстежувати використання сховища, активність користувачів і налаштовувати права доступу.

5. Інтеграція з локальними серверами:

NextCloud налаштовано для синхронізації з іншими сервісами локальної мережі, що дозволяє користувачам отримувати доступ до даних незалежно від місцезнаходження.

Технічні особливості та оптимізація

1. Підтримка Raspberry Pi:

NextCloud може бути встановлений і налаштований на платформі Raspberry Pi

завдяки її легковажності. У цьому проєкті використовується NextCloud у поєднанні з веб-сервером Apache і СУБД MySQL, що забезпечує стабільну та швидку роботу навіть на обмежених ресурсах.

2. Конфігурація безпеки:

Для зменшення ризиків несанкціонованого доступу використовується SSL-сертифікат, а також увімкнено двофакторну аутентифікацію. Сервер конфігуровано для обмеження кількості підключень і використання сильних паролів.

3. Оптимізація продуктивності:

У проєкті використано кешування за допомогою Redis і ввімкнено компресію даних, що зменшує навантаження на сервер.

4. Автоматизація резервного копіювання:

Налаштовано регулярне створення резервних копій бази даних і файлів, які автоматично зберігаються на зовнішньому носії або в іншій хмарі.

Переваги використання NextCloud

1. Приватність:

Дані користувачів зберігаються на власному сервері, що гарантує повний контроль над ними і відповідає сучасним стандартам конфіденційності.

2. Гнучкість:

NextCloud легко масштабується й адаптується під потреби різних проєктів — від невеликих локальних установок до великих корпоративних рішень.

3. Економічність:

Завдяки відсутності ліцензійних платежів NextCloud є економічно вигідним рішенням порівняно з комерційними хмарними сервісами.

4. Підтримка спільної роботи:

Інтеграція з такими інструментами, як Collabora Office або OnlyOffice, дозволяє редагувати документи в реальному часі прямо у веб-інтерфейсі.

Майбутній розвиток та інтеграції

NextCloud активно розвивається, постійно додаючи нові функції. У майбутньому можливе розширення його функціональності в проєкті за рахунок інтеграції з AI-

модулями для аналізу IoT-даних, підключення до інших хмарних платформ і впровадження додаткових інструментів спільної роботи.

NextCloud є потужним і універсальним інструментом для зберігання даних, управління файлами та спільної роботи. У рамках цього проєкту його використання забезпечило високу надійність, безпеку та зручність роботи з даними, що робить його ключовим компонентом системи.

РОЗДІЛ 4 ВСТАНОВЛЕННЯ І КОНФІГУРУВАННЯ ПРОГРАМНО-АПАРАТНОГО СТЕКА

4.1 Встановлення операційної системи

Установка операційної системи Raspbian займає небагато часу, якщо використовувати для цих цілей операційну систему Windows. Установка ОС включає в себе наступні дії:

– завантаження образу операційної системи в форматі .ISO;

– за допомогою програмного продукту Etcher відбувається підготовка інформаційного носія MicroSD до запису образу ОС, відбувається форматування і розбиття носія на необхідну кількість розділів. В підсумку записується образ операційної системи.

На рисунку 4.1 представлено робочій простір після входу в систему після установки операційної системи на носій інформації.

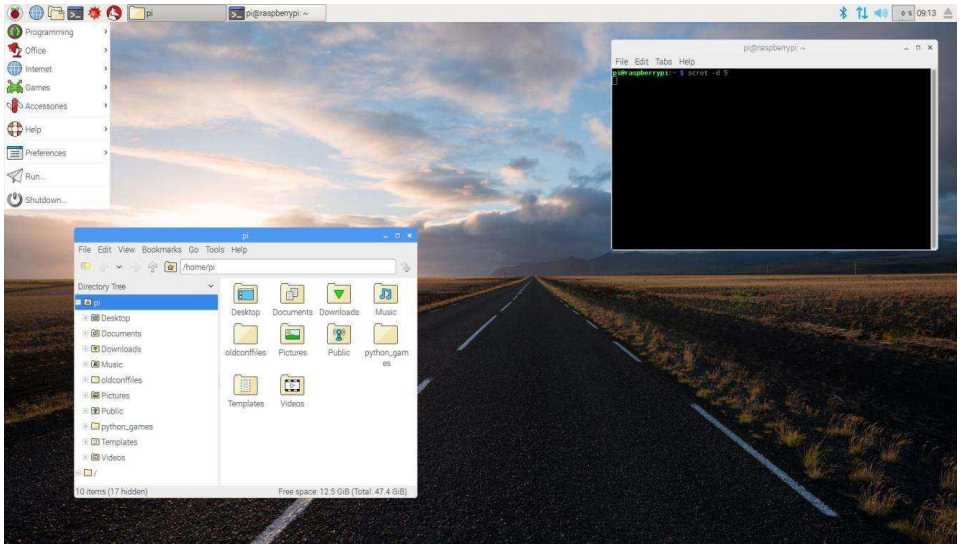


Рисунок 4.1 – Простір робочий Raspbian

Після установки операційної системи була розширена файлова система, так як під час запису образу системи створюється 2 розділа:

– розділ завантажувача системи, що дорівнює 100 Мб;

– розділ файлової системи ext4, що дорівнює розміру файлів системи в процесі запису образу системи.

Після розширення файлової системи весь вільний простір носія тепер належить файлової системи і служить для запису файлів. Було встановлено коректний час, часовий пояс, додані необхідні розкладки клавіатури, оновлені всі стандартні пакети, оновлена прошивка Raspberry Pi 3 і ядро системи.

Так, як модель використання мікрокомп'ютера є застосування його у вигляді веб-сервера, то логічно було б відключення графічного інтерфейсу системи для поліпшення продуктивності і зменшення використання ресурсів.

Основним моментом можна вважати включення віддаленого управління мікрокомп'ютером по протоколу SSH. Дане управління так само повністю відображає модель використання мікрокомп'ютера в якості веб-сервера, так як не вимагає підключення монітора, клавіатури, миші та інших маніпуляторів. Необхідно лише подача живлення і підключення до мережі за допомогою WIFI або LAN-порту. На рисунку 4.2 представлено повністю налаштоване оточення системи після входу в систему для використання мікрокомп'ютера в вигляді веб-сервера.

Рисунок 4.2 – Запуск у вигляді веб-сервера

Управління мікрокомп'ютером здійснювалося за допомогою програми PuTTY[37] версія 0.69 на 64-х бітної версії операційної системи Windows 10. Для управління необхідна наявність комп'ютерів в одній локальній мережі, знання IP-адреси мікрокомп'ютера, його логін і пароль. На рисунку 4.3 показаний інтерфейс програми PuTTY після підключення до мікрокомп'ютеру, введення відповідних логіну і пароля.

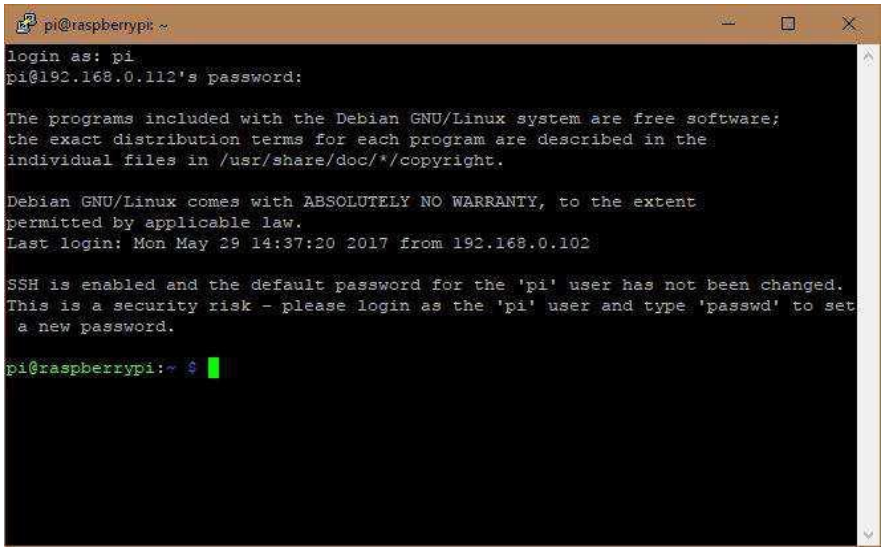


Рисунок 4.3 – Вікно програми PuTTY

4.2 Встановлення і конфігурація програмного стека

Під час установки програмного стека було виявлено одну особливість за замовчуванням конфігураційні файли були відсутні. А так як запуск модулів сервера здійснюється на підставі присутності конфігураційного файлу; якщо файлів немає - модуль не запущено, якщо файл є - модуль запускається. Конфігураційний файл являє собою файл розширення .conf, .ini, або .tld. Містить всі функціональні можливості програм, в ньому описано механізм поведінки програми та параметри ядра і вказані налаштування програми.

Синтаксис віддалено нагадує мову XML і заснований на блоках директив. Кожна директива містить передані аргументи. Написання відбувається в

стандартному середовищі bash мікрокомп'ютера. На рисунку 4.4 показано процес написання конфігураційного файлу HTTP-сервера Apache.

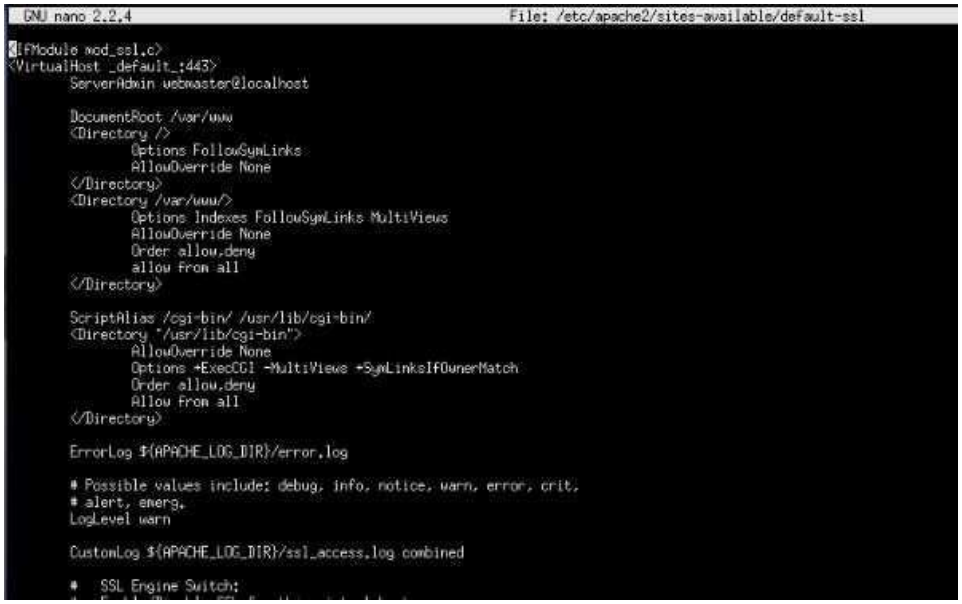


Рисунок 4.4 – Написання конфігураційного файлу HTTP-сервера Apache

В цілому було розроблено 8 конфігураційних файлів:

– Httpd.conf. Файл конфігурації сервера, містить основний технічний опис роботи домена;

– Srm.conf. Карта ресурсів сервера, вказує домену HTTPd порядок надання файлів;

– Access.conf. Файл конфігурації доступу містить інформацію про те, хто має право здійснювати доступ до вашого сервера;

– .htaccess. Файл додаткової конфігурації дозволяє налаштовувати функціонал для окремих каталогів;

– Apache2.conf. Конфігураційний файл сервера Apache;

– Php.ini. Файл конфігурації модуля PHP для сервера Apache;

– Vhost.conf. Файл конфігурації віртуальних хостів.

– Domain.tld. Файл конфігурації доменів.

4.3 Веб-сервер Apache

На момент написання дипломного проекту, в репозиторіях Debian, версія Apache відповідає 2.4.10-10 + deb8u8 вона і була встановлена і налаштована на мікрокомп'ютер. У разі успішної конфігурації HTTP-сервера при наборі IP-адреси пристрою в браузер вивантажується сторінка про успішне налаштування, показано на рисунку 4.5.

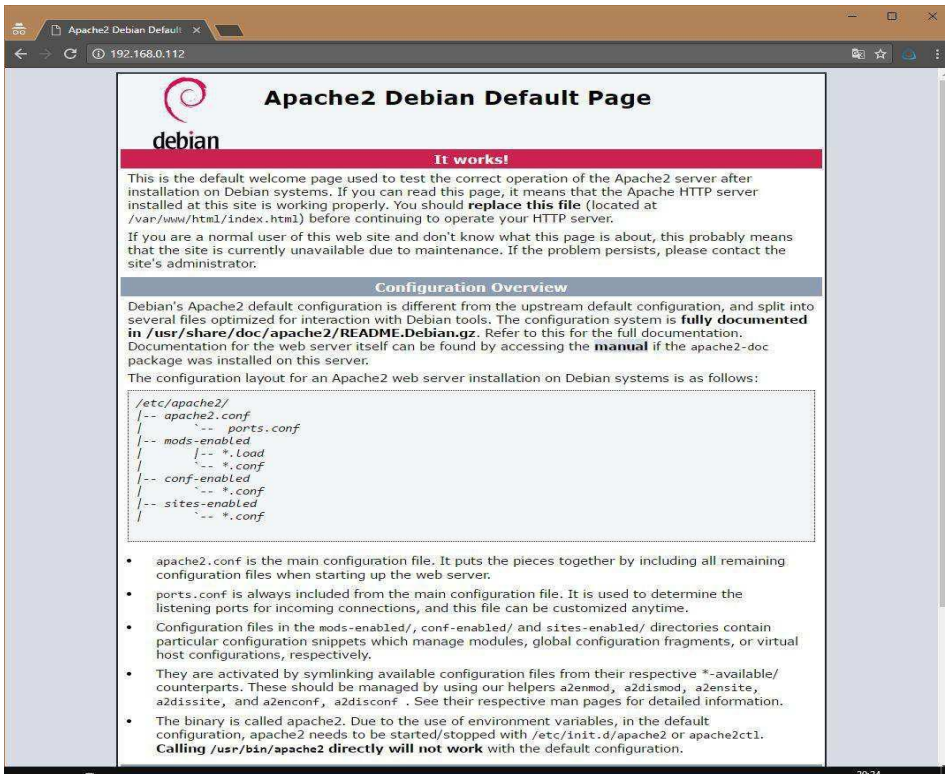


Рисунок 4.5 – Вікно встановленого веб-сервера

Наступним важливим рішенням було розмежування прав доступу до файлів і папок сайту. За замовчуванням власник файлів і папок є веб-сервер (користувач і група www-data), але зручніше і безпечніше запускати вміст сайтів від імені користувача, для цього був встановлений і налаштований пакет Apache-mpm-itk, що дозволяє виконувати розмежування прав, після цього скрипти і файли одного

віртуального хоста не були доступними для інших віртуальних хостів, що істотно підвищило безпеку сервера.

4.4 PHP

На момент написання дипломного проекту, в репозиторіях Debian, версія PHP відповідає 5.6.30-0 + deb8u1 вона і була встановлена і налаштована на мікрокомп'ютер. Встановлено як модуль для Apache, так як це трохи підвищує швидкодію, через те, що модуль завантажується один раз при запуску веб-сервера. Під час установки інтерпретатора, по замовчуванню були встановлені стандартні модулі. Але цього не вистачає для повноцінного функціонування сервера, тому для роботи з графікою був встановлений і налаштований модуль графічної бібліотеки GD2. У плюси бібліотеки можна записати низьке споживання ресурсів і високу швидкість роботи. На рисунку 4.6 наведено приклад працездатності модуля PHP, виведена стандартна сторінка з інформацією про PHP.

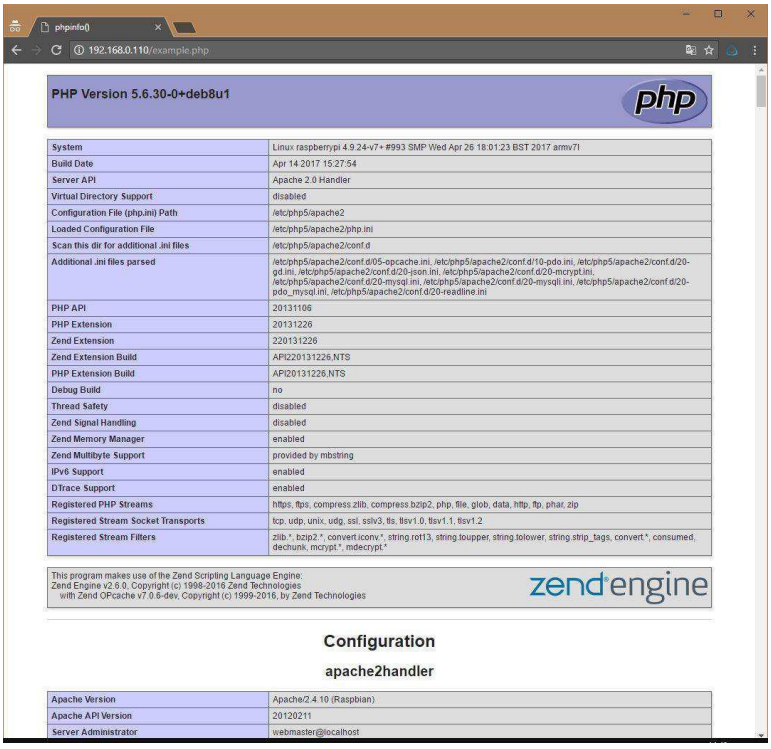


Рисунок 4.6 – Вікно встановленого інтерпретатора PHP

4.5 MySQL

На момент написання дипломного проекту, в репозиторіях Debian, версія MySQL відповідає 5.5.54-0 + deb8u1 вона і була встановлена на мікрокомп'ютер. СУБД за важливістю перевершує всі інші компоненти веб-сервера, втрата бази даних сайтів загрожує втраті всієї інформації ресурсу. Тому в процесі конфігурації був встановлений пароль суперкористувача MySQL (Root), процес установки показаний на рисунку 4.7 Так само для підвищення безпеки користувачеві Root була заблокована можливість підключатися з віддалених хостів.

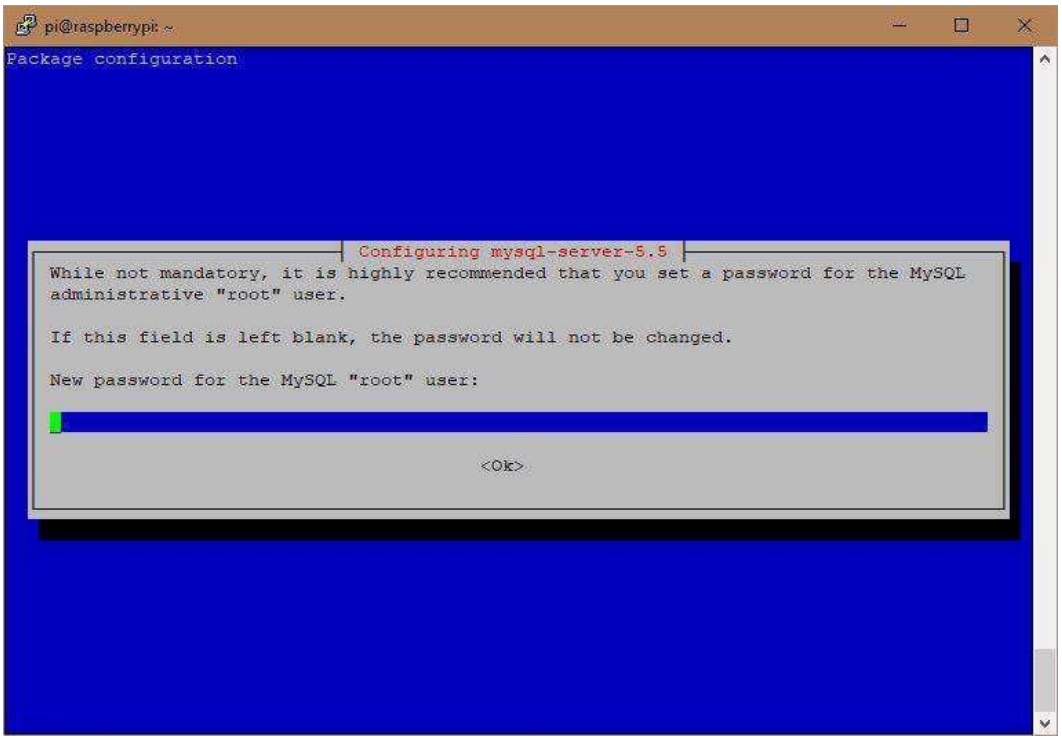


Рисунок 4.7 – Вікно встановлення паролю на СУБД MySQL

4.6 PhpMyAdmin

На момент написання дипломного проекту, в репозиторіях Debian, версія PhpMyAdmin відповідає 4: 4.2.12-2 + deb8u2 вона і була встановлена на мікрокомп'ютер. Працюючий модуль представлений на рисунку 4.8.

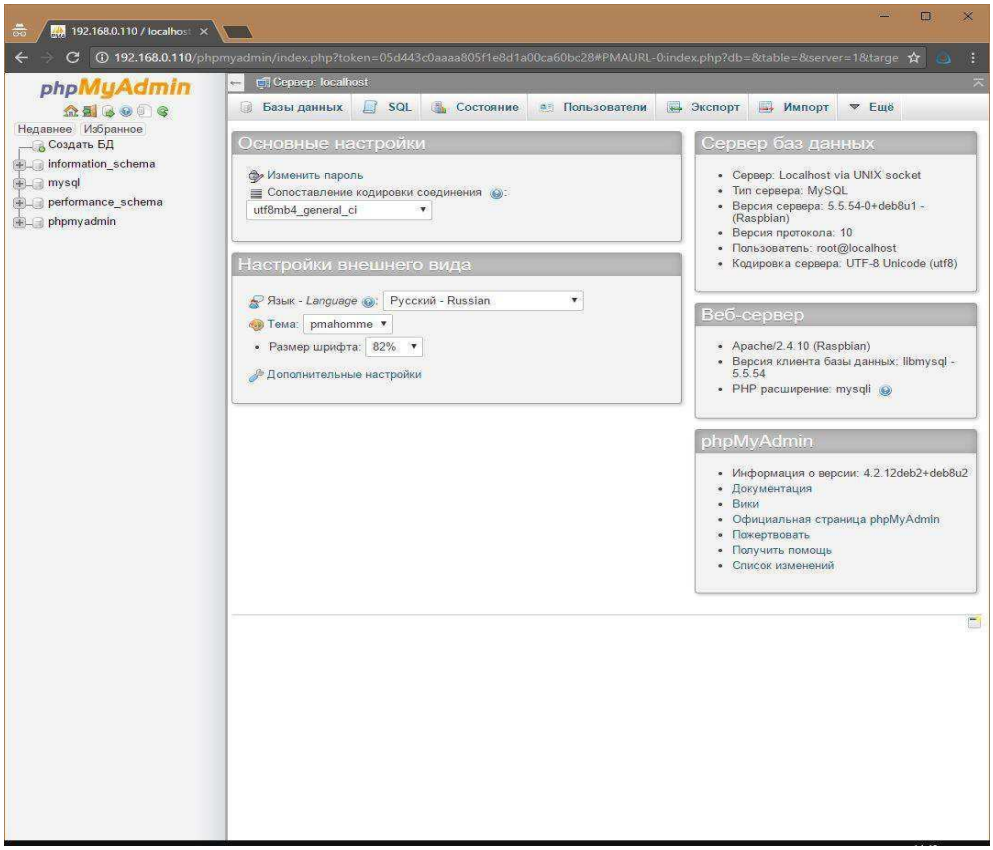


Рисунок 4.8 – Модуль PhpMyAdmin

З міркувань безпеки щоб забезпечити надійне управління базами даних було проведено конфігурацію авторизації на рівні Apache додавання авторизації на рівні веб-сервера.

На рисунку 4.9 показано авторизація на рівні веб-сервера, при зверненні до phpMyAdmin з'являється вікно із запитом імені користувача та пароля, після чого необхідно ввести дані користувача PhpMyAdmin.

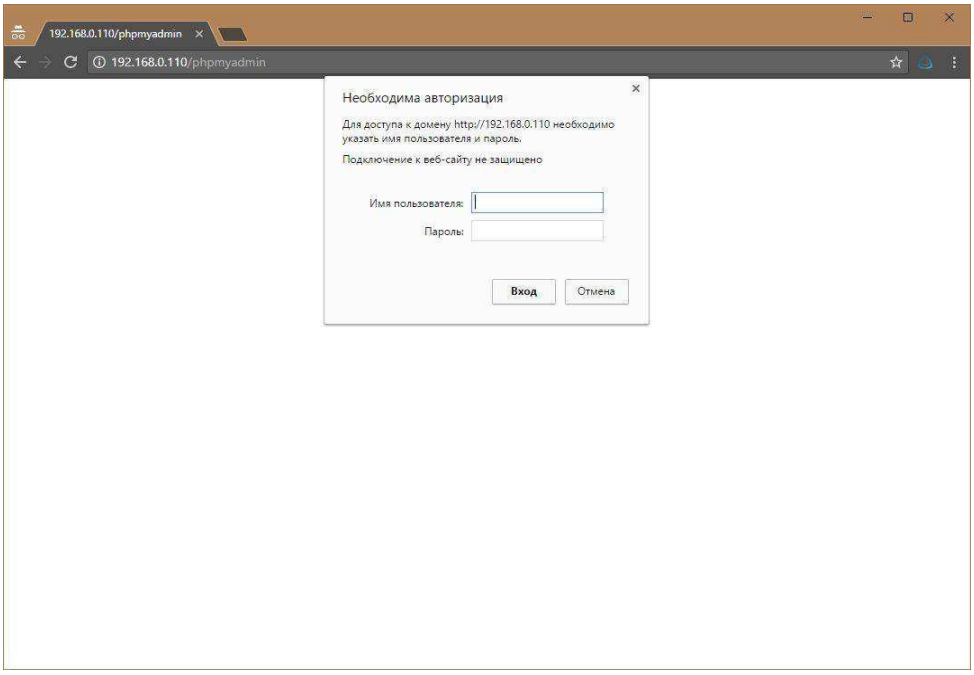


Рисунок 4.9 – Авторизація на рівні веб-сервера

4.7 VSFTPD

На момент написання дипломного проекту, в репозиторіях Debian, версія vsftpd відповідає 3.0.2-17 + deb8u1. З метою підвищення безпеки було встановлено заборону на підключення анонімних користувачів і включений параметр, коли всі системні користувачі будуть знаходитися тільки в межах chroot і не зможуть отримати доступ до інших серверів. На рисунку 4.10 представлений повністю налаштований FTP-сервер, проведена авторизація користувача, виведений список директорій з FTP-сервера. В якості користувача виступила стандартна консоль CMD, операційна система Windows 10.

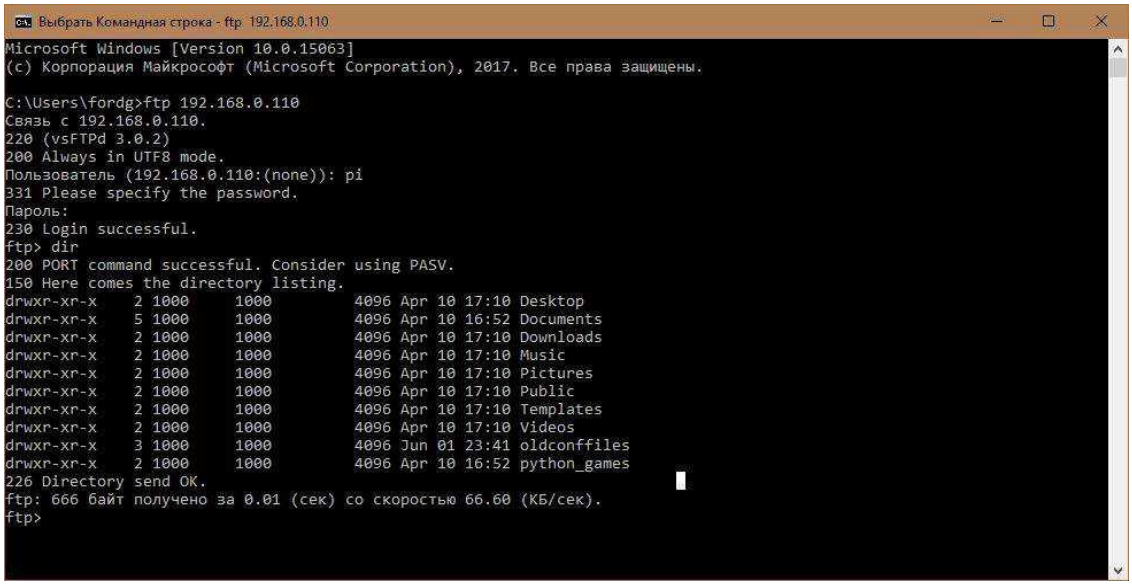


Рисунок 4.10 – Вікно налаштованого FTP-серверу

4.8 NextCloud

На момент написання диломного проекту актуальною і стабільною версією NextCloud була 26.0.1 і на її прикладі продемонстровано роботу хмари. На рисунку 4.11 продемонстровано вікно авторизації

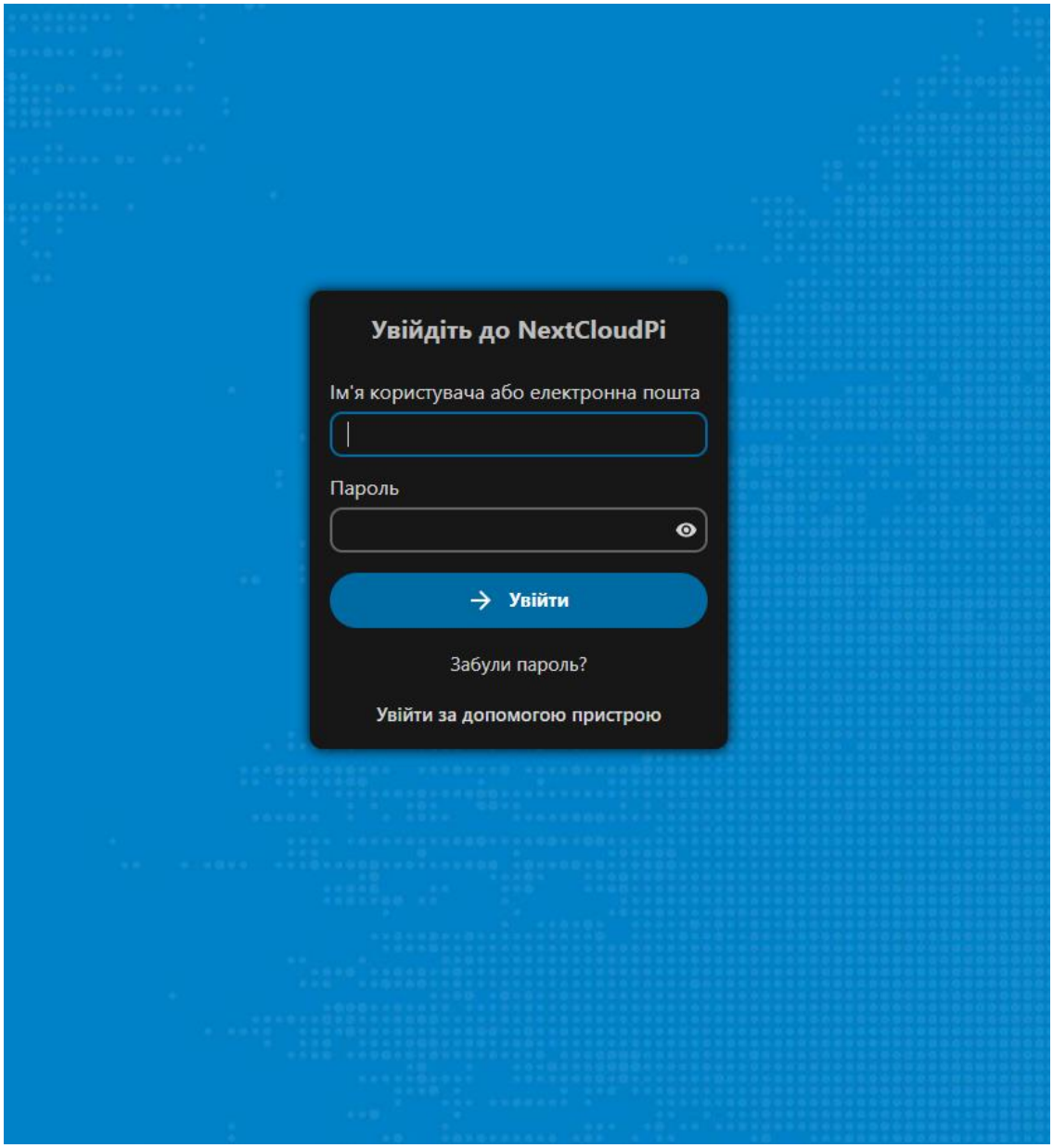


Рисунок 4.11 – Вікно авторизації в хмарі

На рисунку 4.12 продемонстроване привітальне вікно після встановлення хмари

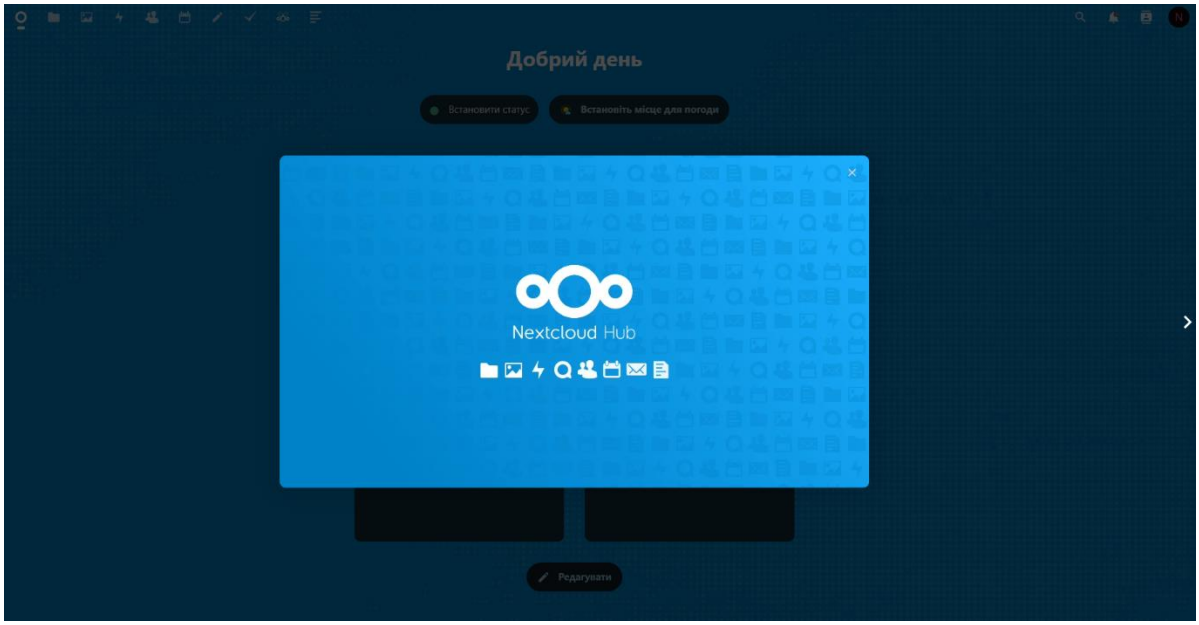


Рисунок 4.12 – Вікно початку роботи з мережевою хмарою

На рисунку 4.13 продемонстроване вікно перегляду доступних файлів

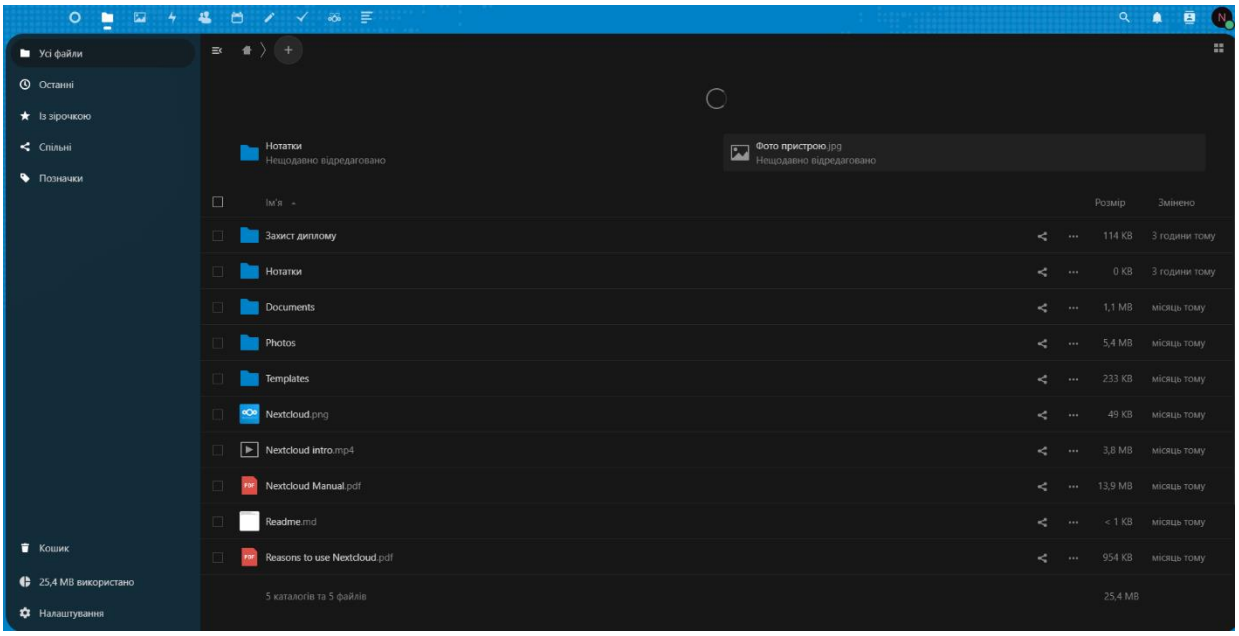


Рисунок 4.13 – Вікно перегляду усіх файлів

На рисунку 4.14 продемонстроване вікно кошику

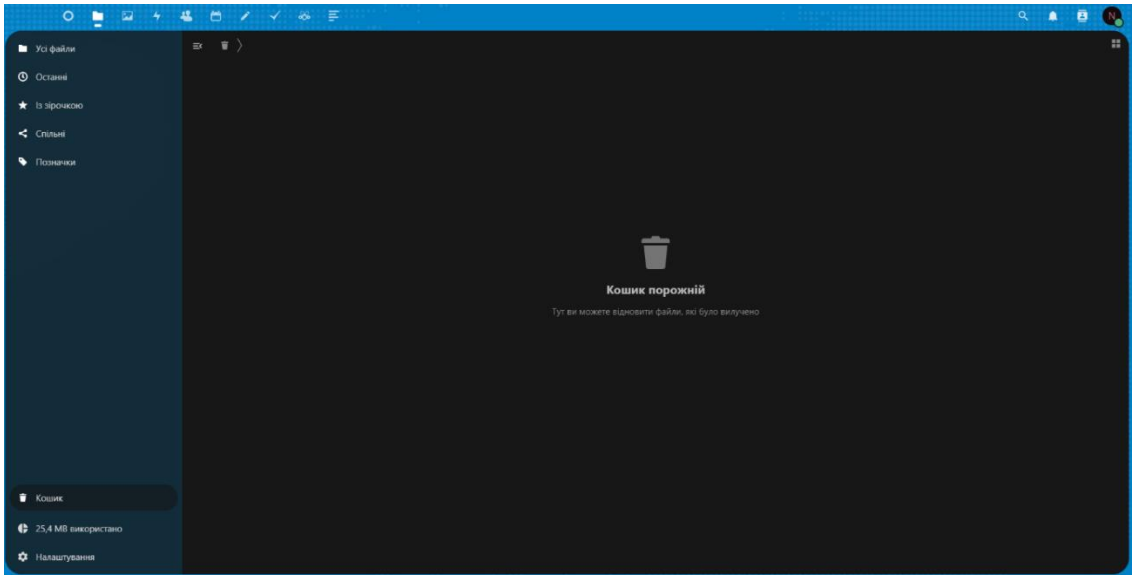


Рисунок 4.14 – Вікно кошику

На рисунку 4.15 продемонстроване вікно перегляду доступних зображень

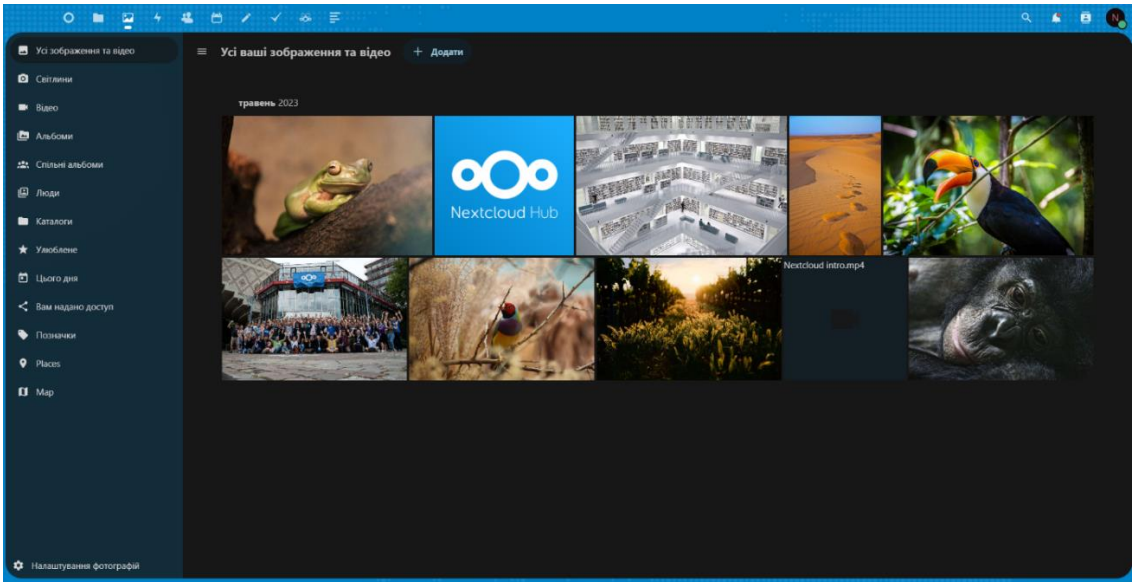


Рисунок 4.15 – Вікно перегляду зображень

На рисунку 4.16 продемонстроване вікно останніх подій на хмарі

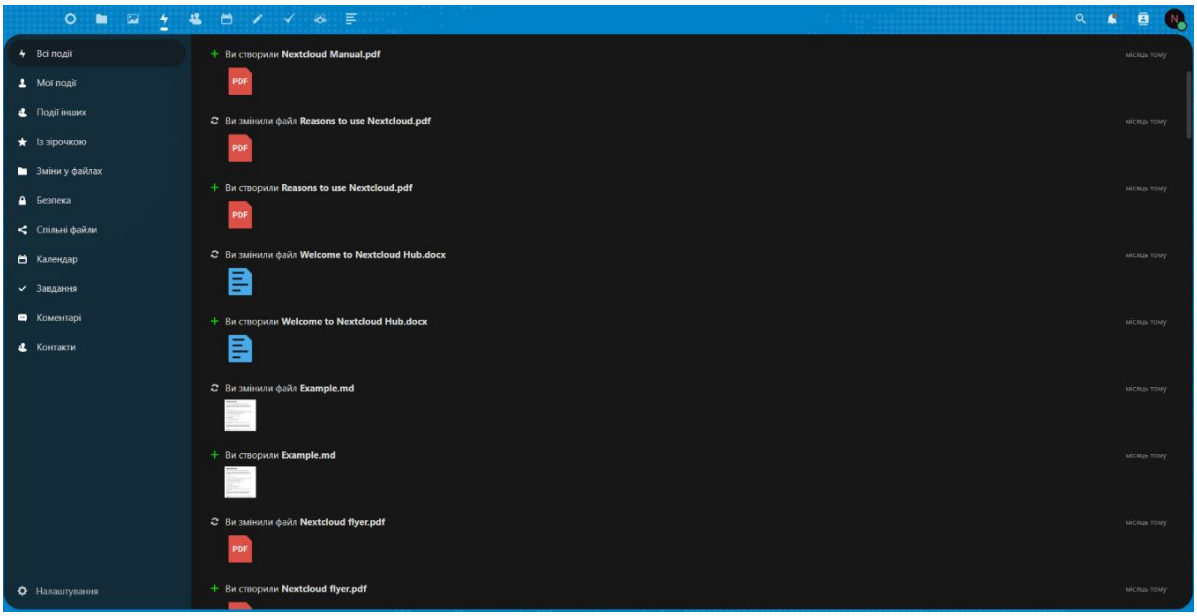


Рисунок 4.16 – Вікно перегляду останніх подій на хмарі

На рисунку 4.17 продемонстроване вікно перегляду контактів

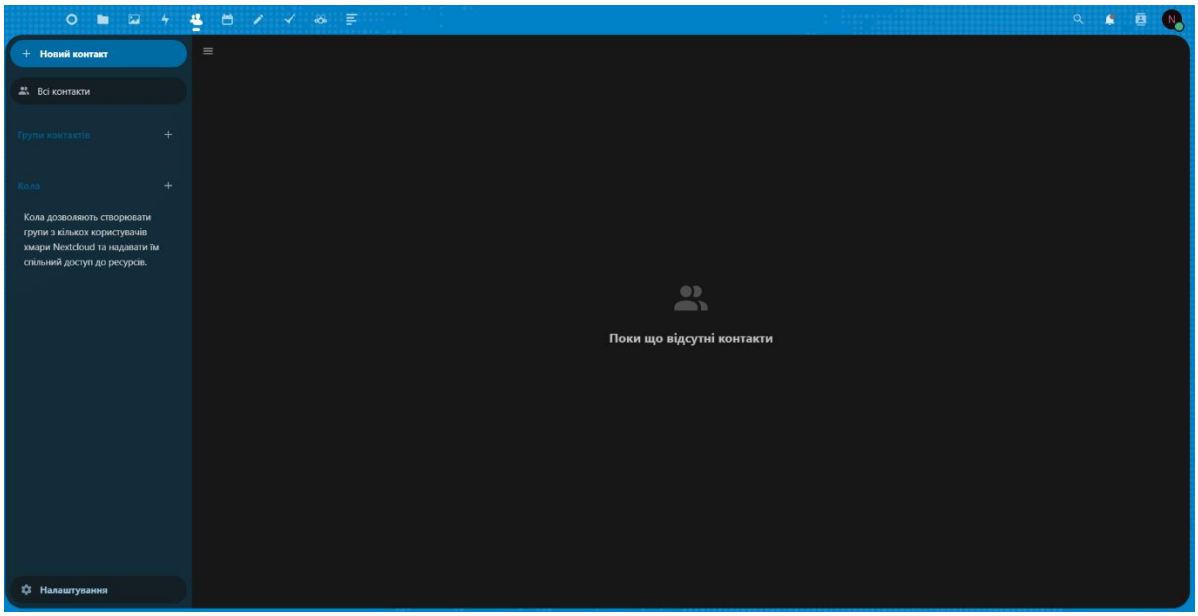


Рисунок 4.17 – Вікно перегляду контактів

На рисунку 4.18 продемонстроване вікно календаря

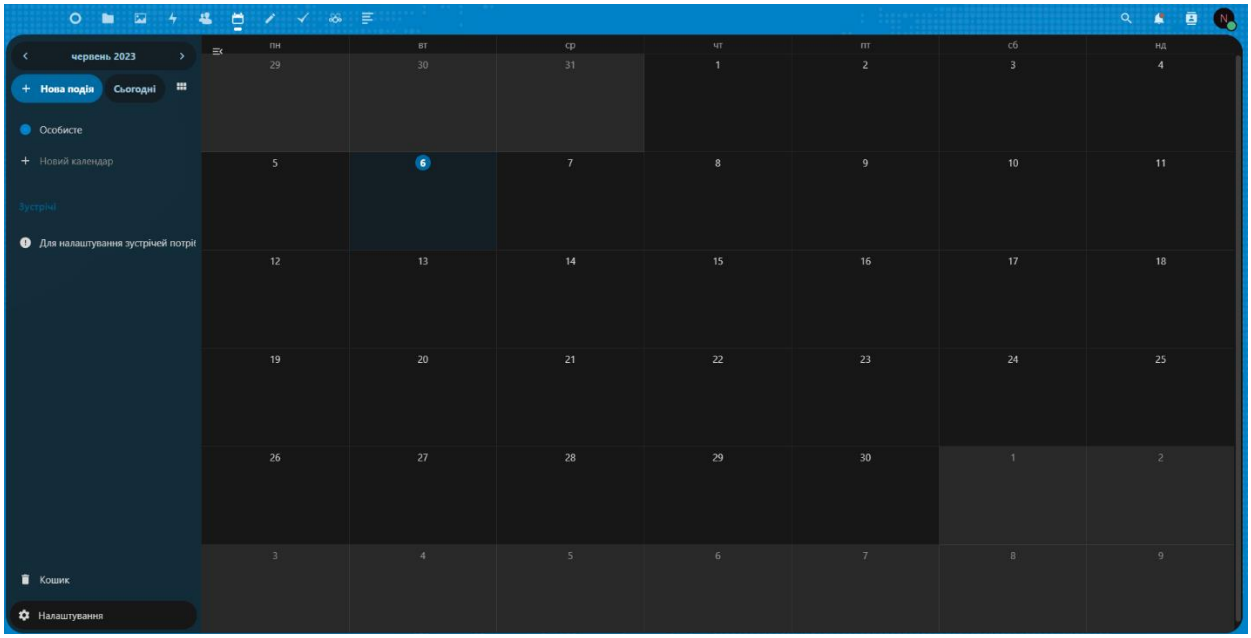


Рисунок 4.18 – Вікно вбудованого календаря

На рисунку 4.19 продемонстроване вікно нотатків

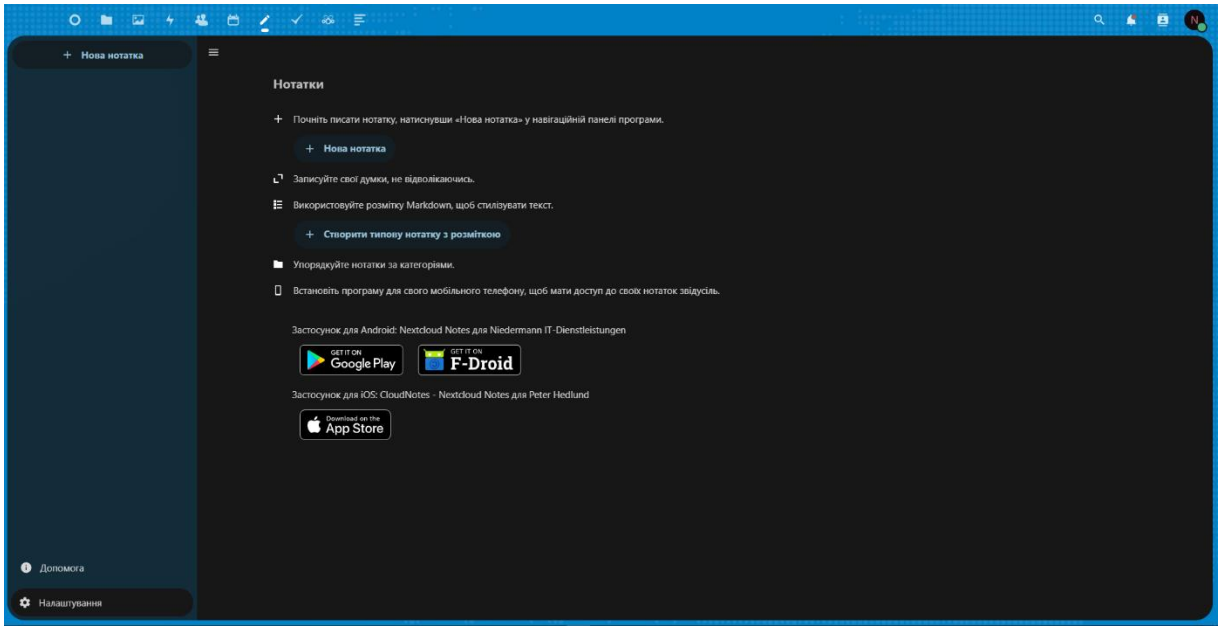


Рисунок 4.19 – Вікно нотатків

На рисунку 4.20 продемонстроване вікно менеджера завдань

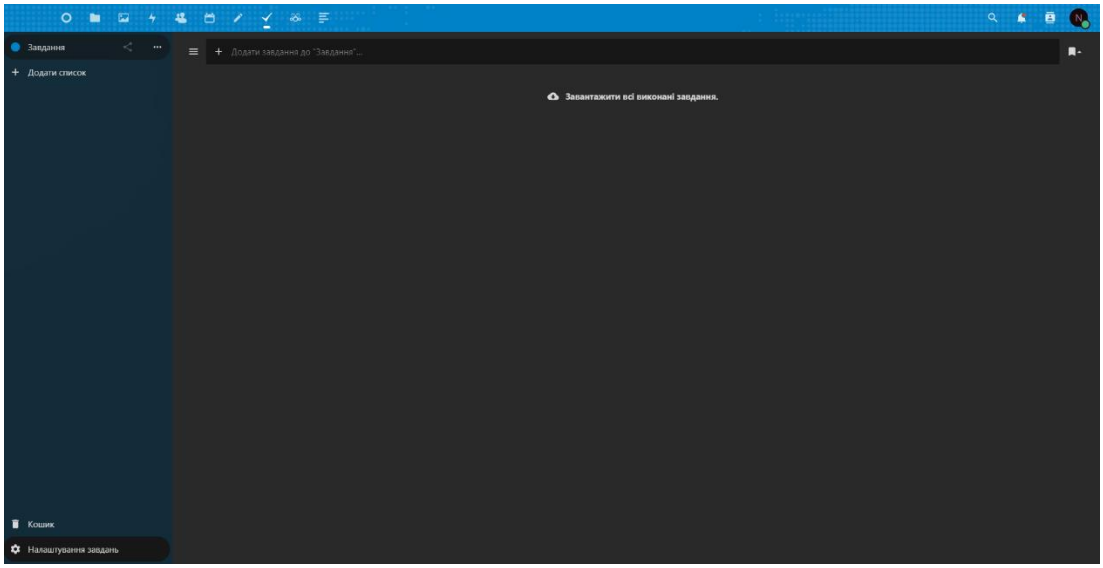


Рисунок 4.20 – Вікно менеджера завдань

На рисунку 4.21 продемонстроване вікно з новинами

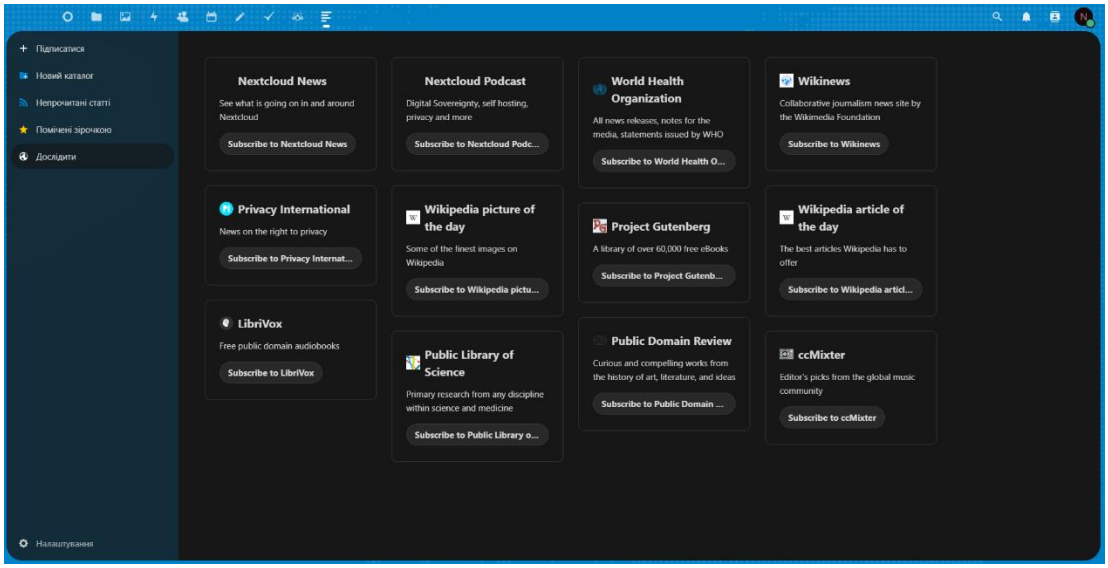


Рисунок 4.21 – Вікно з новинами

4.9 Тестування веб-сервера

Тестування проводилося утилітою ab - Apache HTTP server benchmarking tools.[38] Перша група тестів полягала в тому, щоб від клієнта відбувався запит html-файлу розміром 10,7 Кб. Було організовано навантаження на сервер в одну тисячу послідовних запитів. На рисунку 4.22 представлені результати виконання тесту.

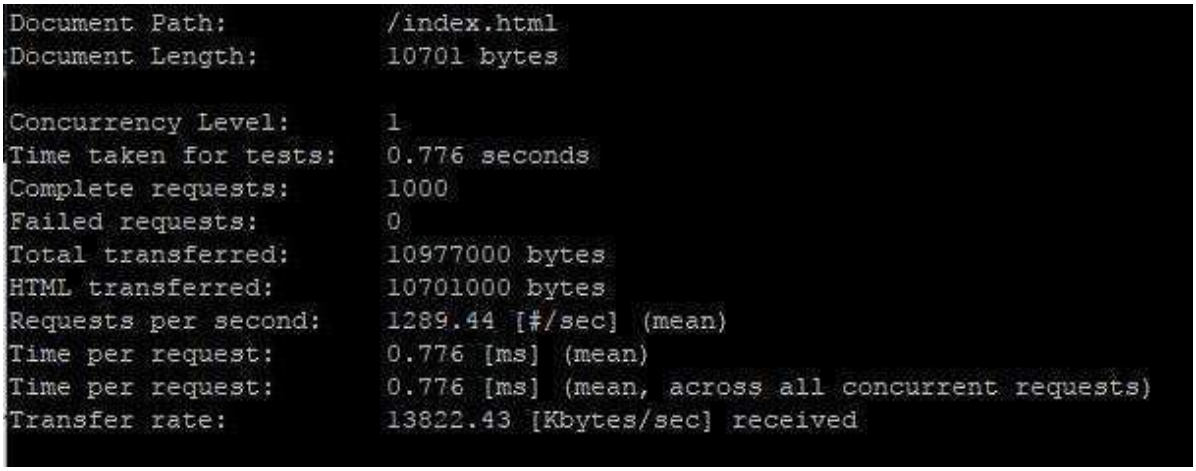


Рисунок 4.22 – Результат виконання 1000 запитів для 1 підключення

Як видно з рисунка 4.22 було виконано тисячу запитів в 1 потік, що еквівалентно 1 реальному користувачеві, докладніше:

– Concurrency Level: кількість одночасно-відправляючих запитів - 1;

– Time taken for tests: 1000 запитів до сервера зайняла 0,7 секунди;

– Complete requests: успішно отримана відповідь на всю 1000 запитів;

– Failed requests: невдалих запитів - нуль;

– Total transferred: загальний обсяг переданих даних: 10977000 байт;

– HTML transferred: з них HTML даних - 10701000 байт;

– Requests per second: середня кількість запитів в секунду склала 1289.5;

– Time per request: середній час на один запит 0,7 мілісекунди;

– Transfer rate: швидкість обміну даним з сервером склала 13822.4 кілобайтів за секунду.

Наступним тестом було проведення 1000 запитів, але вже від 500 користувачів, на рисунку 4.23 показаний результат виконання тесту.

```
Document Path:          /index.html
Document Length:        10701 bytes

Concurrency Level:      500
Time taken for tests:   20.426 seconds
Complete requests:      1000
Failed requests:        39
   (Connect: 0, Receive: 0, Length: 39, Exceptions: 0)
Total transferred:      10653097 bytes
HTML transferred:       10385101 bytes
Requests per second:    48.96 [#/sec] (mean)
Time per request:       10212.806 [ms] (mean)
Time per request:       20.426 [ms] (mean, across all concurrent requests)
Transfer rate:          509.33 [Kbytes/sec] received
```

Рисунок 4.23 – Результат виконання 1000 запитів для 500 підключень

Як видно що, 500 одночасних підключень сервер витримати не може, рядок Failed requests сигналізує нам про це, кажучи, що 39 запитів не було оброблено, значить клієнтові не був виданий будь-який контент. Тож було Встановлено, що сервер може обробити одночасно 450 відвідувачів, підтвердженням цьому є рисунок 4.24.

```
Document Path:          /index.html
Document Length:        10701 bytes

Concurrency Level:      450
Time taken for tests:   0.913 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      10977000 bytes
HTML transferred:       10701000 bytes
Requests per second:    1095.21 [#/sec] (mean)
Time per request:       410.880 [ms] (mean)
Time per request:       0.913 [ms] (mean, across all concurrent requests)
Transfer rate:          11740.35 [Kbytes/sec] received
```

Рисунок 4.24 – Результат виконання 1000 запитів для 450 підключень

Але це були тести сторінки розміром 10,7 КБ, наступним тестом була «Складна» включаючи в себе безліч динамічних об'єктів сторінка розміром 190,639

КБ, що знаходиться на іншому сервері. Тобто для початку веб-сервер завантажив всю необхідну інформацію собі, обробив і тільки потім вже видав клієнту. На рисунку 4.25 показані результати виконання 1000 запитів «Складна» динамічна сторінка.

```
Server Software:        nginx/0.8.53
Server Hostname:        www.sfu-kras.ru
Server Port:            80

Document Path:          /
Document Length:        190639 bytes

Concurrency Level:      1
Time taken for tests:   384.883 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      191125000 bytes
HTML transferred:       190639000 bytes
Requests per second:    2.60 [#/sec] (mean)
Time per request:       384.883 [ms] (mean)
Time per request:       384.883 [ms] (mean, across all concurrent requests)
Transfer rate:          484.94 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        2    7   9.1      5     117
Processing:   295  378  70.7    361     982
Waiting:      204  253  35.6    245     527
Total:        298  385  72.8    369     986
```

Рисунок 4.25 – Результат виконання 1000 запитів для 1 підключення динамічної сторінки обробленої з іншого сервера

Заключним тестом з використанням HTML-сторінки було тестування «складної» динамічної сторінки, по знаходженню кількості оброблених одночасних підключень. На рисунку 4.26 показано що веб-сервер здатний обробити безпомилково 9 одночасних підключень, динамічної сторінки розміром 190,639 КБ, що знаходиться на іншому сервері.

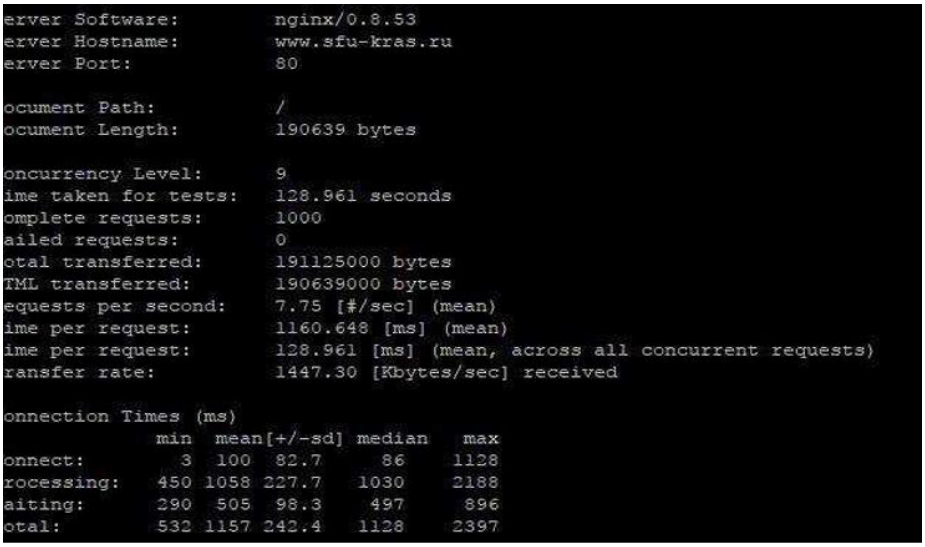


Рисунок 4.26 – Результат виконання 1000 запитів для 9 підключень динамічної сторінки обробленої з іншого сервера

Підсумковим тестом був тест на вимогу PHP-сценарію, тут веб-серверу необхідно не просто віддати файл, а запустити інтерпретатор, дочекатися від нього виведення і надати цей результат клієнту. На рисунку 4.27 показаний результат виконання 1000 запитів від 50 одночасних підключень.

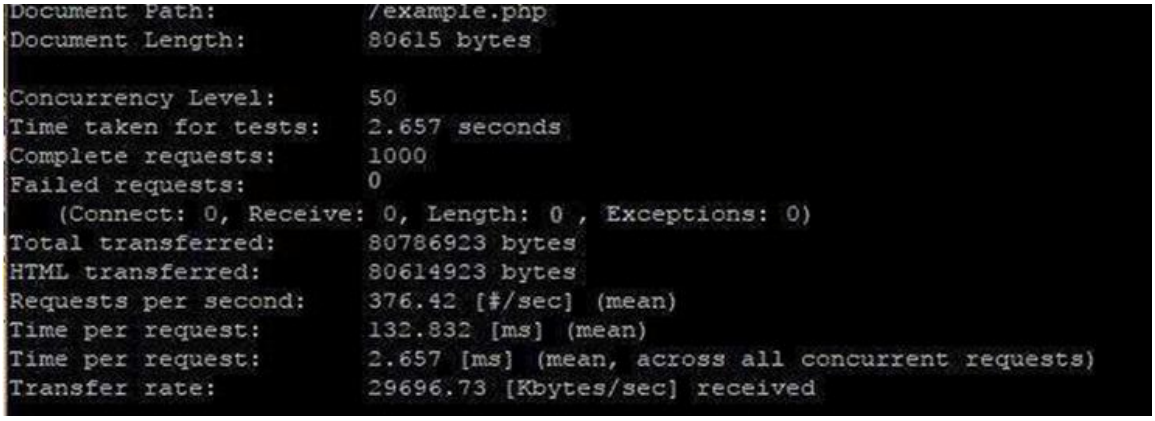


Рисунок 4.27 – Результат виконання 1000 запитів від 50 одночасних підключень

Як результат, (швидкість сховищ даних залежна від швидкості запису і читання карти MicroSD), є цілком очевидний висновок - повноцінний веб-сервер на Raspberry Pi 3B+ реальний. Як по програмній частині, так і за технічними

параметрами. Багатопоточність процесора допомагає оперативно справлятися із запитами, пам'яті цілком достатньо для кешування, веб-сервер може спокійно обслуговувати пару трійку тисяч відвідувачів сайтів в день.

РОЗДІЛ 5 ЕКОНОМІЧНИЙ АНАЛІЗ ВИТРАТ ТА ОБСЛУГОВУВАННЯ

5.1 Розрахунок вартості обладнання

Для того, щоб виконати розрахунки експлуатаційних витрат для організації хмари необхідно визначити загальну вартість основних фондів, що використовуються. Загальна вартість обладнання розрахована в таблиці 2.

Таблиця 2 – Перелік обладнання для організації хмари

№ п/п	Найменування обладнання	Одиниця виміру	Кількість	Ціна, грн.
1	Raspberry Pi 3 Model B+	шт.	1	1345,0
2	Зовнішній жорсткий диск	шт.	1	1409,0
Всього, грн.				2754,0

5.2 Розрахунок балансу робочого часу та чисельності персоналу

При проведенні розрахунків балансу робочого часу та чисельності персоналу слід враховувати, що персонал, який буде обслуговувати мережеве облоднання буде працювати 4 годин в першій половині дня. Також при розрахунку балансу робочого часу слід врахувати, що кількість календарних днів становить 365, кількість вихідних і святкових днів за рік становить 114, номінальний фонд робочого часу становить 251 день.

Розрахунок ефективного фонду робочого часу виконуємо за формулою:

$$F_{еф}=(H_{фрч}-H)\cdot n_{зм}\cdot T_{зм}, \quad (8)$$

де $F_{еф}$ – ефективний фонд часу;

$H_{фрч}$ – номіальний фонд часу;

H – невиходи на роботу;

$n_{зм}$ – кількість змін (приймаємо 1 зміну);

$T_{зм}$ – тривалість робочого дня.

$F_{еф} = 251\cdot1\cdot4 = 1004$ год.

В таблиці 2 приведено основні показники балансу робочого часу.

Таблиця 3 – Баланс робочого часу

№ п/п	Назва	Значення
1	Кількість календарних днів	365 днів
2	Вихідні та святкові дні	114 днів
3	Номінальний фонд робочого часу	251 день
4	Тривалість робочого дня	4 год
5	Ефективний фонд часу	1004 год

Визначаємо чисельність персоналу $Ч_п$ за формулою:

$$Ч_п = \frac{О_р \cdot n_{зм}}{Н_{обл}}, \quad (9)$$

де $О_р$ – кількість комп'ютерів;

$n_{зм}$ – кількість змін;

$Н_{обсл}$ – норма обслуговування (нормативна кількість обладнання становить 1 комп'ютер на 1 працівника).

5.3 Розрахунок експлуатаційних витрат

Основною статтею експлуатаційних витрат є витрати на заробітну плату.

З 1 січня 2016 року Законом України «Про внесення змін до Податкового кодексу України та деяких законодавчих актів України щодо забезпечення збалансованості бюджетних надходжень у 2016 році» від 24.12.2015 р. № 909-VIII внесено зміни до Закону України «Про збір та облік єдиного внеску на загальнообов'язкове державне соціальне страхування» від 08.07.2010 р. № 2464-VI. Тому з 1 січня 2016 року підприємство сплачує ЄСВ за ставкою 22%, і скасовується єдиний соціальний внесок, що утримується із заробітної плати (доходу) працівників. А також внесені зміни, передбачені Законом України від 24 грудня 2015 р. № 909-VIII «Про внесення змін до Податкового кодексу України та деяких законодавчих актів України щодо забезпечення збалансованості бюджетних надходжень у 2016 році» на податок на доходи фізичних осіб, тепер встановлено єдину базову ставку у розмірі 18%.

Водночас Законом України від 28 грудня 2014 року № 71-VIII «Про внесення змін до Податкового кодексу України та деяких законодавчих актів України щодо податкової реформи» (далі Закон № 71), який набрав чинності з 01.01.2015, оподаткування військовим збором продовжено до набрання чинності рішенням

Верховної Ради України про завершення реформи Збройних Сил України (пункт 161 підрозділ 10 розділу XX Перехідних положень ПКУ).

Платниками збору є особи, визначені пунктом 162.1 статті 162 цього ПКУ (підпункт 1.1 пункт 161 підрозділу 10 ПКУ), а саме:

— фізична особа - резидент, яка отримує доходи як з джерела їх походження в Україні, так і іноземні доходи;

— фізична особа - нерезидент, яка отримує доходи з джерела їх походження в Україні;

— податковий агент.

Військой збір сплачуєть у розмірі 1,5%.

Для обслуговування комутатора Cisco 2960 передбачено посаду системного адміністратора. Ставка системного адміністратора – 7500 грн.

Отже, визначаємо річний фонд заробітної плати лаборанта, що працює неповну ставку, за формулою:

$$Ф_{річ}= Ч \cdot ТС \cdot 12 \quad (10)$$

Визначаємо річний фонд заробітної плати мережевого адміністратора, що працює повну ставку:

$$ЗП=1 \cdot 7500 \cdot 12=90000 \text{ грн.}$$

Податок на доходи фізичних осіб та військовий збір розраховуємо за формулою:

$$В=ЗП \cdot 18\% \cdot 1,5\% \quad (11)$$

Отже, визначаємо річні відрахування із заробітної плати для працівника:

$$В = 90000 \cdot (0,18+0,015)=17550,0 \text{ грн.}$$

Визначаємо суму амортизаційних відрахувань основних фондів.

Основні фонди – це засоби праці, що діють протягом тривалого терміну служби і свою вартість на вартість випущеної продукції переносять частково по мірі зносу у вигляді амортизаційних відрахувань.

Отже, з переліченого вище обладнання до основних фондів відносимо робочу станцію та Raspberry Pi 3 Model B+.

Визначаємо суму амортизаційних відрахувань за формулою:

$$А_м = \frac{В_{об} \cdot Н_а}{100}, \quad (12)$$

де $В_{об}$ – вартість обладнання;

$Н_а$ – норма амортизаційних відрахувань (становить 25%).

$$А_м = \frac{6345{,}0 \cdot 25}{100} = 1586{,}25 \text{ грн.}$$

Щоб визначити витрати на електроенергію, слід розрахувати потужність робочої станції та мережевого обладнання.

Визначимо споживчу потужність обладнання першого поверху за формулою:

$$N_{уст} = Р_{пк} \cdot О_р + Р_д \cdot О_р, \quad (13)$$

де $Р_{пк}$ – потужність ПК;

$Р_д$– потужність допоміжного обладнання;

$О_р$ – кількість одиниць.

Потужність одного ПК становить 500 Вт, а потужність допоміжного обладнання – 5,5 Вт.

Встановлена потужність мережі дорівнює

$$N_{уст} = 500 \cdot 1 + 5{,}5 \cdot 1 = 545 \text{ Вт} = 0{,}505 \text{ кВт}$$

Одиницею виміру кількості спожитої енергії є ват-година (Вт·год), але при розрахунку використовується одиниця виміру кіловат-години (кВт·год) . Одна кВт·год коштує 1,68 грн.

Отже, визначаємо річні витрати на електроенергію за формулою:

$$В_е = Ц_{кв} \cdot F_{еф} \cdot П, \quad (14)$$

де $Ц_{кв}$ – ціна одного кВт/год, грн;

П – потужність обладнання, кВт.

$В_е$ =1,68 · 1004 · 0,505 = 851,79 грн.

Затрати на поточний ремонт за рік складають 5% від вартості обладнання:

$$В_{пр} = В_{об} \cdot 0,05, \quad (15)$$

де $В_{об}$ – вартість обладнання, грн.

$В_{пр}$ = 6345· 0,05 = 317,25 грн.

Сума річних експлуатаційних витрат на обслуговування мережі та витрати на обладнання представлена в таблиці 4.

Таблиця 4 – Витрати річні експлуатаційні на обслуговування мережі та на обладнання

№ п/п	Статті витрат	Сума, грн.
1	Річні амортизаційні відрахування	1586,25
2	Річні витрати на електроенергію	851,79
3	Річний фонд заробітної плати	90000,0
4	Відрахування на соціальні потреби працівників	17550,0
5	Витрати на поточний ремонт обладнання	317,25
6	Всього	110305,29

Розрахунок загальнозаводських витрат (витрати на загальне обслуговування та організацію локальної мережі) становить 55% від заробітної плати основних працівників:

$$В_{зз} = ЗП \cdot 55\%, \quad (16)$$

де $В_{зз}$ – витрати загальнозаводські, грн.

$$В_{зз} = 90000,0 \cdot 0,55 = 49500,0 \text{ грн.}$$

Виробнича собівартість включає в себе поточні витрати і загальнозаводські витрати:

$$С_{вир} = В_{п} + В_{зз}, \quad (17)$$

де $С_{вир}$ – собівартість виробнича, грн.;

$В_{п}$ – витрати поточні, грн.

$С_{вир} = 110305,29 + 49500,0 = 159805,29$ грн.

Позавиробничі витрати становлять 13% від виробничої собівартості:

$$В_{пв} = С_{вир} \cdot 13\%, \quad (18)$$

де $В_{пв}$ – витрати позавиробничі, грн.

$В_{пв} = 159805,29 \cdot 0,13 = 20774,68$ грн.

Повна (проектована) собівартість визначається як сума виробничої собівартості та позавиробничих витрат:

$$С_{п} = С_{вир} + В_{пв}, \quad (19)$$

де $С_{п}$ – собівартість повна, грн.

$С_{п} = 159805,29+ 20774,68 = 180579,97$ грн.

Отже, повна собівартість на утримання та обслуговування робочої станції та мережевого обладнання 180579,97 грн.

ВИСНОВКИ

Отже, проект зі створення індивідуального програмно-апаратного рішення на базі мікрокомп'ютера Raspberry Pi 3B+ для використання в екосистемі розумного будинку досягнув своїх основних цілей. Завдяки ретельному підбору ефективного програмного стеку, налаштуванню операційної системи, розробці конфігураційних файлів та встановленню програмного стеку було створено функціональне та зручне хмарне сховище даних.

Протягом роботи над даною темою я глибоко занурився у розуміння різних ролей серверів, їхньої функціональності та практичного застосування в сучасній IT-інфраструктурі. Це дослідження дало змогу побачити не лише технічний бік роботи серверів, а й їхню ключову роль у забезпеченні стабільності, продуктивності та безпеки всієї системи.

Я вважаю, що правильний підхід до вибору, конфігурації та використання серверних ролей є основоположним фактором для побудови надійної та ефективної мережі. Ця робота дала мені змогу не лише систематизувати наявні знання, але й глибше усвідомити важливість кожної ролі в загальній архітектурі сучасних цифрових систем.

Значення цього проекту полягає у визнанні зростаючого значення хмарних технологій у забезпеченні зручного віртуального середовища для обробки та зберігання інформації. Завдяки використанню можливостей мікрокомп'ютера Raspberry Pi 3B+, цей проект пропонує доступне рішення для приватних осіб, малого бізнесу та організацій для використання переваг хмарних сховищ.

Успішне завершення цього проекту підкреслює потенціал використання недорогого обладнання, такого як Raspberry Pi, для розробки практичних та ефективних рішень, пристосованих до конкретних потреб. В умовах зростаючої залежності від хмарних сервісів розроблене програмно-апаратне рішення пропонує користувачам інноваційний підхід до безпечного зберігання та доступу до своїх даних.

Загалом, цей проект демонструє адаптивність та універсальність мікрокомп'ютера Raspberry Pi 3B+ у створенні індивідуальних рішень, що відповідають вимогам сучасного цифрового середовища.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What are IoT Devices and How Do They Communicate? [Електронний ресурс] — Режим доступу: https://www.airtel.in/blog/business/what-are-iot-devices-and-how-do-they-communicate/
2. The history of the Internet of Things [Електронний ресурс] — Режим доступу: https://www.britannica.com/science/Internet-of-Things
3. Architecture of Internet of Things (IoT) [Електронний ресурс] — Режим доступу:
4. 10 security problems of the IOT [Електронний ресурс] — Режим доступу: https://www.chakray.com/10-security-problems-internet-of-things/
5. Future of IoT Technology: 8 Trends for Businesses to Watch in 2023 [Електронний ресурс] — Режим доступу: https://mobidev.biz/blog/iot-technology-trends
6. IoT Update: FTC Settles with Smart Lock Manufacturer and Provides Guidance for IoT Companies [Електронний ресурс] — Режим доступу: https://www.insideprivacy.com/data-security/iot-update-ftc-settles-with-smart-lock-manufacturer-and-provides-guidance-for-iot-companies/
7. Europe's Internet of Things Policy [Електронний ресурс] — Режим доступу: https://digital-strategy.ec.europa.eu/en/policies/internet-things-policy
8. China’s Industrial Standards for the Internet of Things: What the Draft Guidelines Say [Електронний ресурс] — Режим доступу: https://www.china-briefing.com/news/china-internet-of-things-industrial-standards-draft-guidelines-released-5-major-standards/
9. Korea mandates installation of IoT measurement devices for small establishments emitting air pollutants [Електронний ресурс] — Режим доступу: https://enviliance.com/regions/east-asia/kr/report_7135
10. Japan's IoT Security Strategy: Break Into Devices [Електронний ресурс] — Режим доступу: https://www.bankinfosecurity.com/japans-iot-security-strategy-break-into-devices-a-11977

11. India: Internet Of Things – Indian Legal Perspective [Електронний ресурс] — Режим доступу: https://www.mondaq.com/india/privacy-protection/1036182/internet-of-things--indian-legal-perspective
12. Australia's IoT Cybersecurity Guidelines Highlight Importance of Embedded Device Security [Електронний ресурс] — Режим доступу: https://sectigo.com/resource-library/australias-iot-cybersecurity-guidelines-highlight-importance-of-embedded-device-security
13. Internet of Things (IoT) - statistics & facts [Електронний ресурс] — Режим доступу: https://www.statista.com/topics/2637/internet-of-things/#topicOverview
14. Клієнт-серверна архітектура та ролі серверів. [Електронний ресурс] — Режим доступу: https://medium.com/@IvanZmerzlyi/клієнт-серверна-архітектура-та-ролі-серверів-9893d8048229
15. Архітектура клієнт-сервер. Сховища даних. Реляційні бази даних як основа сховищ даних [Електронний ресурс] — Режим доступу: http://educational.mariroz.com/InformTechVInfrastrRynku/lect/lect8.pdf
16. Клієнт-серверна архітектура [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура
17. Transmission Control Protocol [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/TCP
18. HyperText Transfer Protocol [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/HTTP
19. File Transfer Protocol [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/FTP
20. Simple Mail Transfer Protocol [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/SMTP
21. Internet Message Access Protocol [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/IMAP
22. Nextcloud [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/Nextcloud

23. Nextcloud news [Електронний ресурс] — Режим доступу: https://nextcloud.com/blog/
24. Документація Raspberry Pi [Електронний ресурс] — Режим доступу: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/
25. Документація Raspbian [Електронний ресурс] — Режим доступу: http://www.raspbian.org/
26. Linux веб-сервери [Електронний ресурс] — Режим доступу: https://helpiks.org/8-59867.html
27. Apache MPM worker [Електронний ресурс] — Режим доступу: https://httpd.apache.org/docs/2.4/mod/worker.html
28. Apache MPM prefork [Електронний ресурс] — Режим доступу: https://httpd.apache.org/docs/2.4/mod/prefork.html
29. Apache MPM Common Directives [Електронний ресурс] — Режим доступу: https://httpd.apache.org/docs/2.4/mod/mpm_common.html
30. Apache MPM netware [Електронний ресурс] — Режим доступу: https://httpd.apache.org/docs/2.4/mod/mpm_netware.html
31. Apache MPM winnt [Електронний ресурс] — Режим доступу: https://httpd.apache.org/docs/2.4/mod/mpm_winnt.html
32. Документація PHP [Електронний ресурс] — Режим доступу: https://www.php.net/docs.php
33. MySQL [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/MySQL
34. Документація MySQL [Електронний ресурс] — Режим доступу: https://dev.mysql.com/doc/
35. phpMyAdmin [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/PhpMyAdmin
36. Документація phpMyAdmin [Електронний ресурс] — Режим доступу: https://www.phpmyadmin.net/docs/
37. PuTTY [Електронний ресурс] — Режим доступу: https://uk.wikipedia.org/wiki/PuTTY

38. ab - Apache HTTP server benchmarking tool [Електронний ресурс] — Режим доступу: https://httpd.apache.org/docs/2.4/programs/ab.html

ДОДАТОК 1

Конфігураційні файли

Конфігураційний файл 000-default.conf (конфігурація Apache за замовчуванням)

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
```

```
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Конфігураційний файл nextcloud.conf

```
<VirtualHost *:80>
  DocumentRoot /var/www/nextcloud/
  <Directory /var/www/nextcloud/>
    Require all granted
    AllowOverride All
    Options FollowSymLinks MultiViews

    <IfModule mod_dav.c>
      Dav off
    </IfModule>
  </Directory>
</VirtualHost>
```

ДОДАТОК 2

Список необхідних модулів для коректної роботи NextCloud

apache2

php8.1

php8.1-gd

php8.1-curl

php8.1-common

php8.1-intl

php-pear

php-apcu

php8.1-xml

libapache2-mod-php8.1

php8.1-mbstring

php8.1-zip

php8.-mysql

mariadb-server-10.3

smbclient

imagemagick

php-imagick

ДОДАТОК 2

Інформаційні слайди

Київський Національний Університет
Будівництва і Архітектури

Кафедра інформаційних технологій

Кваліфікаційна випускна
робота магістра

на тему: Інформаційні технології для налаштування систем розумового приватного будинку

Виконав: студент групи КНм – 23
Мусієнко В.В.
Керівник: асистент Вацкель В.Ю.

Київ – 2024 рік

Вступ

Концепція IoT (Інтернет речей) в розумному будинку полягає в підключенні різних пристроїв і систем в будинку до мережі, що дозволяє їм спілкуватися і обмінюватися даними для поліпшення автоматизації, контролю і зручності. Технологія Інтернету речей у розумному будинку може перетворити традиційні будинки на інтелектуальні, взаємопов'язані простори, які надають розширені можливості та функції.

Актуальність теми

Актуальність теми моєї роботи полягає в тому, що мережеве зберігання даних відіграє вирішальну роль у середовищі розумного будинку. Оскільки розумні будинки все частіше включають в себе різні підключені пристрої і генерують великі обсяги даних, ефективні і надійні рішення для зберігання даних стають вкрай важливими.

Метою кваліфікаційної випускної роботи

є надання розробленого рішення у вигляді сконфігурованого програмно-апаратного комплексу на основі мікрокомп'ютера Raspberry Pi 3B+ для використання в екосистемі розумного будинку

Основні завдання:

-вибір ефективного програмного стека;

-установка і налаштування операційної системи;

-розробка конфігураційних файлів;

-установка програмного стека;

-проведення підсумкових тестів.

Клієнт-серверна архітектура

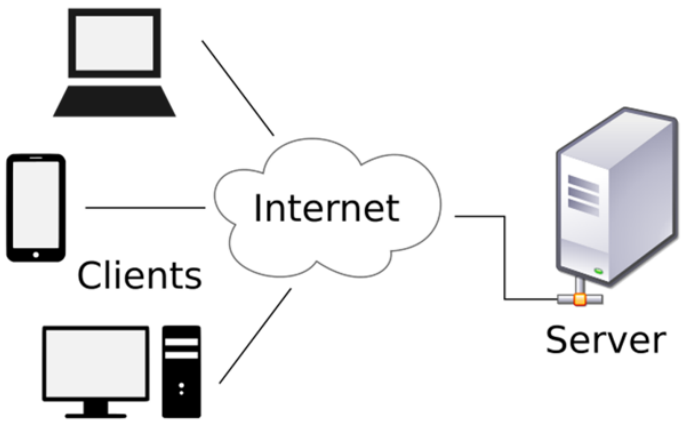
Клієнт-серверна архітектура набула своєї популярності завдяки динамічному розвитку мережі Інтернет та зосередження значної частини інформації в базах даних на серверах.

Клієнт-серверну архітектуру можна означити, як концепцію інформаційної мережі в якій основна частина її ресурсів зосереджена в серверах, обслуговуючих своїх клієнтів.

Така архітектура визначає такі типи компонентів:

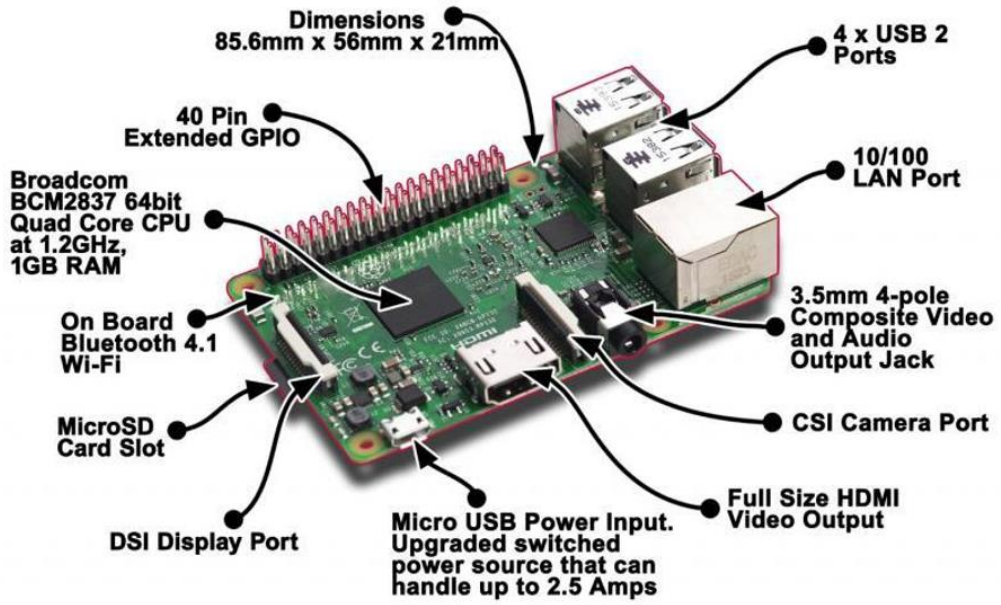
- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
- набір клієнтів, які використовують сервіси, що надаються серверами;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.

Структурна схема клієнт серверної архітектури

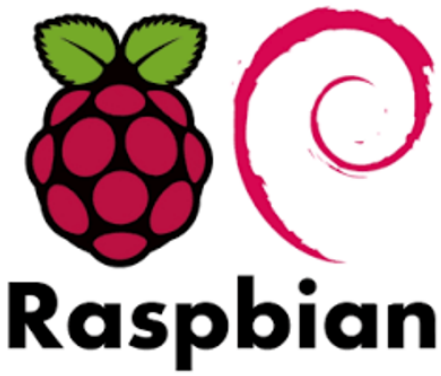


Основна ідея архітектури «клієнт-сервер» полягає в поділі мережевого додатка на кілька компонентів, кожен з яких реалізує специфічний набір сервісів. Компоненти такого додатку можуть виконуватися на різних комп'ютерах, виконуючи серверні або клієнтські функції. Це дозволяє підвищити надійність, безпеку і продуктивність мережевих додатків і мережі в цілому.

МІКРОКОМП'ЮТЕР RASPBERRY PI 3B+



Операційна система



Базовою операційною системою служить операційна система Raspbian від компанії Raspberry Pi Foundation, операційна система поширюється безкоштовно. Операційна система Raspbian входить в сімейство операційних систем Debian GNU/Linux і базується на версії Debian 8.7.1 Buster. Версії ядра системи – 4.4. Raspbian використовує Pixel , як основне оточення робочого столу, це видозмінена і модифікована версія іншого популярного робочого оточення – LXDE

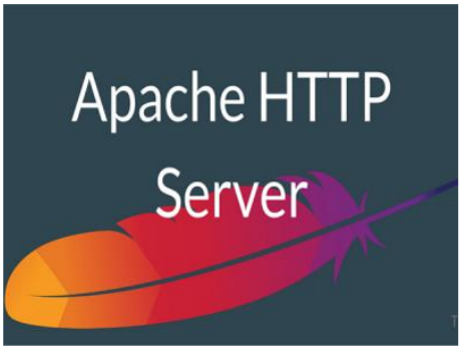
За версією журналу Tagline, ОС Debian GNU/Linux, очолює вершину топу і займає перше місце рейтингу серверних операційних систем. Операційна система Debian GNU/Linux має найбільшу серед всіх дистрибутивів Linux сховище пакетів, яке є репозиторієм готових програм і необхідних бібліотек.

Apache

В якості основого HTTP-сервера був обраний Apache HTTP-сервер.

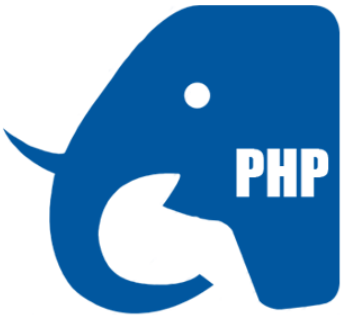
Apache належить до рангу кросплатформенного, який вільно поширює програмне забезпечення і підтримує такі операційні системи: Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS.

Розробкою і підтримкою сервера займається відкрите співтовариство розробників під контролем команії Apache Software Foundation. Apache HTTP Server входить до складу багатьох програмних продуктів.



PHP

Цей компонент Apache, використовується для обробки коду по від ображенню динамічного контенту і додатків. Модуль використовується для запуску скриптів, підключається до баз даних MySQL і передачі обробленого контенту в веб-сервер для візуалізації. На платформі Raspberry Pi.



MySQL

Система управління базами даних, вільно поширювана реляційна СУБД від компанії Oracle. СУБД MySQL дуже гнучка у використанні, тим, що існує підтримка величезної кількості типів таблиць: користувач може здійснити вибір таблиці типу MyISAM, у неї є повнотекстовий пошук, такі таблиці типу InnoDB, підтримуюча транзакції на рівні окремих записів. Крім цього СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, де демонструється принцип створення нових типів таблиць.

VSFTPD

Завантажувати файли на сервер користувач повинен за допомогою FTP-клієнтів. Як FTP-сервера виступає – VSFTPD. Даний FTP-сервер поєднує в собі величезну функціональність, стабільність і безпеку.

phpMyAdmin

Ця программа написана на PHP, має відкритий вихідний код і являє собою веб-інтерфейс для управління базами даних СУБД. MySQL через інтернет-браузер і не тільки. phpMyAdmin виконує адміністрування сервера MySQL, додає нових користувачів, запускає команди і запити SQL, так само надає можливість перегляду таблиць і вмісту без даних.

NextCloud

- Безкоштовна мережева хмара, об’єм якої залежить тільки від розміру накопичувача який використовується;
- Доступ через веб-інтерфейс;
- Додатки для Windows, Linux, Mac OS, Android, IOS;
- Шифрування данних;
- Вбудований календар та контакти;
- Вбудований аудіо та відеоплеєр;
- Вбудований нотатник:
- Можливість працювати як в локальній так і в глобальній мережі.

Фото діючого пристрою

Висновки

Отже, проект зі створення індивідуального програмно-апаратного рішення на базі мікрокомп'ютера Raspberry Pi 3B+ для використання в екосистемі розумного будинку успішно досягнув своїх основних цілей. Завдяки ретельному підбору ефективного програмного стеку, налаштуванню операційної системи, розробці конфігураційних файлів та встановленню програмного стеку було створено функціональне та зручне хмарне сховище даних.

Значення цього проекту полягає у зростаючій популярності хмарних технологій та забезпеченні зручного віртуального середовища для обробки та зберігання інформації. Завдяки використанню можливостей мікрокомп'ютера Raspberry Pi 3B+, цей проект пропонує доступне рішення для приватних осіб які хочуть побудувати зручну і функціональну систему розумного будинку

ДЯКУЮ ЗА УВАГУ