Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Комп'ютерна система зберігання та обробки даних на основі Raspberry Pi

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІз-21-1
Валерія ШАБАТУРА
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник: ст. викл. В'ячеслав РАДЧЕНКО
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Шабатурі Валерії Олександрівні
(прізвище, ім’я, по батькові)

1. Тема роботи Комп'ютерна система зберігання та обробки даних на основі Raspberry Pi

затверджена наказом по університету від “ 05 ” травня 2025 р. № 72 СТз

2. Термін подання здобувачем роботи до екзаменаційної комісії 16 червня 2025 р.

3. Вхідні дані до роботи
Arch Linux ARM
комп'ютерна система
дані
Raspberry Pi

4. Перелік питань, що потрібно опрацювати у роботі
Огляд центрів зберігання даних та систем моніторингу
Системи моніторингу та збергігання даних в ЦОД
Реалізація основних програмних та апаратних засобів

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій 12 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання та аналіз літератури	05.05.2025–10.05.2025	
2	Огляд існуючих аналогів	11.05.2025–03.06.2025	
3	Вибір алгоритмів	04.06.2025–06.06.2025	
4	Вибір програмних засобів	07.06.2025–08.06.2025	
5	Програмна реалізація	09.06.2025–11.06.2025	
6	Аналіз отриманих результатів	12.06.2025–13.06.2025	
7	Оформлення записки	14.06.2025–16.06.2025	

Дата видачі завдання " 05 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. В'ячеслав РАДЧЕНКО
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 59 с., 21 рис., 1 дод., 8 джерел.

КОМП’ЮТЕРНА СИСТЕМА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, RASPBERRY PI, ARCH LINUX ARM, КЛІЄНТ-СЕРВЕР.

Метою кваліфікаційної роботи є є розробка комп’ютерної системи зберігання та обробки даних для Arch Linux ARM.

У ході виконання кваліфікаційної роботи була розроблена система зберігання даних на основі одноплатного комп'ютера Raspberry Pi. Серверна частина КС реалізована у вигляді HTTP-сервера, написаного мовою програмування Go, що відповідає за обробку запитів клієнтів та повернення відповідей у форматі JSON. Архітектура серверного програмного забезпечення базується на парадигмі Model-View-Controller, що забезпечує чіткий розподіл логіки між обробкою запитів, представленням даних та їхньою моделлю.

Клієнтська частина представлена двома застосунками: веб-інтерфейсом, розробленим на основі JavaScript-фреймворку Vue.js, та мобільним застосунком для операційної системи Android. Обидва застосунки взаємодіють із сервером через мережеві запити методами POST, GET, DELETE та UPDATE, використовуючи стандартну авторизацію та зберігаючи відповідну інформацію на стороні клієнта.

ABSTRACT

Bachelor’s thesis: 59 pages, 21 figures, 1 appendices, 8 sources.

COMPUTER SYSTEM, SOFTWARE, RASPBERRY PI, ARCH LINUX ARM, CLIENT-SERVER.

The major goal of this thesis is the development of a computer system for data storage and processing based on Arch Linux ARM.

In order to a data storage system was designed using the single-board computer Raspberry Pi. The server-side component of the computer system was implemented as an HTTP server written in the Go programming language, responsible for handling client requests and returning responses in JSON format. The architecture of the server software is based on the Model-View-Controller (MVC) paradigm, which ensures a clear separation of concerns between request processing, data presentation, and data modeling.

The client-side includes two applications: a web interface developed using the Vue.js JavaScript framework and a mobile application for the Android operating system. Both applications interact with the server via network requests using the POST, GET, DELETE, and UPDATE methods, applying standard authorization mechanisms and storing relevant data on the client side.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ЦОД – центр обробки та зберігання даних

ARM – поліпшена RISC машина

IoT – internet of things

RISC – обчислення зі скороченим набором команд

ВСТУП

Інтенсивне зростання обсягів інформаційних ресурсів та постійне розширення потоків даних зумовлюють необхідність удосконалення інфраструктури центрів обробки даних, що, у свою чергу, вимагає розвитку систем їх моніторингу та керування. Надійність зберігання інформації забезпечується завдяки функціонуванню спеціалізованих систем доступу до файлів, розміщених на серверному обладнанні, якими можуть користуватися авторизовані клієнти. Серед відповідних рішень особливе місце посідає протокол FTP, що реалізує модель взаємодії на основі архітектури «клієнт-сервер» і дозволяє передавати значні обсяги даних у мережевому середовищі. Вдосконалена модифікація FTPS забезпечує додатковий рівень безпеки за рахунок використання транспортних криптографічних механізмів. Станом на сьогодні існує широкий спектр реалізацій клієнтських і серверних програмних засобів, які підтримують цей протокол, зокрема FileZilla.

Технологічною основою для розробки сучасних серверних рішень у контексті веб-інфраструктур є Node.js, що функціонує на основі рушія V8 і підтримує обробку HTTP-запитів. Ця платформа вважається ефективною для побудови мікросервісної архітектури та розробки клієнт-серверних веб-додатків, однак її обмеженням є асинхронна обробка запитів в однопоточному режимі, що може впливати на масштабованість при високому навантаженні. Поширеною практикою в цифровому середовищі є використання веб-серверів для організації доступу до файлів шляхом генерації унікальних посилань. Такі сервіси, як dropfiles.com, забезпечують створення тимчасових адрес для завантаження матеріалів, при цьому встановлюючи обмеження на термін зберігання та максимальний розмір переданих даних.

Сучасні системи моніторингу, які впроваджуються в центрах обробки даних, характеризуються надмірною ресурсомісткістю як у частині

обчислювального навантаження, так і щодо споживання інфраструктурних ресурсів. Їх архітектура є досить складною для розгортання і подальшого обслуговування, що суттєво підвищує вартість реалізації подібних рішень. Високі витрати на впровадження і супровід гальмують інтеграцію нових підходів до моніторингу в існуючих центрах. Саме тому подальший розвиток цієї галузі передбачає необхідність спрощення архітектурних рішень, зменшення енергоспоживання та оптимізацію обчислювальних ресурсів. Такий підхід дозволить не лише знизити вартість впровадження, але й спростити адаптацію систем моніторингу в уже діючих інфраструктурах, сприяючи більш динамічному впровадженню сучасних технологій у сфері обробки даних.Метою роботи є розробка комп’ютерної системи зберігання та обробки даних для Arch Linux ARM.

Завдання:

- аналіз сервісів зберігання даних;
- огляд існуючих систем зберігання даних;
- розробка серверу;
- розробка веб-застосунку клієнта;
- розробка мобільного застосунку клієнта.

1 ОГЛЯД ЦЕНТРІВ ЗБЕРІГАННЯ ДАНИХ ТА СИСТЕМ МОНІТОРИНГУ

1.1 Загальні відомості про ЦОД

У сучасному тлумаченні дата-центр, або центр обробки даних (ЦОД), постає як інтегроване організаційно-технічне середовище, призначене для формування масштабованої, надійної та високодоступної інформаційної інфраструктури. У більш прикладному значенні цей термін стосується спеціалізованого приміщення, в якому розміщується технічне обладнання для обробки, збереження та трансляції цифрових даних [1].

Типологія дата-центрів визначається функціональним призначенням та моделлю експлуатації, що відповідає специфіці організаційної діяльності. Розрізняють інфраструктури корпоративного рівня, які інтегровані у внутрішню систему підприємства, сервіси, орієнтовані на надання обчислювальних ресурсів у вигляді послуги, а також архітектури, що реалізують технологічні принципи Web 2.0. Характеристики таких центрів можуть суттєво відрізнятися за параметрами внутрішнього й зовнішнього трафіку, механізмами маршрутизації, рівнем серверної віртуалізації, застосовуваними технологіями зберігання інформації та загальним масштабом обчислювальних ресурсів [2].

Складові сучасного центру обробки даних охоплюють декілька взаємопов’язаних функціональних груп. Технічна інфраструктура забезпечує сталість обчислювальних процесів та охоплює серверні системи, масиви зберігання й резервування даних, телекомунікаційні рішення для забезпечення взаємодії між інформаційними потоками та зовнішніми підключеннями, а також інженерні рішення, спрямовані на підтримку енергетичної стабільності, кліматичної рівноваги й протипожежної безпеки. Додаткову критичну роль відіграє система захисту, що забезпечує контроль

доступу до конфіденційних даних та реагування на потенційні інциденти.

Програмне забезпечення формує логічний рівень функціонування ЦОД, реалізуючи повний спектр інструментів для підтримки бізнес-процесів. До таких засобів відносять операційні системи для серверів і робочих станцій, платформи управління базами даних, механізми резервного копіювання, системи кластеризації, програмні інтерфейси пристроїв зберігання, сервіси адміністрування та моніторингу, інструменти інвентаризації, засоби для роботи з документами, електронну пошту та засоби доступу до Інтернету.

Організаційне середовище ЦОД забезпечує узгоджене надання інформаційно-технологічних послуг відповідно до сучасних стандартів управління, таких як ISO/IEC 20000. Його функціонал включає процеси управління якістю та доступністю сервісів, взаємодії з клієнтами та контрагентами, вирішення інфраструктурних інцидентів, підтримку конфігураційного обліку та технічної документації. Окрім того, забезпечується планування змін у сервісах із мінімальними ризиками для користувачів, а також здійснюється керування впровадженням нових релізів у продуктивне середовище [3].

1.2 Топологія ЦОД

Мережа центрів обробки даних проєктується з урахуванням потреб розміщення віртуалізованих серверів і гнучких середовищ зберігання, що динамічно змінюються в залежності від вимог обчислювальних процесів. Архітектура мережевої інфраструктури ЦОД є критично важливим чинником, який визначає рівень продуктивності системи в цілому. Вона суттєво впливає на здатність обслуговувати мікросервісні запити, підтримувати оперативність функціонування та забезпечувати гнучке пристосування до змін у програмному середовищі.

З огляду на структурні особливості, топології центрів обробки даних поділяють на кілька основних класів. До поширених відносяться ієрархічні

структури, що мають деревоподібну побудову; рекурсивні моделі, які базуються на повторюваних елементах зв’язку; гібридні архітектури, що поєднують оптичні та електричні канали передачі даних; а також прямі мережі, які утворюють топології з безпосередніми з’єднаннями між вузлами.

Залежно від здатності до масштабування та перебудови, топології також поділяються на статичні, які не змінюються після початкового розгортання інфраструктури, та динамічні, що можуть адаптуватися до змін у навантаженні або конфігурації обладнання.

Одним із найпоширеніших підходів в українській практиці є застосування традиційної деревоподібної архітектури. Така структура зазвичай містить два або три рівні і включає доступний рівень, рівень агрегації та ядро. Сервери приєднуються на периферії, формуючи листя мережевого дерева, а зв’язок між ними забезпечується комутаторами верхніх стійок. Ці комутатори, у свою чергу, підключаються до агрегаційних модулів, які спрямовують трафік до ядра. Ярус агрегації виконує роль зони обробки, відповідальної за балансування навантаження, тоді як ядро реалізує глобальну маршрутизацію між сегментами мережі та забезпечує вихід до зовнішніх систем. Така ієрархічна модель дозволяє централізовано координувати трафік і масштабувати інфраструктуру у відповідь на зростання обчислювальних потреб.

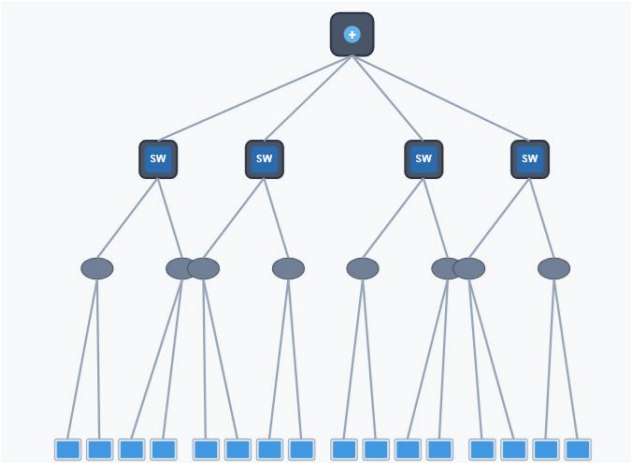


Рисунок 1.1 – Традиційна топологія базових деревних мереж

Топологія Fat-Tree, зображена на рисунку 1.2, базується на Clos-архітектурі та реалізована у вигляді багаторівневої ієрархії, що функціонує як мережеве дерево з рівномірним розподілом каналів зв’язку. Такий підхід дозволяє досягати високої масштабованості та балансування навантаження між мережевими сегментами. Концепція Fat-Tree стала основою для створення низки дослідницьких мереж у сфері обробки даних, серед яких Portland, Hedra, Elastic Tree та інші. Її структура забезпечує симетричність маршрутизації та сприяє ефективному використанню пропускної здатності в межах дата-центру.

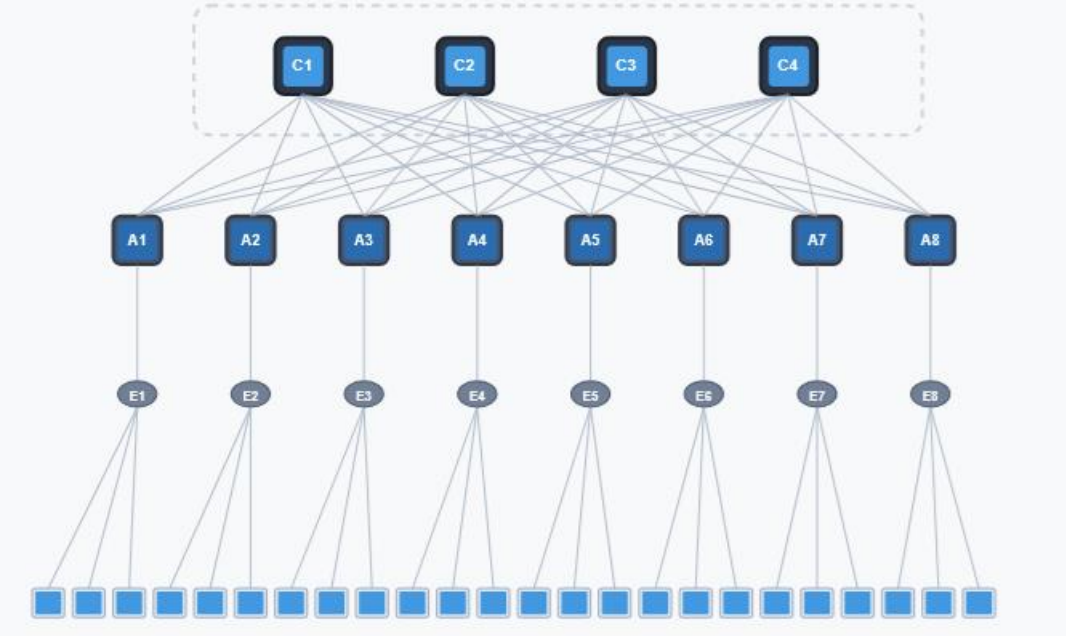


Рисунок 1.2 – Проста 3-рівнева топологія Fat-Tree

Архітектура VL2 [4] характеризується гнучкістю та оптимальним співвідношенням вартості й продуктивності. Вона побудована на базі Clos-топології, що включає численні комутатори, організовані у багаторівневу структуру. Основною особливістю цієї моделі є використання механізму достовірного балансування навантаження (VLB), що дозволяє рівномірно розподіляти трафік по різних маршрутах мережі. Такий підхід дає змогу підтримувати великі серверні пули та усувати проблему фрагментації ресурсів. Крім того, архітектура передбачає можливість призначення будь-

якої служби на будь-який сервер у межах центру обробки даних, незалежно від його фізичного розташування. Проте за умови інтенсивного трафіку система може демонструвати знижену ефективність через потенційні перевантаження каналів передачі даних.

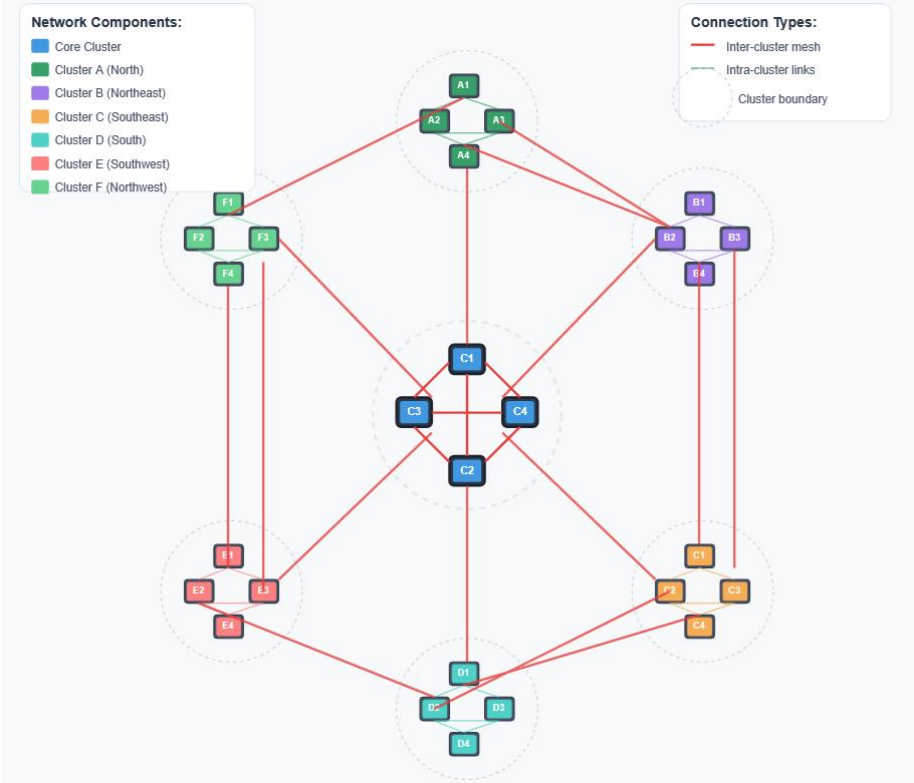


Рисунок 1.3 – Приклад топології Dcell

У свою чергу, модель DCell [5], яка ілюстрована на рисунку 1.3, реалізує рекурсивну структуру за участі серверів з кількома мережевими портами та комутаторів нижчого рівня. Кожен базовий кластер DCell формується із n серверів і одного n-портового комутатора, до якого підключаються всі сервери відповідної групи. Така організація дає змогу досягати високої масштабованості та відмовостійкості мережі. Проте в ситуаціях, коли значна частина трафіку передається через канали нижнього рівня, виникає перенавантаження вищих ланок мережі, що знижує загальну пропускну спроможність вузлів і може призводити до втрат продуктивності при зростанні інтенсивності запитів.

1.3 Основні недоліки ЦОД

Серед основних викликів, пов’язаних із функціонуванням сучасних центрів обробки даних, слід відзначити складність їх архітектурної побудови, високий ризик відмов, а також труднощі в технічному супроводі та управлінні. Відсутність належної інтеграції ресурсів віртуалізації знижує ефективність використання апаратної інфраструктури та ускладнює логіку маршрутизації і кросування. Обмежена оптимізація ресурсів, що проявляється у надлишковому використанні окремих фізичних з'єднань для керування та зберігання даних, створює перевантаження в мережевих каналах, знижує гнучкість конфігурації та вимагає високого рівня контролю з боку персоналу.

Дата-центр у своєму фізичному вимірі репрезентує сукупність функціонально пов’язаних просторових зон, у тому числі будівель, інженерних споруд, майданчиків та приміщень, що оснащені відповідними системами життєзабезпечення та інфраструктурним обладнанням. Усі елементи цього середовища формують єдину платформу для розміщення обчислювальних засобів, засобів зберігання й обробки інформації, що повинні функціонувати з гарантованою доступністю й надійністю згідно з заданим режимом експлуатації [11].

Процеси проєктування, зведення та подальшої експлуатації таких об’єктів відбуваються відповідно до жорстко регламентованих стандартів. Дотримання цих норм ускладнює адміністрування інфраструктури, підвищує вимоги до точності архітектурних рішень і, як наслідок, призводить до зростання загального рівня складності обслуговування. Обмежене використання віртуалізаційних технологій не лише знижує ефективність ресурсного управління, але й збільшує обсяг кабельного з’єднання між окремими підсистемами. Це ускладнює конфігурацію внутрішньої інфраструктури, погіршує утилізацію кабельних трас і водночас підвищує вимоги до протипожежної безпеки та технічної надійності систем загалом.

2 СИСТЕМИ МОНІТОРИНГУ ТА ЗБЕРГІГАННЯ ДАНИХ В ЦОД

2.1 Існуючі системи моніторингу та центри обробки та зберігання даних

Система моніторингу визначається як комплекс апаратно-програмних засобів, призначених для безперервного збору, аналізу та обробки даних, що використовуються для обґрунтування управлінських рішень, оцінювання реалізованих проєктів або формування нових політик. Вона може виконувати як функцію безпосереднього інформаційного забезпечення керівництва, так і слугувати інструментом зворотного зв’язку, у тому числі з громадськістю. Такі системи зазвичай спрямовані на вирішення одного або кількох завдань організаційного характеру: діагностику поточного стану об'єктів або процесів, формування звітності щодо ефективності впроваджених ініціатив або забезпечення дотримання нормативно-правових вимог і договірних зобов’язань.

Інструменти контролю зазвичай застосовуються для нагляду за подіями всередині інформаційно-технологічної інфраструктури. У залежності від завдань можуть реалізовуватись два основних підходи: безпосереднє спостереження в режимі реального часу або накопичення даних у спеціалізованих лог-файлах із подальшим аналізом. Перший варіант є доцільним для оптимізації функціонування обчислювальних систем і підвищення їх ефективності, тоді як другий підхід найчастіше використовується в автоматизованих або віддалених сценаріях, з можливістю передачі результатів моніторингу до зовнішніх служб технічної підтримки для діагностики помилок, що виникають у роботі як апаратного, так і програмного забезпечення [6].

Моніторинг інженерної інфраструктури здійснюється у трьох основних напрямах. Перший пов’язаний із використанням автономних сенсорних

пристроїв, які фіксують критичні фізичні параметри середовища, зокрема протікання рідин, температуру або фізичну активність у зоні обладнання. Зокрема, датчики протікання є обов’язковими у випадках, коли для охолодження серверних залів застосовуються рідинні системи або фреон із функцією зволоження. Температурні сенсори розміщуються у ключових зонах теплових потоків, як-от гарячі та холодні коридори, а також технічні приміщення. Додатково використовуються сенсори, що виявляють рух або відкривання/закривання серверних стійок.

Другий напрям стосується безпосереднього контролю технічного обладнання, зокрема кондиціонерів, джерел безперебійного живлення, відеоспостереження та інших пристроїв. У цьому випадку оцінюється коректність функціонування техніки, наявність критичних помилок, а також аналізуються ключові параметри – напруга, сила струму, температурні характеристики тощо.

Третій напрям передбачає комплексний моніторинг усієї системи в цілому, що включає збір агрегованих показників, аналіз міжсистемної взаємодії, виявлення відхилень від номінальних режимів роботи та побудову загального уявлення про технічний стан центру обробки даних

2.2 Типи та топологія систем моніторингу

Система моніторингу в контексті мережевої інфраструктури розглядається як ключовий функціональний елемент, інтегрований у середовище обчислювальних мереж. Її завданням є здійснення періодичної перевірки стану доступності окремих вузлів і компонентів, а також оперативне виявлення збоїв у роботі. У разі виявлення аномалій або недоступності певних сегментів мережі система автоматично генерує повідомлення, яке спрямовується відповідальним адміністраторам або технічному персоналу. В окремих випадках система моніторингу не лише виконує функцію спостереження, а й може втручатися в керування мережею,

реалізуючи сценарії реагування на критичні ситуації, зокрема недоступність ключових вузлів. Поведінка таких систем у кризових умовах визначається їхньою функціональною архітектурою, яка може суттєво відрізнятись залежно від типу реалізації – від серверних додатків до автономних апаратних модулів.

На основі технічних можливостей виділяють три основні типи систем моніторингу. Перший тип складають базові реалізації, що здебільшого використовують ICMP-протокол і надають мінімальну інформацію про мережеву доступність у вигляді індикатора стану (наприклад, доступний або недоступний вузол) або часу відгуку. Через обмежений функціонал такі системи застосовуються переважно в межах невеликих локальних мереж або в інфраструктурах із мінімальними вимогами до глибини діагностики.

Другу категорію становлять розширені системи, які оперують багатьма протоколами, зокрема SNMP, CDP, SSH та іншими, що дає змогу отримувати розгорнуту інформацію про стан мережевого обладнання та сервісів. Ці системи здатні збирати дані про рівень завантаження ресурсів, активність служб, інтенсивність передавання трафіку та інші параметри. У середовищі серверів вони зазвичай функціонують у взаємодії з локальними агентами.

Найвищий рівень функціональності демонструють системи з активним контролем. Вони мають змогу не лише ідентифікувати проблеми, а й управляти пристроями в автоматичному або напівавтоматичному режимі, що дозволяє реалізовувати гнучкі сценарії реагування на події. Подібні системи вважаються доцільними для використання в центрах обробки даних, масштабних мережах, а також у середовищах із підвищеними вимогами до надійності, таких як високо доступні кластери.

2.3 Архітектура систем моніторингу та механізми моніторингу

Попри те, що кожна мережа передачі даних має індивідуальні особливості, у галузі телекомунікацій сформувалася узагальнена концепція

побудови архітектур, орієнтованих на забезпечення надійності, високої швидкості та стабільності обміну інформацією. Найбільш репрезентативним прикладом реалізації такої моделі вважається концепція Cisco Enterprise Campus 3.0 [7], що базується на трирівневій ієрархічній структурі мережі, яка включає ядро, рівень розподілу та рівень доступу. Така архітектура не лише полегшує впровадження нових функціональних можливостей і масштабування інфраструктури, а й оптимізує процеси маршрутизації, адресного планування та локалізації логічних сегментів.

У контексті проєктування систем моніторингу ключовим є питання їхньої фізичної і логічної локалізації в межах контрольованої мережі. З урахуванням принципів доступності й керованості доцільно розміщувати засоби моніторингу в зоні ядра, що забезпечує найбільш повне охоплення усіх функціональних рівнів інфраструктури.

Вибір архітектури системи моніторингу залежить від масштабів і складності мережі. Для відносно компактних середовищ, що не передбачають наявності віддалених філій, доцільним є застосування централізованої архітектури. У такій моделі моніторинг здійснюється за допомогою одного сервера, який забезпечує контроль за всією мережею. Якщо необхідно охопити також зовнішні вузли, використовуються проміжні вузли доступу (наприклад, VPN-сайти), але при цьому важливим залишається вибір оптимального місця інтеграції основного модуля моніторингу. Логічною точкою для його розташування вважається ядро мережі, що дозволяє забезпечити рівномірний доступ до всієї інфраструктури.

До переваг централізованого підходу належать відносна простота реалізації та оперативність у налаштуванні. Водночас найбільш критичним аспектом є формалізація концептуальної моделі моніторингу, яка визначає подальшу ефективність усієї системи. Візуальна схема цієї архітектури представлена на рисунку 2.1.

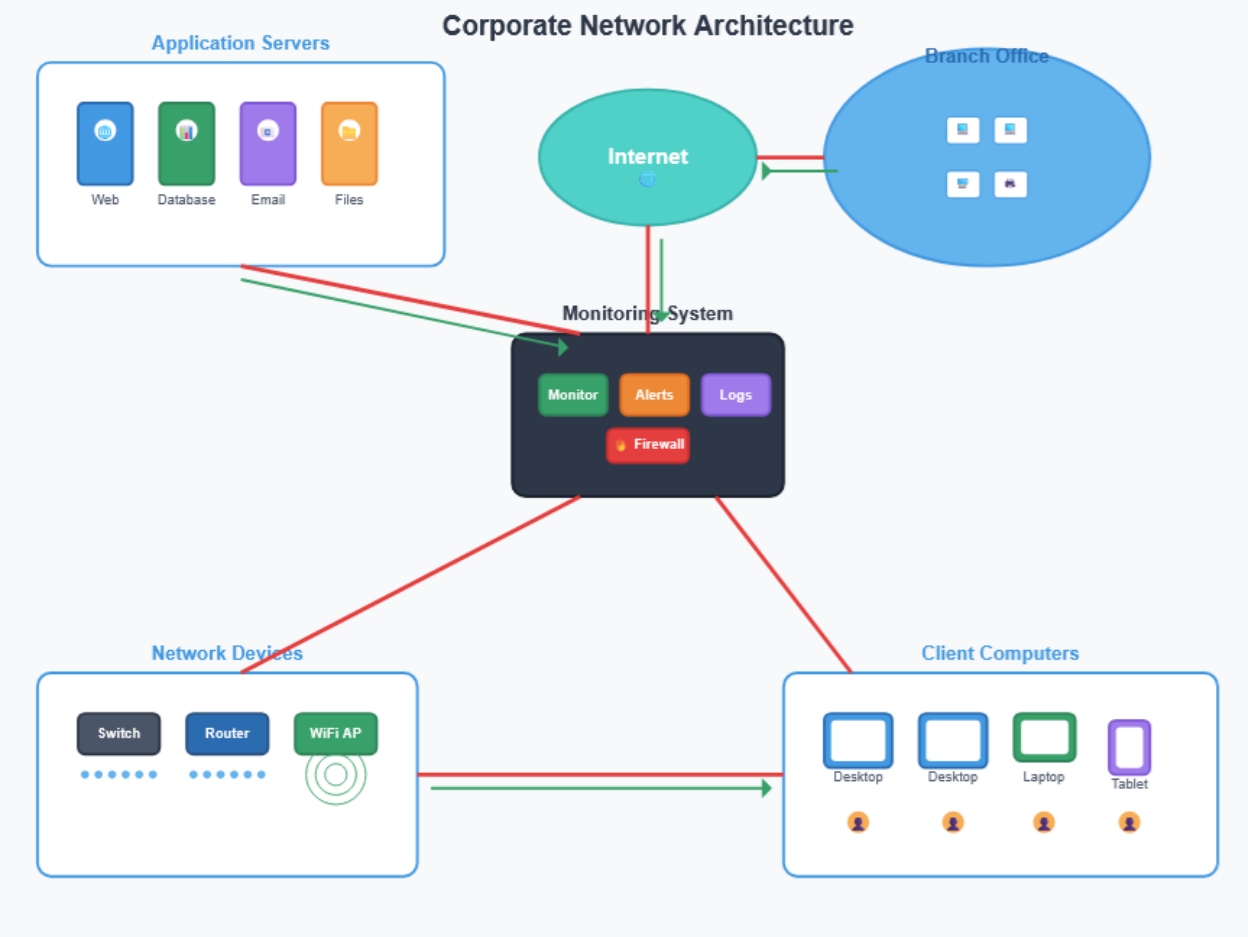


Рисунок 2.1 – Архітектура з централізованою системою моніторингу

Архітектура федеративної системи моніторингу ґрунтується на принципі розподілу контрольованої мережі на окремі логічні або фізичні сегменти, кожен з яких обслуговується власною автономною підсистемою моніторингу. Такі локальні системи функціонують незалежно одна від одної, проте регулярно передають узагальнені дані про стан своїх відповідних сегментів до центрального вузла. Завдяки цьому центральний сервер отримує повну аналітичну картину мережі, сформовану на основі децентралізованих джерел, що дозволяє точно ідентифікувати проблемні ділянки та швидко локалізувати наслідки збоїв.

Однією з ключових переваг цієї архітектури є стійкість до відмов: у випадку виходу з ладу центрального модуля зберігається можливість доступу до телеметричних даних через периферійні вузли, що забезпечує збереження моніторингових функцій у критичних ситуаціях. Такий підхід виявляється

ефективним у складних і розгалужених мережах, які обслуговують віддалені майданчики або структурно незалежні підрозділи. Він також є доцільним у середовищі провайдерів послуг, які інтегрують власні системи моніторингу для спостереження за станом клієнтських мереж або сервісів, з можливістю передавання агрегованої інформації до центрального адміністративного центру.

Федеративна модель сприяє підвищенню масштабованості, розподілу навантаження між модулями та підвищенню надійності мережевого нагляду. Концептуальна схема такої архітектури подана на рисунку 2.2.

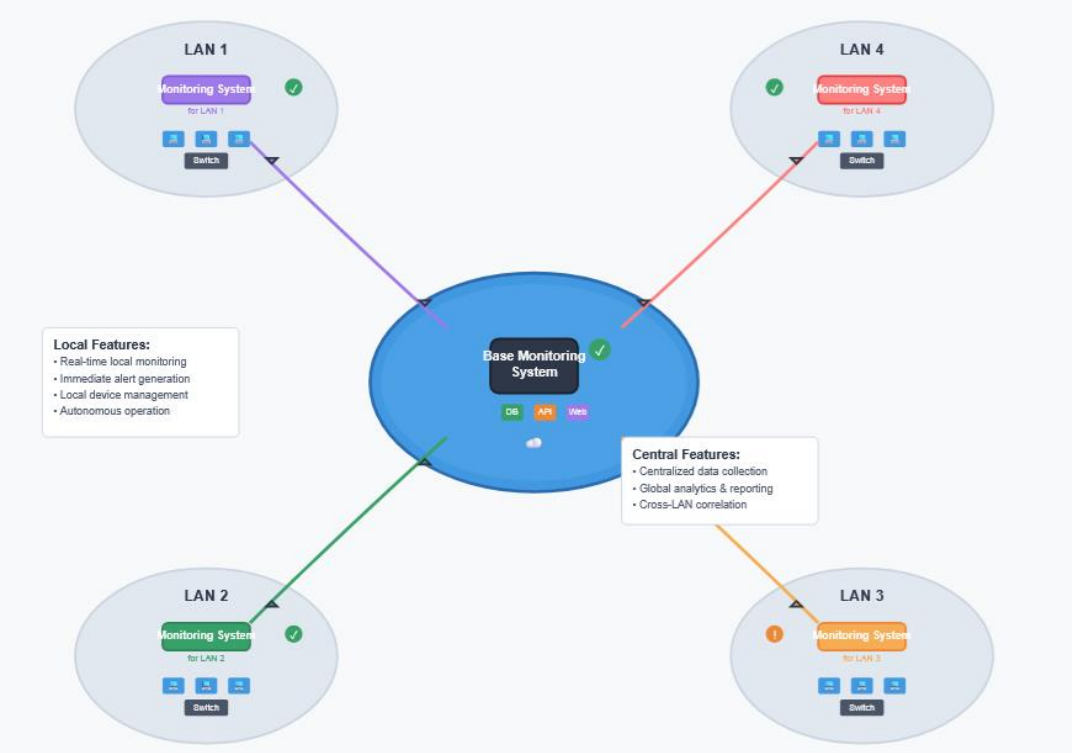


Рисунок 2.2 – Архітектура з федеративною системою моніторингу

Сучасні системи моніторингу застосовують кілька підходів для збирання телеметричних даних із контрольованих пристроїв. Одним із найпоширеніших є використання стандартних мережевих протоколів, таких як ICMP, TCP чи SNMP, що дає змогу здійснювати базову перевірку стану обладнання без необхідності глибокої модифікації конфігурації систем. Зазвичай достатньо лише надати винятки у політиках міжмережевого екрану

для забезпечення доступу. Перевагою такого підходу є його універсальність і сумісність із широким спектром пристроїв незалежно від виробника. Водночас цей метод обмежений у деталізації отриманих даних і не завжди надає доступ до специфічних функцій обладнання.

Альтернативний спосіб передбачає використання повідомлень типу SNMP Trap, які ініціюються самими пристроями. На відміну від активного опитування, притаманного першому підходу, в цьому випадку ініціатива надсилання інформації належить елементу інфраструктури, який у разі збою, перевищення порогових значень або виникнення критичних подій надсилає повідомлення до системи моніторингу. Такий механізм дозволяє своєчасно фіксувати відмови та мінімізувати затримку у виявленні несправностей. На практиці він часто поєднується з періодичним опитуванням з боку системи моніторингу, що забезпечує більшу надійність і повноту контролю [8].

Ще одним ефективним методом збору інформації є впровадження спеціального програмного агента, який функціонує безпосередньо на контрольованому пристрої та взаємодіє із системою моніторингу за моделлю клієнт-сервер. Такий агент, як правило, працює через стандартні транспортні протоколи, зокрема TCP/IP, і не потребує специфічної мережевої інфраструктури для передавання даних. Серед ключових переваг цього методу варто відзначити високу деталізацію отриманої інформації та, у разі використання активного підходу, можливість дистанційного впливу на параметри функціонування контрольованої системи. Найчастіше цей підхід застосовується для моніторингу серверів, де необхідна глибока аналітика ресурсів і контроль стану в реальному часі.

2.4 Порівняльні характеристики систем моніторингу

Для забезпечення об'єктивного порівняння різних систем моніторингу необхідно застосовувати чітко визначені критерії оцінювання та відповідні методики збору й аналізу даних. Процес такого порівняння повинен

охоплювати кілька послідовних етапів, серед яких ключовими є формулювання технічних вимог, дослідження ринку щодо доступних рішень, експериментальне тестування відповідно до обраної методології та фінальний вибір оптимального варіанта.

Одним із найважливіших критеріїв при прийнятті рішення залишається вартість впровадження системи моніторингу. Цей показник має оцінюватися не ізольовано, а у співвідношенні з потенційною економією ресурсів, зменшенням трудомісткості адміністрування та оптимізацією внутрішніх процесів. Часто саме неправильне розуміння значення цього критерію призводить до недооцінки ефективності інвестицій. Не менш важливим параметром є відповідність системи мінімальним технічним вимогам, що включає характеристики навантаження на ресурси, а також потребу у додатковому програмному забезпеченні.

Оцінка користувацького інтерфейсу, хоча й важлива, має обмежене значення через високу суб’єктивність сприйняття користувачем зручності та логіки взаємодії. Натомість складність первинного налаштування та інсталяції відіграє помітну роль у виборі системи, адже саме цей етап часто визначає успішність подальшої експлуатації. Важливою характеристикою системи є також її здатність швидко реагувати на порушення в роботі мережі, включаючи здатність ідентифікувати джерело проблеми у межах топологічної структури.

Значну увагу варто приділяти механізмам автоматичного виявлення вузлів мережі, що є частиною початкового етапу налаштування. Ефективність сканування, точність і повнота відображення реальної топології прямо впливають на надійність моніторингу. Не менш важливо оцінити способи сповіщення адміністратора про події, включно з підтримкою електронної пошти, SMS та інших каналів, які забезпечують гнучкість та адаптивність системи в умовах різного інформаційного навантаження.

Сучасні системи також можуть пропонувати додаткові функції, що виходять за межі базової специфікації, зокрема розширену аналітику,

інтеграцію з системами управління інцидентами, побудову дашбордів у реальному часі тощо. У цьому контексті важливою залишається здатність системи до інтеграції з іншими інформаційними рішеннями, що дає змогу створювати комплексні архітектури контролю та управління.

Ще одним важливим параметром є глибина моніторингу, що визначається обсягом і деталізацією зібраної інформації про керовані пристрої. Цей показник дозволяє оцінити ефективність системи з точки зору діагностики, аналізу навантажень і прогнозування відмов.

У межах дата-центрів системи моніторингу поділяються на два функціональні напрями: контроль параметрів зовнішнього середовища та контроль стану інфраструктурного обладнання. Такий розподіл забезпечує комплексне охоплення як фізичних, так і технологічних складових інформаційної інфраструктури, що дозволяє підтримувати її стабільне функціонування і вчасно виявляти потенційні загрози [3].

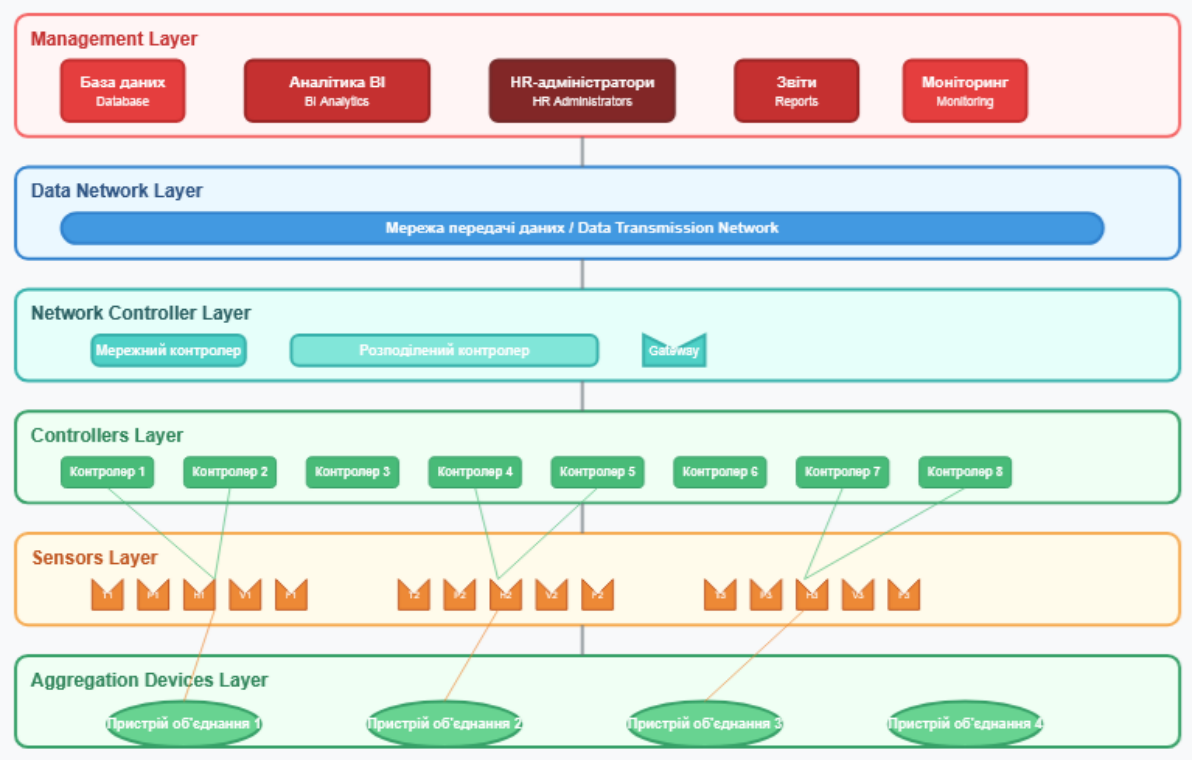


Рисунок 2.3 – Архітектура систем моніторингу інженерної інфраструктури і параметрів зовнішнього середовища ЦОД

Ключовими структурними складовими систем моніторингу незалежно від їхньої функціональної орієнтації залишаються датчики та контролери, що забезпечують вимірювання та передавання відповідних параметрів до центрального вузла обробки даних. До складу систем також входить спеціалізоване програмне забезпечення, яке виконує функції конфігурування, управління та візуалізації отриманих показників. На ринку домінують виробники, що пропонують комплексні рішення з урахуванням інтеграції модулів резервного живлення, систем охолодження, розподілу електроенергії та супутніх інженерних елементів для центрів обробки даних. Серед найбільш відомих брендів, присутніх на вітчизняному ринку, слід відзначити компанії APC, Conteg, Eaton, Knurr, Rittal і Vutlan.

Найчастіше застосування модулів моніторингу пов’язане з IT-інфраструктурою, де вони забезпечують контроль критичних параметрів функціонування ЦОД, таких як температура, вологість, стан живлення, виявлення протікань або загроз загоряння. Водночас останнім часом спостерігається розширення сфер використання подібних систем, зокрема для моніторингу стану технічного обладнання, телематичних об'єктів, елементів промислової автоматики та інтелектуальних інженерних систем житлових і комерційних будівель.

Для оцінювання практичної ефективності обраних рішень доцільно здійснити порівняльний аналіз двох популярних систем моніторингу інфраструктурних параметрів – APC NetBotz 200 та Vutlan SC8100. Аналіз передбачає дослідження трьох ключових аспектів: технічних характеристик із точки зору масштабованості, функціональних можливостей систем та економічної складової.

У технічному аспекті аналіз зосереджується на визначенні конструктивних відмінностей, які можуть впливати на зручність інсталяції та подальшого обслуговування. Особлива увага приділяється кількості доступних портів для підключення сенсорів, можливостям розширення системи та типології підтримуваних датчиків.

Що стосується функціональних можливостей, то ефективність системи визначається не лише здатністю здійснювати збір даних, але й швидкістю та якістю реагування на вихід параметрів за встановлені межі. Важливою є наявність механізмів інтелектуального прогнозування динаміки змін, що дозволяє запобігти критичним відхиленням ще до їх фактичного виникнення. Таким чином, система повинна не просто повідомляти про порушення, а забезпечувати проактивне управління.

На завершальному етапі оцінюється вартість типового комплекту, що включає базовий модуль і датчики, необхідні для повноцінної реалізації моніторингової функції. Врахування ціни в контексті функціоналу дає змогу сформувати уявлення про доцільність інвестицій у конкретне рішення, виходячи зі співвідношення ціна/можливості.

2.5 Можливості системи зберігання та моніторингу даних

Обидва розглянуті модулі систем моніторингу ефективно реалізують своє базове призначення, яке полягає в зборі, агрегуванні та передачі даних із підключених сенсорів. Отримані значення відображаються через інтегрований веб-інтерфейс, що забезпечує доступ до показників у режимі реального часу. Такий підхід дає змогу забезпечити дистанційний контроль стану об’єктів моніторингу без потреби у фізичному втручанні до обладнання. Інтерфейсна реалізація одного з модулів системи моніторингу наведена на рисунку 2.4.

Обидві системи моніторингу підтримують функціональність, яка дозволяє визначати допустимі діапазони значень контрольованих параметрів та реалізовувати механізми сповіщення у разі перевищення або зниження значень за встановлені межі. Це забезпечує можливість своєчасного реагування на потенційні відхилення в роботі інженерної інфраструктури. Приклад реалізації такого механізму наведено на рисунку 2.5.

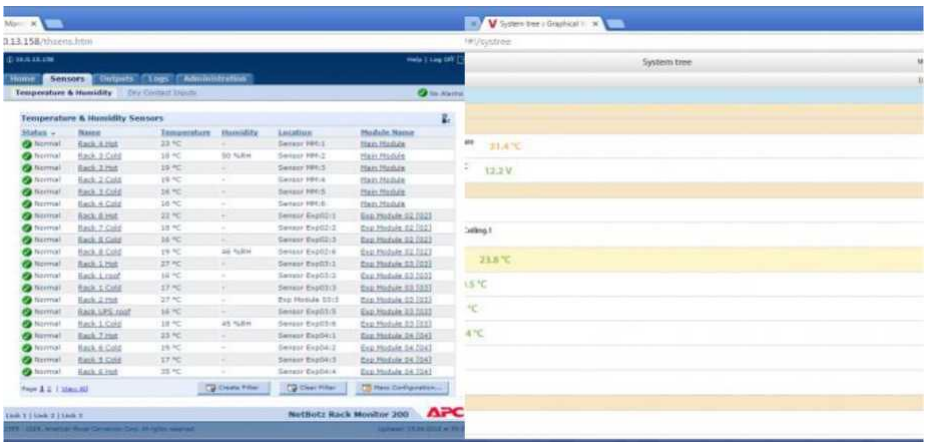


Рисунок 2.4 – Відображення значень моніторингу по Web-протоколу

Крім основної функції збору та візуалізації даних, у обох рішеннях реалізовано широкий спектр допоміжних можливостей, які підвищують функціональну гнучкість систем. До таких належать функції резервного копіювання через FTP, підтримка авторизації на основі протоколу RADIUS, синхронізація часу за допомогою NTP, передавання системних повідомлень через SYSLOG, а також побудова графічного відображення даних із сенсорів. Функціональні можливості даного типу демонструються на рисунку 2.6 [9–10].

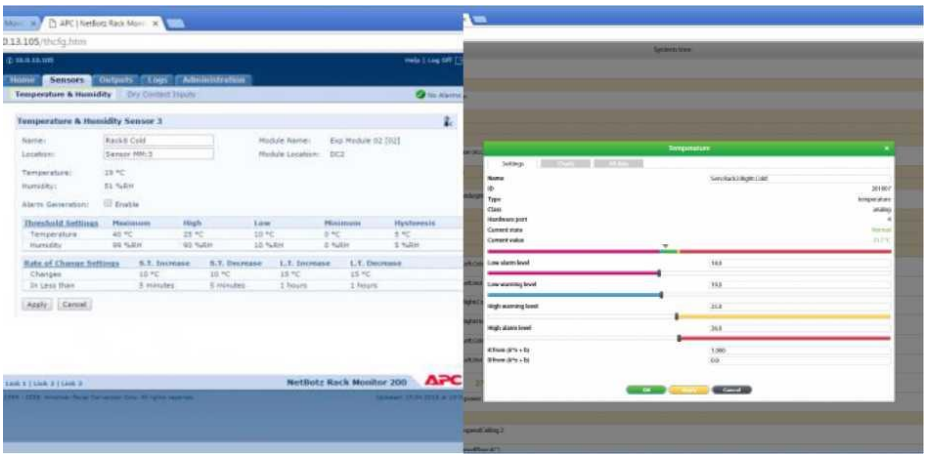


Рисунок 2.5 – Відображення значень моніторингу при виході показників за межу норми

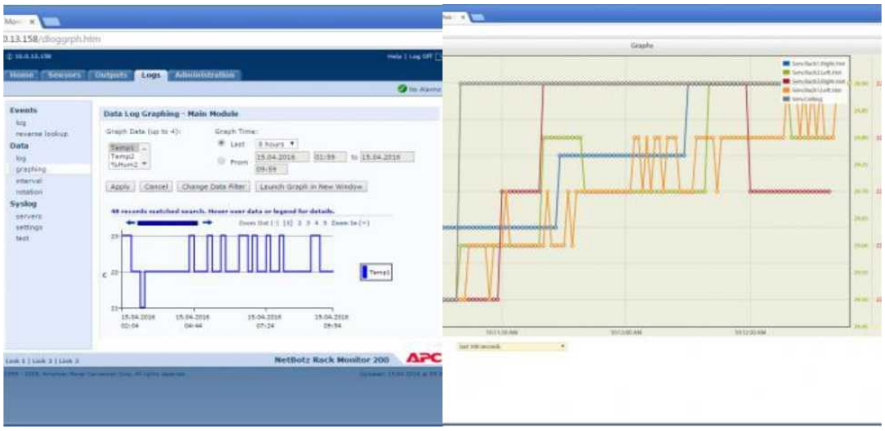


Рисунок 2.6 – Відображення значень моніторингу з додатковими можливостями

Система моніторингу репрезентує собою інтегрований комплекс апаратного забезпечення та програмних компонентів, орієнтований на безперервний збір, обробку та аналіз інформації, що надходить із контрольованих об'єктів. Отримані дані використовуються для підвищення обґрунтованості управлінських рішень, а також можуть служити джерелом для зовнішнього інформування або зворотного зв’язку, зокрема в межах реалізації проєктів, оцінки ефективності програм чи формування відповідної політики.

Функціональне призначення таких систем охоплює виявлення змін у стані критично важливих об’єктів або середовищ, щодо яких необхідне своєчасне прийняття стратегічних рішень. Системи також сприяють встановленню інформаційної взаємодії між інфраструктурою та зовнішнім середовищем, забезпечуючи зворотний зв'язок для аналізу результативності вже реалізованих дій. Крім того, вони підтримують відповідність вимогам чинних нормативів, регламентів або контрактних умов.

3 РЕАЛІЗАЦІЯ ОСНОВНИХ ПРОГРАМНИХ ТА АПАРАТНИХ ЗАСОБІВ

У якості апаратної та програмної основи для реалізації системи моніторингу планується застосування одноплатного комп’ютера Raspberry Pi 3 Model B+ у поєднанні з серверним обладнанням HP ProLiant DL120 G5. Функціонування системи буде забезпечено на основі операційних систем сімейства Linux, зокрема дистрибутивів Raspbian та Ubuntu, що відзначаються стабільністю, гнучкістю налаштування та широкими можливостями адаптації до різноманітних сценаріїв використання. У якості програмного засобу моніторингу обрано систему Nagios, яка дозволяє реалізувати централізований контроль параметрів мережевої інфраструктури, забезпечуючи виявлення збоїв, оповіщення про критичні події та збирання статистики для подальшого аналізу. Така конфігурація поєднує доступність апаратного забезпечення з високою функціональністю моніторингової платформи.

3.1 Одноплатний комп' ютер - Raspberry Pi

Платформа Raspberry Pi є компактним, енергоефективним і функціонально гнучким обчислювальним пристроєм, який активно застосовується як у навчальних цілях, так і в інженерних рішеннях. Завдяки своїм мініатюрним розмірам, співставним із банківською карткою, низькій вартості та відкритій апаратній і програмній архітектурі, Raspberry Pi набула широкого поширення серед розробників у всьому світі. Ініціатором створення цієї платформи став Девід Брабен, а її основною метою було забезпечення доступного середовища для швидкого прототипування електронних пристроїв як для початківців, так і для досвідчених користувачів.

Однією з характерних особливостей пристрою є здатність взаємодіяти з фізичним середовищем за допомогою датчиків і виконавчих механізмів. Таким чином, платформа дозволяє розширити межі використання комп'ютерних технологій поза межами віртуального середовища. Розробка програмного забезпечення для Raspberry Pi може здійснюватися як безпосередньо на самому пристрої, так і за допомогою підключення через інтерфейс USB.

Технічно Raspberry Pi побудований на базі процесора з архітектурою ARM11 із тактовою частотою 700 МГц, яка, завдяки офіційній підтримці розгону, може бути підвищена до 1000 МГц. Це забезпечує належний рівень продуктивності при вкрай низькому енергоспоживанні, яке становить приблизно 1 Вт, що є суттєво нижчим порівняно з традиційними ПК, які споживають щонайменше 250 Вт.

Додатковою перевагою платформи є її вартість, що зазвичай не перевищує 25–35 доларів США, а також підтримка відкритих програмних рішень, оптимізованих для цієї апаратної архітектури. В умовах реалізації даного проєкту передбачається використання операційної системи Debian Linux у модифікації Raspbian, яка характеризується високою стабільністю, мінімальними вимогами до ресурсів і попередньо налаштованим набором базового ПЗ, включаючи легкий веб-браузер Midori. Усі компоненти програмного забезпечення розповсюджуються безкоштовно й орієнтовані на ефективне використання ресурсів пристрою. Крім того, система дозволяє оптимізувати розподіл доступної пам'яті, що є додатковим аргументом на користь її використання в енергоощадних автоматизованих системах.

Зовнішній вигляд плати Raspberry Pi наведено на рисунку 3.1.

У якості основного програмного забезпечення для платформи Raspberry Pi використовується відкрита операційна система Raspbian, що базується на дистрибутиві Debian Linux та адаптована для архітектури ARM.

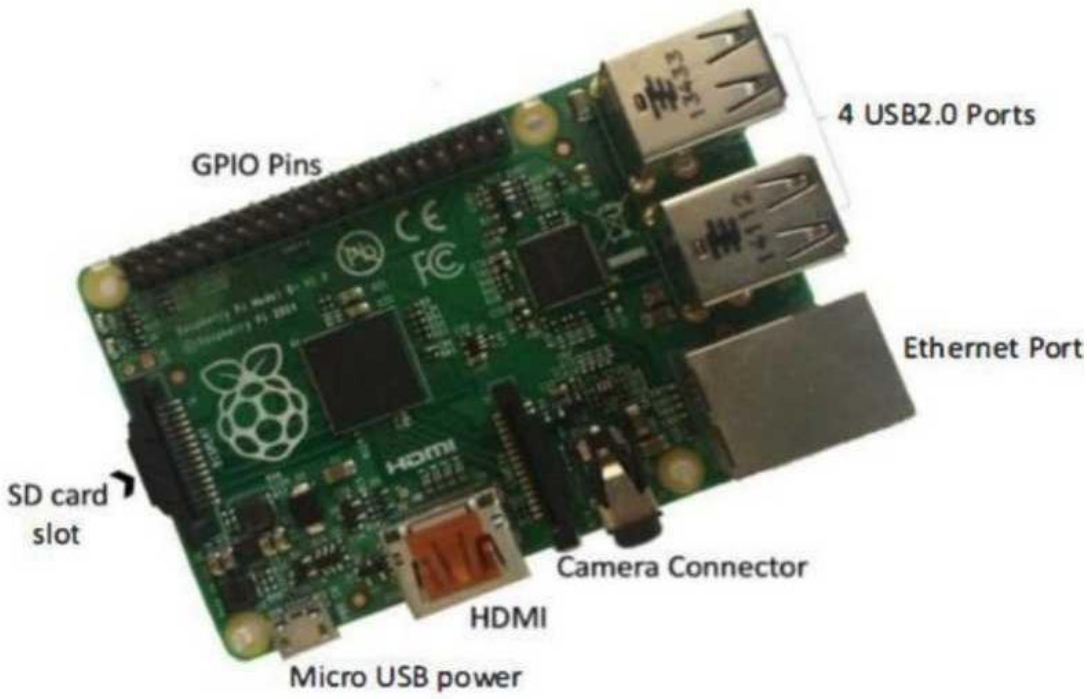


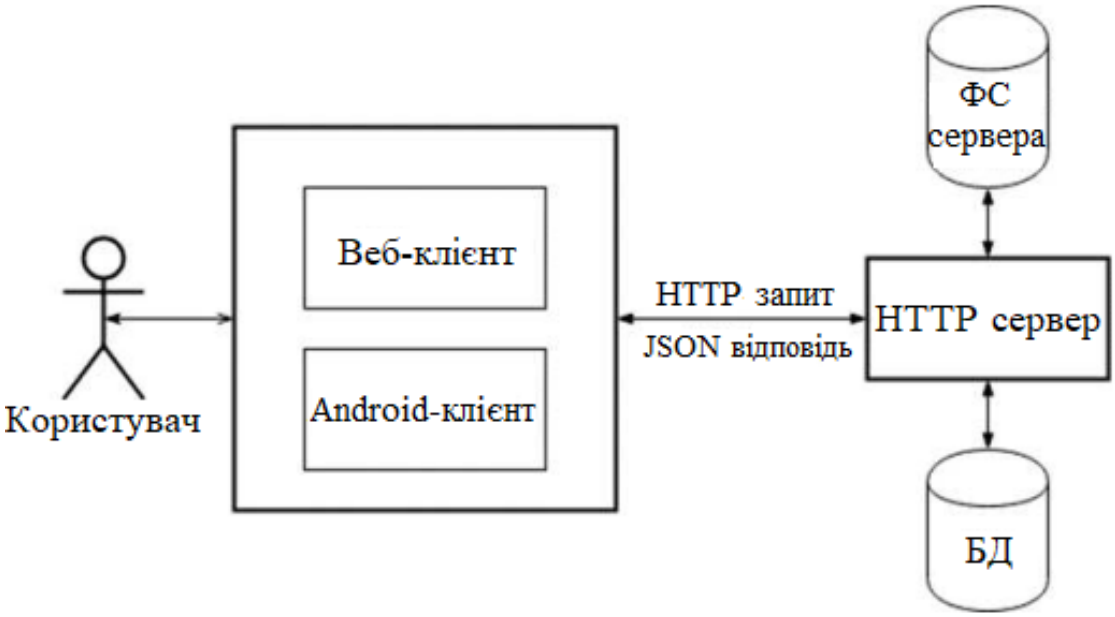
Рисунок 3.1 – Загальний вигляд плати Raspberry Pi 2

Серед встановлених програмних засобів передбачено легкий веб-браузер Midori, призначений для роботи в умовах обмежених апаратних ресурсів, а також інтерпретатор мови програмування Python, який широко застосовується для розробки скриптів і керування периферійними пристроями [13].

3.2 Переваги одноплатних комп'ютерів

Одноплатний комп’ютер (Single-board computer) являє собою повноцінну обчислювальну систему, апаратні компоненти якої інтегровані на єдиній друкованій платі. Такий пристрій містить центральний мікропроцесор, модулі оперативної пам’яті, інтерфейси введення-виведення та інші функціональні елементи, необхідні для автономного функціонування. Одноплатні комп’ютери можуть використовуватись як у навчальному процесі та дослідницькій діяльності, так і в якості інженерних платформ для прототипування або вбудованих рішень у промислових системах. Завдяки

компактності, енергоефективності та простоті інтеграції, вони знаходять широке застосування у сфері розподілених обчислень, автоматизації та інтернету



речей.

Рисунок 3.2 – Діаграма контексту системи

Одноплатні комп’ютери демонструють значну універсальність у вирішенні широкого спектра завдань, починаючи від базових функцій персонального комп’ютера для повсякденного використання і завершуючи виконанням ролі мережевого обладнання, зокрема маршрутизаторів або модемів. Завдяки можливості інсталяції повнофункціональної версії операційної системи на базі Linux, такий пристрій здатен забезпечити виконання офісних операцій, веб-навігації, відтворення аудіо- та відеоконтенту, включаючи підтримку відео з роздільною здатністю до 1080p. При цьому інтерфейс користувача залишається інтуїтивно зрозумілим і візуально звичним. Основним обмеженням у використанні подібних платформ залишається апаратна архітектура, оскільки не все програмне забезпечення має відповідні версії для ARM-процесорів. Однак у більшості випадків існують функціональні альтернативи або адаптовані аналоги, що дозволяє зберігати стабільність і функціональність роботи системи.

До найважливіших переваг одноплатних комп’ютерів відносяться їх висока енергоефективність, компактність, невелика вартість, зручність у налаштуванні, а також підтримка широкого спектра периферійних пристроїв і розширювальних модулів, що значно розширює сферу їх застосування.

3.3 Система моніторингу та зберігання даних з використанням Raspberry Pi

У сучасних системах моніторингу все ширше застосовуються одноплатні комп’ютери як базові елементи обчислювальної інфраструктури. Зокрема, одним із ефективних рішень є використання протоколу ZigBee для організації бездротової сенсорної мережі. У такій архітектурі кожен сенсор взаємодіє з мікроконтролером, який забезпечує збір і попередню обробку даних, а також виконує функцію посередника між сенсором та модулем бездротового зв’язку типу XBee. Центральний елемент системи представлений одноплатним комп’ютером, який оснащений аналогічним модулем для забезпечення зв’язку з периферійними вузлами.

Енергозабезпечення всієї сенсорної мережі відбувається автономно, шляхом використання акумуляторних батарей, що дозволяє розгортати такі системи в умовах, де підключення до постійного джерела живлення ускладнене або недоцільне. На центральному вузлі реалізується веб-застосунок, який відповідає за отримання та обробку даних від мікроконтролерів, а також за формування графічного інтерфейсу для взаємодії з кінцевим користувачем.

Особливістю протоколу ZigBee є наявність координатора, який виконує керування топологією мережі. У даній реалізації ця роль покладена на одноплатний комп’ютер, який забезпечує централізовану логіку управління мережею. Структурна організація такої архітектури відображена на рисунку 3.3.

У якості периферійних елементів системи застосовуються мікроконтролери Arduino, що представляють собою апаратно-програмну платформу для реалізації простих автоматизованих рішень. Вона орієнтована на широке коло користувачів, включаючи ентузіастів, викладачів та розробників, що не мають спеціалізованої інженерної підготовки. Програмна частина реалізується через відкриту середу розробки (IDE), яка підтримує написання, компіляцію і завантаження програмного коду на апаратну платформу. Апаратна частина включає набір друкованих плат, які виготовляються як офіційним виробником, так і численними сторонніми розробниками [12]. Завдяки відкритій архітектурі платформи, можливе не лише використання стандартних рішень, але й створення модифікованих апаратних конфігурацій. Схематичне зображення центрального та периферійних вузлів представлено на рисунку 3.3.

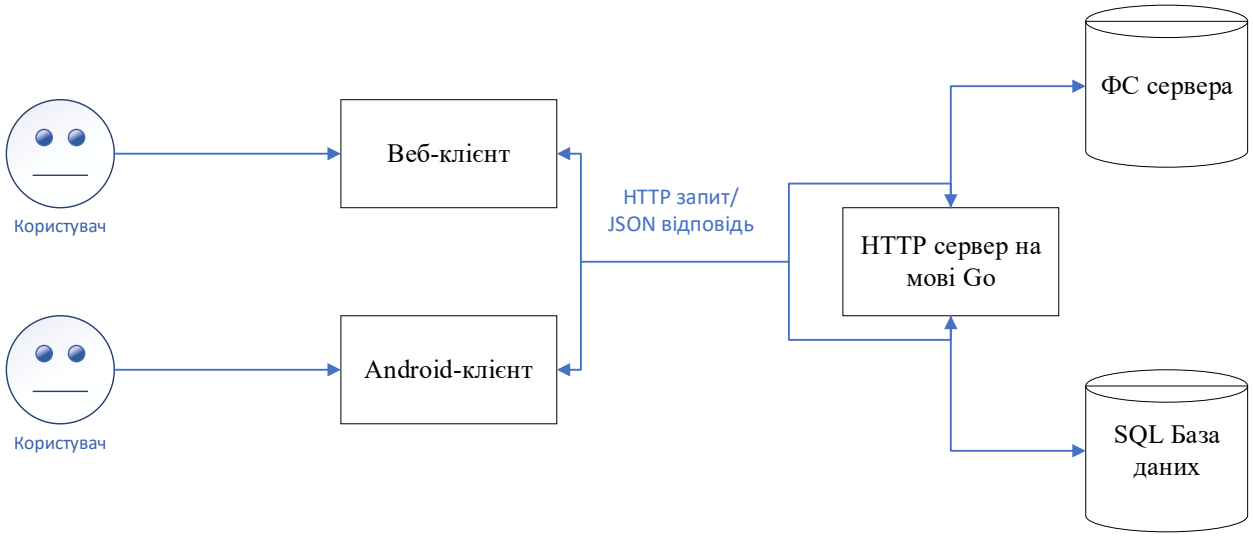


Рисунок 3.3 – Структура системи моніторингу та зберігання даних на основі одноплатних комп’ютерів

Усі зовнішні пристрої, інтегровані до мережевої інфраструктури, потребують постійного контролю в режимі реального часу для забезпечення стабільності функціонування всієї системи. Завдання мережевого моніторингу полягає у збиранні актуальних даних, формуванні статистичних

звітів та оцінюванні ефективності роботи мережевих компонентів. У разі виникнення технічних збоїв або відхилень у роботі елементів мережі система повинна своєчасно інформувати адміністратора, що дозволяє запобігати ескалації нештатної ситуації до критичного рівня. Реалізація механізмів сповіщення може здійснюватися через електронну пошту, SMS або інші засоби миттєвого повідомлення, включаючи мобільні платформи.

Поняття "мережевий моніторинг" охоплює сукупність засобів та процесів, спрямованих на безперервне спостереження за топологічною структурою мережі, виявлення уповільнень, збоїв або перевантаження системи. Основною функцією таких систем є оперативне виявлення несправностей і автоматизоване сповіщення відповідального персоналу про виявлені порушення. Водночас ефективність мережевого моніторингу значною мірою залежить від правильного визначення критичних параметрів, які підлягають відстеженню. Серед найбільш типових напрямів контролю – оцінювання використання пропускної здатності каналів, рівень продуктивності серверів, а також ефективність функціонування прикладного програмного забезпечення.

Варто зазначити, що моніторинг серверної інфраструктури є невід’ємною складовою будь-якої комплексної архітектури моніторингу центрів обробки даних. Однак, на практиці ця складова часто недооцінюється або реалізується фрагментарно, що суттєво обмежує ефективність всієї моніторингової платформи. Таким чином, інтеграція серверного моніторингу повинна розглядатися як пріоритетна задача при побудові цілісного середовища контролю за станом інформаційної інфраструктури.

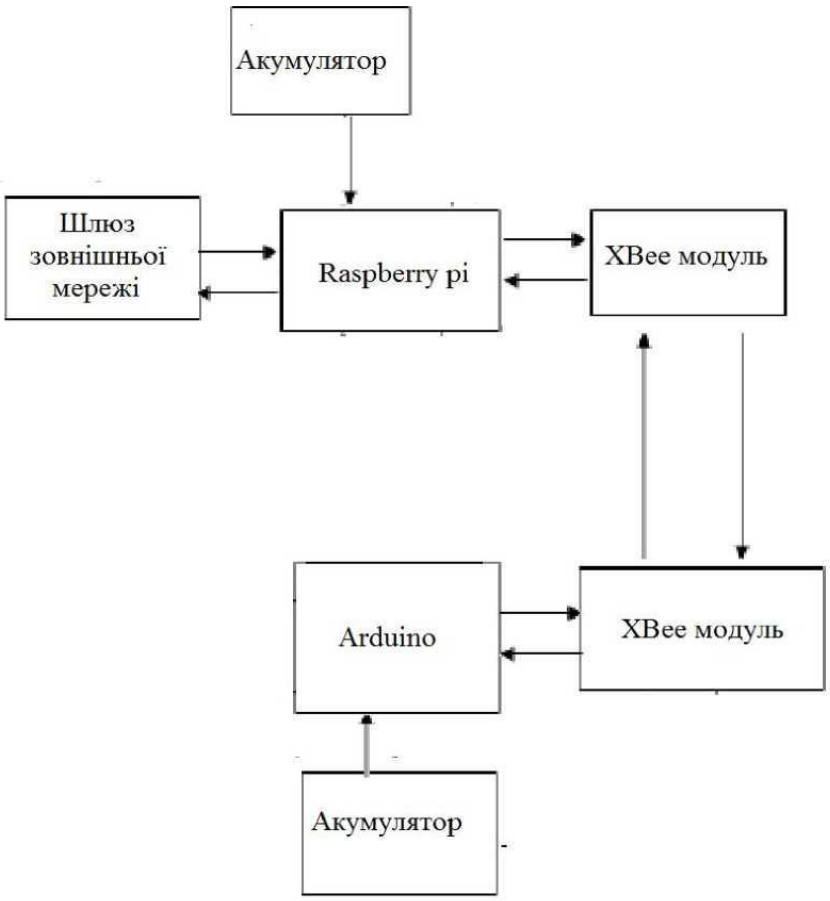


Рисунок 3.4 – Блок-схема центрального і дочірнього вузлів системи

3.4 Система моніторингу Nagios

Моніторинг серверної інфраструктури передбачає постійне спостереження за операційною системою та ключовими апаратними показниками серверів, які виконують функції хостів для прикладного програмного забезпечення. Такий підхід забезпечує системний контроль за зовнішнім функціонуванням сервера без детального заглиблення у внутрішні процеси операційного середовища. Серед основних параметрів, що підлягають спостереженню, виділяються навантаження на процесор, рівень використаної та вільної оперативної пам’яті, активність дискових підсистем, швидкість роботи мережевих інтерфейсів, а також загальний стан обробки запитів. Впровадження подібного моніторингу є стандартною практикою для більшості IT-організацій.

Більшість сучасних засобів реалізовано на основі протоколу SNMP, а ринок відкритого програмного забезпечення пропонує велику кількість інструментів, таких як Nagios, Zabbix, Cacti, Zenoss, OpenNMS тощо. Вибір системи зазвичай залежить від типу бізнесу, інфраструктурних вимог і вподобань у частині відкритого або комерційного ліцензування. Багато компаній схиляються до впровадження рішень з відкритим кодом, оскільки вони дають змогу адаптувати інструменти до індивідуальних потреб і масштабів проєктів.

Одним із найпоширеніших і найгнучкіших інструментів у цій категорії є система моніторингу Nagios. Вона підтримується практично всіма дистрибутивами Linux та може бути розгорнута також у середовищах UNIX. Розроблена Ethan Galstad, ця система дозволяє здійснювати моніторинг мережевих пристроїв, серверів, служб та додатків, а також створювати розширювану інфраструктуру завдяки підтримці плагінів. Nagios надає адміністраторам інструменти для оперативного контролю над критичними сервісами, такими як SMTP, POP3, HTTP, ICMP, а також ресурсами, включаючи файлові системи, завантаження процесора, пам’ять, журнали подій тощо. У разі виявлення порушень система негайно надсилає сповіщення за попередньо визначеними каналами – електронною поштою, SMS або іншими механізмами, залежно від налаштування.

Завдяки гнучкій архітектурі розширень, користувачі можуть створювати власні модулі моніторингу на основі будь-якої зручної мови програмування, що суттєво розширює функціонал базової системи. Крім того, Nagios підтримує паралельну перевірку служб і побудову ієрархічної моделі хостів, що дозволяє диференціювати між безпосередньо недоступними вузлами й тими, що втратили з’єднання через відмову вищого рівня. У випадку виявлення критичних подій можуть автоматично запускатися обробники, що виконують коригувальні дії, що є основою для реалізації проактивної моделі управління.

Серед додаткових функцій варто відзначити можливість створення розподілених інсталяцій для підвищення надійності, підтримку захищеного віддаленого моніторингу через тунелі SSH або SSL, а також вбудовану утиліту nagiostats, яка формує зведену статистику по всіх контрольованих елементах. Значну увагу приділено візуалізації результатів у реальному часі через веб-інтерфейс, що дозволяє не лише оперативно реагувати на проблеми, а й аналізувати тенденції зміни параметрів.

Окрім Nagios, популярністю користуються й інші інструменти. Наприклад, Cacti реалізовано на основі LAMP-стеку з використанням RRD (Round Robin Database), що дозволяє ефективно збирати й візуалізувати дані у вигляді графіків. Zenoss пропонує комплексний підхід до моніторингу мережі, серверів і додатків із використанням централізованої бази даних подій на основі MySQL. Система Zabbix, створена Олексієм Владисьєвим, дозволяє працювати без агентів із ключовими протоколами (SMTP, HTTP, ICMP) і передбачає наявність демонів, агентів і веб-інтерфейсу як основних компонентів архітектури.

Загалом, саме Nagios, завдяки своїй надійності, гнучкості, широкому спектру плагінів і активній спільноті розробників, залишається базовим інструментом у сфері моніторингу мережевих систем і інфраструктур ЦОД [15–16].

Система Nagios виступає як ефективний засіб аналізу та візуалізації мережевого трафіку в режимі реального часу, що дозволяє проводити діагностику інфраструктури без порушення стабільності її функціонування. За допомогою цієї системи забезпечується моніторинг пропускної здатності каналів передачі даних, а також оперативне сповіщення адміністратора у випадках зниження доступності або перевищення встановлених порогових значень навантаження. Додатково, завдяки інтеграції з інструментом MRTG, адміністратору надається можливість отримати статистику щодо мережевого навантаження під час пікових періодів активності, що сприяє попередженню потенційних відмов.

Ключовою функцією Nagios є спостереження за станом мережевих пристроїв та служб, із можливістю негайного інформування відповідальних осіб у разі виникнення збоїв. Основу системи складає демон планувальника, який здійснює регулярну перевірку заздалегідь визначених елементів інфраструктури. У разі виявлення аномалій ініціюється механізм сповіщення, що використовує електронну пошту, месенджери чи інші засоби комунікації. З доступом до веб-інтерфейсу адміністратор має змогу контролювати загальний стан мережі, переглядати події та звіти, що значно спрощує керування у віддаленому режимі.

Архітектура системи побудована таким чином, що дозволяє надсилати інформацію про збої не лише IT-персоналу, а й бізнес-одиницям і кінцевим користувачам. Завдяки мобільній підтримці користувачі можуть отримувати звіти з Nagios на пристроях з ОС Android чи iOS, що підвищує рівень мобільності та оперативності реагування. Крім того, Nagios передбачає багатокористувацький доступ до інформації через веб-інтерфейс із можливістю призначення рівнів дозволу, що дозволяє користувачам бачити лише ті частини інфраструктури, до яких вони мають авторизований доступ. Такий підхід підвищує безпеку та оптимізує адміністративну роботу.

Веб-інтерфейс системи забезпечує адаптацію до індивідуальних потреб кожного користувача завдяки налаштуванню шаблонів і дизайну відображення даних. Попри численні переваги, варто враховувати й певні обмеження: зокрема, конфігурація системи базується на текстових файлах, що ускладнює її розгортання без попередньої підготовки. Інтерфейс базується на застарілій CGI-технології та не підтримує сучасні інтерактивні елементи, що створює труднощі для інтеграції з новітніми веб-технологіями. Окрім цього, Nagios не має вбудованої бази даних для накопичення історичних даних, що обмежує можливості довгострокової аналітики.

Система підтримує плагінну архітектуру, де кожен плагін є окремою програмною одиницею з визначеним набором параметрів, які виконують діагностичні дії або надсилають повідомлення. Контрольні плагіни

формують висновки про стан мережі, тоді як плагіни повідомлень відповідають за інформування про зміну статусу компонентів. За допомогою NRPE розширення стає можливим моніторинг локальних ресурсів, як-от завантаження процесора чи обсяг доступної пам’яті, без необхідності прямого доступу до пристроїв. Однією з найважливіших особливостей є можливість створення власних плагінів, що дає змогу гнучко адаптувати систему під специфічні умови конкретної мережі.

Nagios надає розробникам інтерфейс під назвою Nagios Event Broker (NEB), що дозволяє реалізовувати додаткові функціональні модулі, які взаємодіють із ядром системи під час виникнення певних подій. Наприклад, можна реалізувати обробку результатів перевірок після завершення роботи відповідних плагінів. Проте слід враховувати, що некоректна реалізація таких модулів може призводити до блокування основного процесу, що потенційно знижує загальну продуктивність системи.

Встановлення та налаштування Nagios потребує розуміння структури конфігураційних файлів, що робить початкове розгортання непростим завданням. Для спрощення цього процесу спільнота розробила низку інструментів, зокрема Lilac і Fruity, які надають графічний інтерфейс на базі PHP для конфігурування системи. У випадках складних розгортань у корпоративних середовищах рекомендовано використовувати NagiosQL або NConf, які пропонують засоби шаблонізації та підтримку великомасштабних топологій. Сам веб-інтерфейс системи, створений на основі CGI, хоч і функціональний, але потребує компіляції під час внесення змін, що ускладнює його модернізацію та адаптацію до сучасних стандартів веб-розробки. Проте відкритість архітектури Nagios дозволяє поступово замінювати його компоненти на сучасніші.

Для запуску Nagios потрібне середовище Linux або інша Unix-подібна система, компілятор мови C, веб-сервер на кшталт Apache, а також бібліотеки для графічного виводу, необхідні для генерації динамічних зображень. Конфігурація системи передбачає чотири типи текстових файлів, структура

яких схематично відображена на рисунку 3.5.

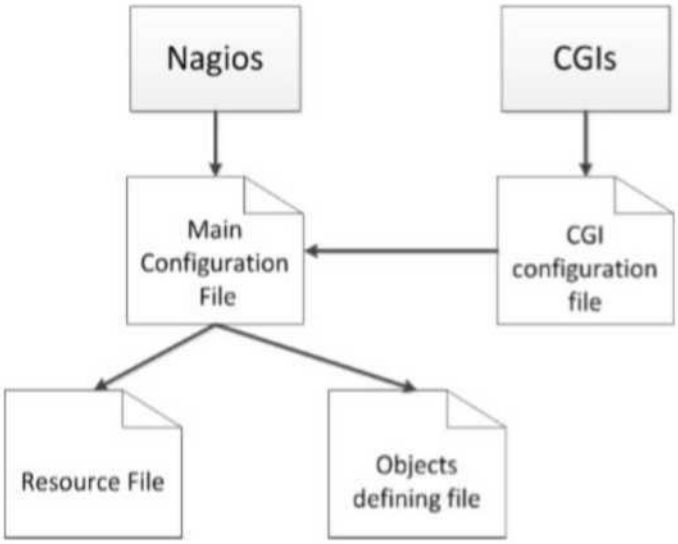


Рисунок 3.5 – Конфігураційні файли та їх взаємодія

У структурі системи моніторингу Nagios ключову роль відіграє конфігураційне забезпечення, яке складається з кількох логічно взаємопов’язаних файлів. Центральним елементом є головний конфігураційний файл, який містить усі необхідні директиви для запуску та функціонування демона Nagios. Цей файл виконує функцію керівного механізму, визначаючи параметри обробки подій, таймінги перевірок та маршрути доступу до інших складових конфігурації.

Окремий конфігураційний блок представлений файлом ресурсів, у якому зберігаються користувацькі макроси, що часто включають чутливу інформацію, зокрема паролі. Такий файл не є доступним для веб-інтерфейсу CGI, що забезпечує додатковий рівень безпеки при обробці конфіденційних параметрів.

Опис об’єктів системи моніторингу здійснюється через спеціалізовані конфігураційні файли, які містять логічні структури, що визначають параметри хостів, груп вузлів, контактів, сервісів та інших об'єктів, що підлягають контролю. У головному файлі конфігурації ці файли або каталоги

з такими описами включаються за допомогою директив cfg_file або cfg_dir. Такий підхід дозволяє масштабувати систему, додаючи нові об'єкти без необхідності змінювати основну структуру.

Ще однією важливою складовою є конфігураційний файл CGI, який містить налаштування, пов’язані із поведінкою веб-інтерфейсу. Через нього визначається, яким чином операційна система взаємодіє з CGI-компонентами, що забезпечують візуалізацію результатів моніторингу.

Механізм зворотного зв'язку в системі реалізовано через взаємодію з плагінами, які виконують заплановані запити до системи. Після завершення кожного запиту плагін повертає код стану, що інтерпретується ядром системи як показник поточного стану об’єкта моніторингу. Значення 0 свідчить про нормальну роботу (OK), 1 – про попереджувальну ситуацію (WARNING), а 2 – про критичний стан (CRITICAL). Така класифікація дозволяє системі оперативно реагувати на зміну стану інфраструктури та генерувати відповідні сповіщення для адміністратора.

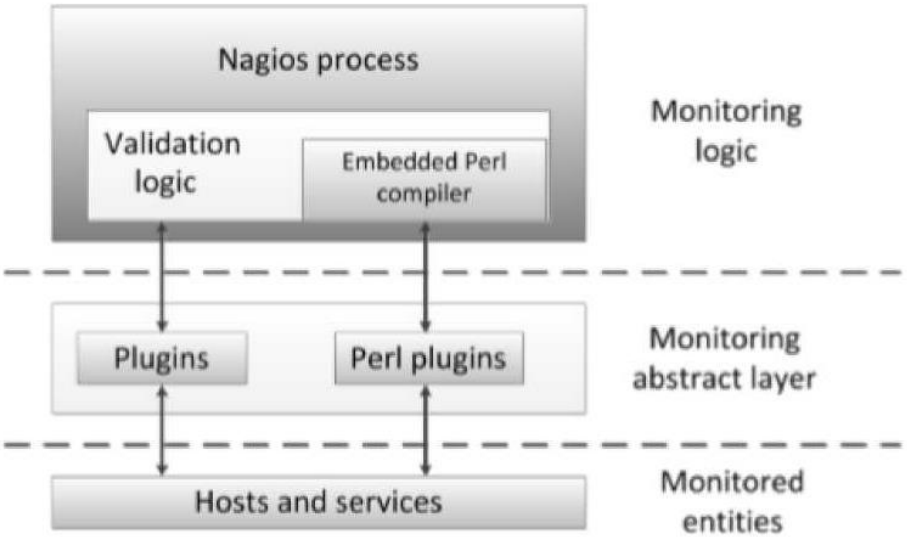


Рисунок 3.6 – Логіка монітору

У процесі реалізації моніторингу серверів у середовищі Nagios використовується підхід, що передбачає взаємодію з віддаленими агентами за допомогою спеціалізованих плагінів, які залежать від типу операційної

системи цільового пристрою. Для контролю вузлів, що функціонують під управлінням операційної системи Windows, застосовуються плагіни типу check_nt у поєднанні з агентом NSClient++, який забезпечує збирання та передачу необхідної інформації до центрального сервера моніторингу. Цей агент встановлюється на віддалену систему та надає доступ до базових метрик, зокрема використання процесора, пам'яті, дискового простору й статусу служб, що дозволяє здійснювати контроль за ключовими параметрами функціонування системи (див. рисунок 3.7).

У випадку систем, що працюють на базі Linux або UNIX-подібних операційних середовищ, передбачено використання плагіна check_nrpe, який працює в тандемі з агентом NRPE (Nagios Remote Plugin Executor). Цей підхід дозволяє виконувати розширені перевірки, запускаючи локальні плагіни безпосередньо на віддалених машинах. Завдяки цьому досягається більша гнучкість у моніторингу специфічних процесів, системних журналів, стану демонів та інших параметрів, критично важливих для підтримання стабільної роботи IT-інфраструктури (рисунок 3.8).

Обидва варіанти реалізації забезпечують централізовану фіксацію стану підконтрольних систем із можливістю оперативного інформування адміністратора про критичні відхилення, що є основою проактивного управління мережевими ресурсами.

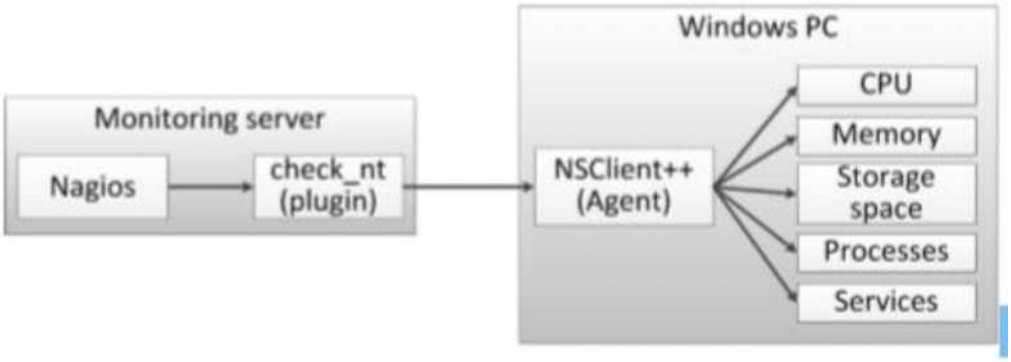


Рисунок 3.7 – Моніторинг хоста запуску Windows

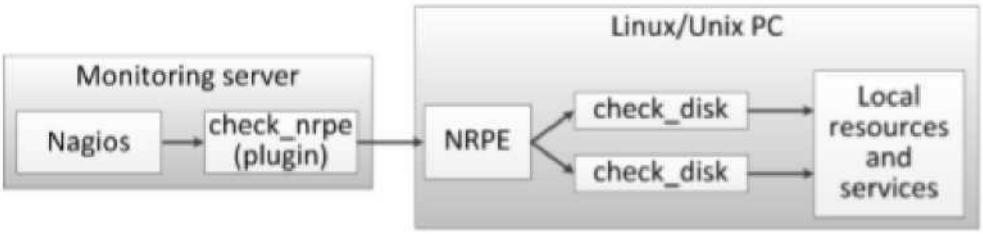


Рисунок 3.8 – Моніторинг хоста під керуванням Linux / Unix

NSClient++ є ефективним агентом моніторингу для операційних систем сімейства Windows, який був розроблений як сервіс для інтеграції з Nagios, прослуховуючи за замовчуванням TCP-порт 12489. Його гнучка архітектура дозволяє адаптувати агент до використання в поєднанні з іншими системами моніторингу, що робить його універсальним інструментом для збору телеметричних даних. Завдяки модульній побудові, NSClient++ підтримує запуск вбудованих перевірок, а також можливість виконання зовнішніх скриптів, що значно розширює спектр доступних функцій. Програмне забезпечення функціонує як фоновий процес (служба), надаючи набір плагінів для стандартних перевірок, які не вимагають додаткового налаштування з боку адміністратора. Цей агент сумісний із широким колом версій Windows, починаючи від NT4 і закінчуючи Windows 7 та Server 2008, де для його повноцінної роботи вимагається запуск із правами адміністратора.

З міркувань безпеки, NSClient++ може бути налаштований на використання зашифрованих з’єднань через протоколи SSH або SSL, що забезпечує захист передаваних даних при віддаленому моніторингу.

Для серверних платформ Linux і UNIX основним механізмом взаємодії з Nagios є використання плагіна check_nrpe, який вимагає встановлення агента NRPE (Nagios Remote Plugin Executor) на віддаленій системі. Хоча спочатку check_nrpe був орієнтований саме на UNIX-системи, він може також використовуватися спільно з NSClient++, підключаючись через порт TCP 5666.

Порівняльний аналіз NSClient++ та NRPE виявляє ключову перевагу першого – наявність великої кількості вбудованих перевірок, які зменшують потребу в написанні додаткових скриптів. Натомість NRPE передбачає повністю скриптову організацію перевірок, що надає високу гнучкість, але вимагає більше зусиль для налаштування. Таким чином, вибір між цими двома агентами зазвичай визначається специфікою операційного середовища, рівнем необхідної кастомізації та вимогами до безпеки.

3.5 Реалізація системи

На основі проведеного аналізу технічного завдання була побудована загальна структурна схема функціонування системи, яка відображає логіку взаємодії її основних компонентів. Архітектура передбачає наявність серверної файлової системи, централізованої бази даних і трьох функціональних програмних модулів. До них належить веб-клієнт, мобільний клієнт, орієнтований на платформу Android, а також серверний компонент, відповідальний за обробку запитів, координацію взаємодії клієнтів із ресурсами системи й керування файловою структурою.

Серверна частина здійснює обслуговування запитів як від веб-, так і мобільного клієнта, забезпечуючи доступ до збережених даних та інтеграцію з файловою системою. Центральна база даних виконує роль сховища метаінформації про користувачів і розміщення відповідних файлів на сервері, забезпечуючи ефективне управління цими об'єктами. З огляду на потребу у високій швидкодії запитів і структурованості даних, для реалізації обрано реляційну систему керування базами даних PostgreSQL, яка надає необхідні інструменти для реалізації складних запитів і гарантує цілісність та узгодженість інформації.

На рисунку 3.9 зображено логічну структуру бази даних, що містить таблиці з інформацією про користувачів, файлові ресурси, шляхи доступу до них та пов’язані атрибути. Між таблицями визначено відповідні зв’язки, що

забезпечують цілісність і логіку взаємозв’язків усередині інформаційної моделі системи.

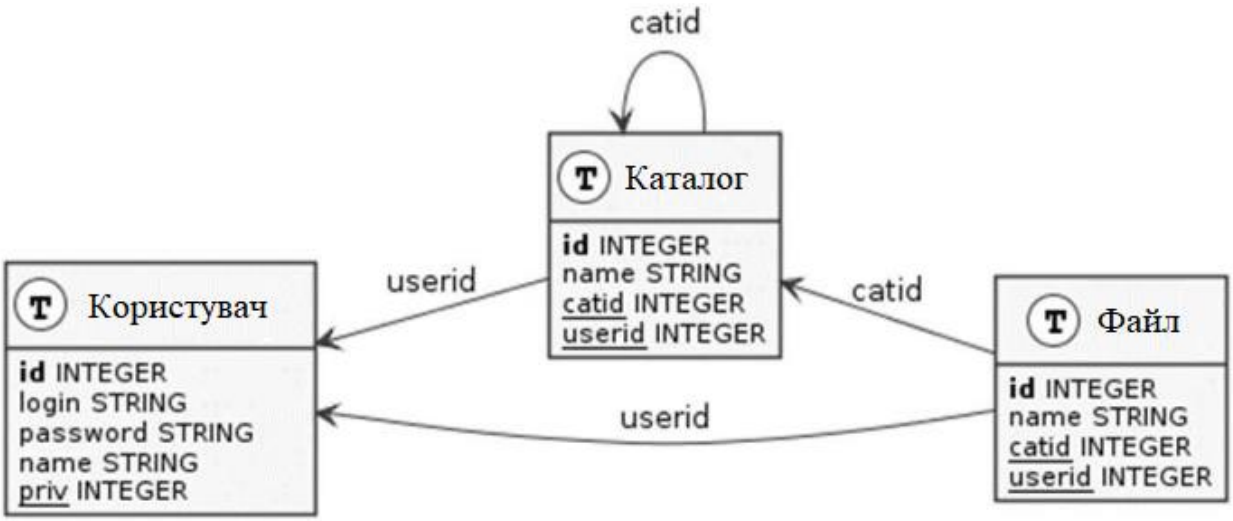


Рисунок 3.9 – Схема БД та зв’язків між таблицями

Серверна частина системи реалізується у вигляді HTTP-сервера, що виступає як основний посередник між клієнтськими додатками – вебінтерфейсом і мобільним застосунком для Android. Взаємодія здійснюється через обмін HTTP-запитами, які клієнт надсилає за допомогою стандартних методів протоколу: POST для створення нових записів або файлів, GET для отримання даних, DELETE для їх видалення та UPDATE (частіше як PUT або PATCH) для модифікації наявної інформації.

У процесі авторизації дані користувача зберігаються локально: у браузері – через cookie або localStorage, а на мобільному пристрої – в кеші або внутрішньому сховищі, що дозволяє підтримувати сесію користувача протягом заданого періоду часу. Таким чином, при повторному запиті система може автоматично розпізнати користувача без необхідності повторного введення облікових даних.

У разі надходження запиту від авторизованого клієнта серверна частина виконує послідовність дій, описану функціональною схемою, що включає:

- перевірку автентичності запиту та відповідність користувача наданим токенам доступу;

- ідентифікацію типу запиту та маршрутизацію до відповідного обробника на сервері;

- здійснення запиту до бази даних у випадку потреби у доступі до метаінформації про файли або користувача;

- надання запитуваних файлів або відповідних даних клієнтові, або виконання дії, визначеної типом запиту;

- генерацію відповіді у форматі JSON або іншому, узгодженому з клієнтським застосунком, з вказівкою статусу виконання операції.

Цей підхід дозволяє забезпечити гнучку, масштабовану і безпечну архітектуру серверної частини, здатну обслуговувати одночасну взаємодію з кількома типами клієнтів в реальному часі.

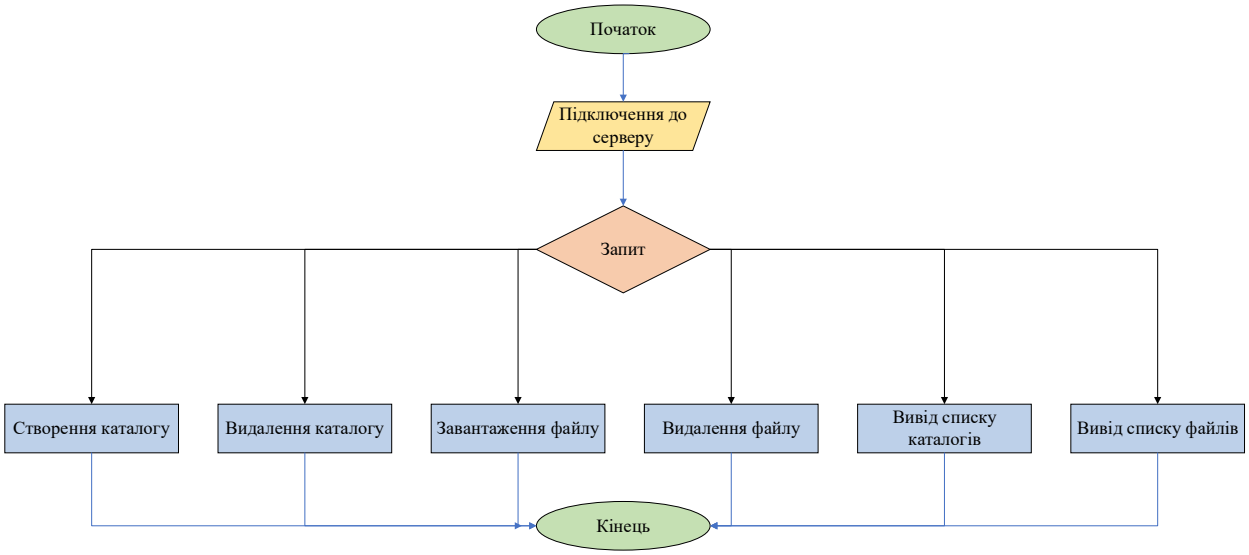


Рисунок 3.10 – Схема роботи серверного застосунку для звичайного користувача

У даній системі в ролі сервера використовується одноплатний комп’ютер Raspberry Pi Model B, що представлений на рисунку 3.11. На цей мініатюрний пристрій інстальовано операційну систему ArchLinux ARM, яка оптимізована під архітектуру ARM і забезпечує високий рівень

контрольованості, легкість, а також мінімальне використання апаратних ресурсів, що особливо важливо для малопотужних платформ.

Для забезпечення роботи серверної частини було встановлено компілятор мови програмування Go, що дозволяє створювати ефективні серверні застосунки з високою продуктивністю та низьким споживанням пам’яті. Використання Go забезпечує просту багатопоточну обробку запитів завдяки вбудованим механізмам паралелізму, що особливо доцільно в умовах обмежених ресурсів пристрою.

Крім того, у якості веб-сервера застосовується nginx – потужний і легкий HTTP-сервер, який виконує функції обробки вхідних HTTP-запитів, маршрутизації, балансування навантаження та проксирування. Завдяки своїй ефективній обробці великої кількості одночасних з’єднань, nginx є оптимальним вибором для інтеграції із застосунками, написаними на Go.

Таким чином, вибір Raspberry Pi Model B з ArchLinux ARM, nginx та Go дозволяє реалізувати повноцінну серверну платформу, здатну підтримувати веб-клієнти, Android-додатки та здійснювати зберігання, обробку й передачу даних у реальному часі при мінімальних апаратних витратах.

Рисунок 3.11 – Апаратний сервер та операційна система

Серверна частина програмного комплексу реалізована у вигляді HTTP-сервера, написаного мовою програмування Go, що забезпечує обробку запитів від клієнтської сторони та повернення відповідей у форматі JSON. Архітектура побудована на основі MVC-підходу (Model-View-Controller), який дозволяє чітко розділити функціональність між логікою доступу до даних (model), обробкою запитів (controller) та представленням (view). Це сприяє підтримуваності, масштабованості та зручності подальшого розширення серверного додатку.

Клієнтська частина системи реалізована за допомогою фреймворку Vue.js, що є сучасним інструментом для створення реактивних вебінтерфейсів. Розробка здійснюється на мові JavaScript із використанням компонентного підходу, де кожен екран або функціональний блок інтерфейсу представлений окремим .vue-файлом. Структура клієнтського застосунку базується на принципі вкладених компонентів, що дозволяє реалізувати гнучку та масштабовану систему з високим рівнем повторного використання коду та ефективною реакцією на зміну стану даних.

Такий підхід забезпечує чіткий поділ відповідальності між клієнтом та сервером, де сервер відповідає за обробку логіки та даних, а клієнт – за взаємодію з користувачем і динамічне відображення інформації у веб-інтерфейсі.

Arch Linux ARM є безпечним, відкритим дистрибутивом операційної системи Linux, орієнтованим на високий рівень гнучкості, легкості та стабільності. Його основною особливістю є підтримка архітектури ARM, що робить його придатним для використання на енергоефективних одноплатних комп’ютерах, таких як Raspberry Pi. Цей дистрибутив дотримується принципу ковзного випуску (rolling release), що дозволяє користувачеві постійно мати доступ до найактуальніших стабільних версій програмного забезпечення без необхідності повної перевстановлення системи.

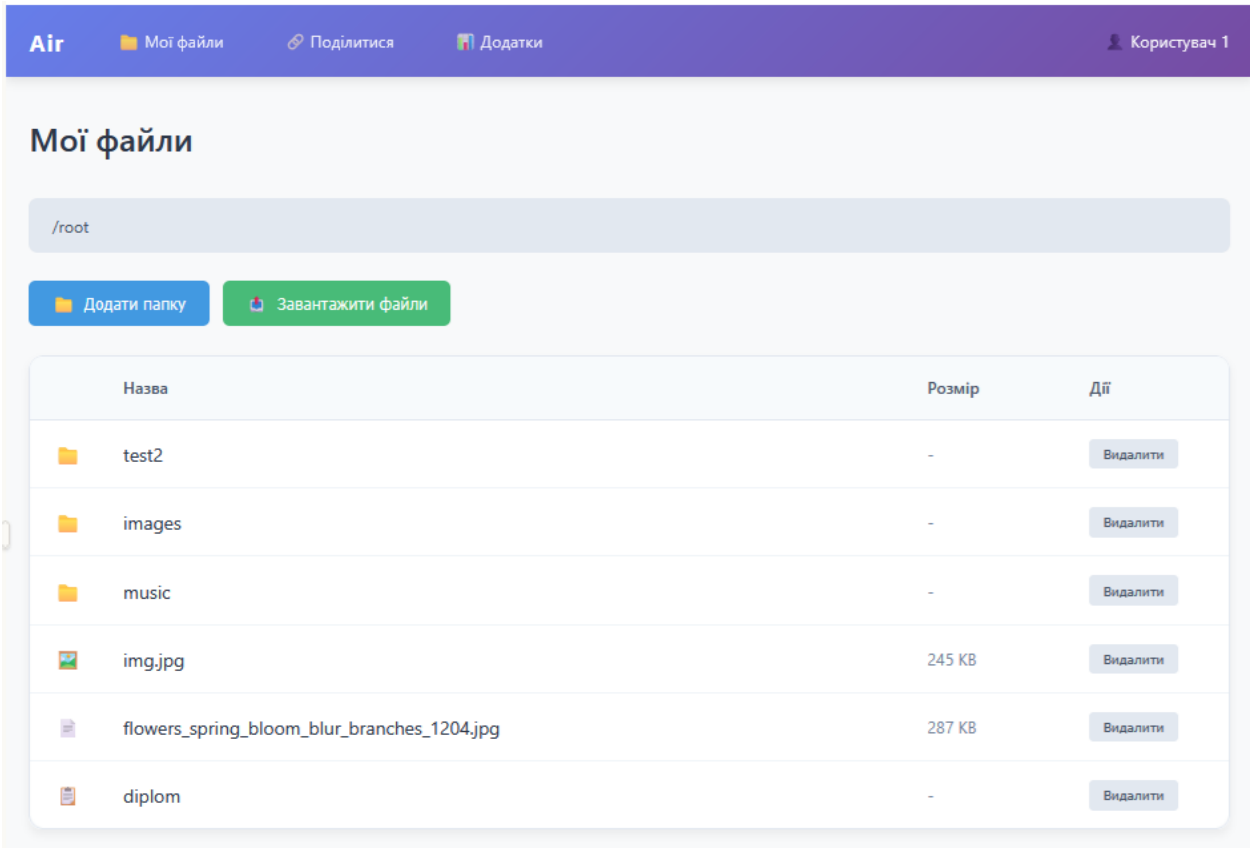


Рисунок 3.12 – Результати роботи

Операційна система Arch Linux ARM широко підтримується як офіційними джерелами, так і активною спільнотою, яка забезпечує розширений доступ до великої кількості пакунків через відповідні репозиторії. Завдяки своїй архітектурі дистрибутив ідеально підходить для гнучкої конфігурації системи, де користувач самостійно формує необхідне програмне середовище, виходячи з конкретних потреб, що особливо цінно для проєктів у сфері вбудованих систем, автоматизації, моніторингу та IoT-рішень.

ВИСНОВКИ

В результаті виконання випускної кваліфікаційної роботи була розроблена система зберігання даних на основі одноплатного комп'ютера Raspberry Pi. Серверна частина КС реалізована у вигляді HTTP-сервера, написаного мовою програмування Go, що відповідає за обробку запитів клієнтів та повернення відповідей у форматі JSON. Архітектура серверного програмного забезпечення базується на парадигмі Model-View-Controller, що забезпечує чіткий розподіл логіки між обробкою запитів, представленням даних та їхньою моделлю.

Клієнтська частина представлена двома застосунками: веб-інтерфейсом, розробленим на основі JavaScript-фреймворку Vue.js, та мобільним застосунком для операційної системи Android. Обидва застосунки взаємодіють із сервером через мережеві запити методами POST, GET, DELETE та UPDATE, використовуючи стандартну авторизацію та зберігаючи відповідну інформацію на стороні клієнта.

У системі використовується реляційна база даних PostgreSQL, яка виконує роль сховища для структурованої інформації про користувачів і файли, розміщені на сервері. Взаємодія із файловою системою Raspberry Pi забезпечує повний цикл доступу до необхідних ресурсів, реалізуючи функціональну модель централізованого зберігання даних з віддаленим доступом. Схема демонструє логічну взаємодію між усіма компонентами системи в контексті розподіленого клієнт-серверного середовища.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. A. Greenberg et al., ‘‘VL2: A scalable and flexible data center network’’ ACM SIGCOMM Comput. Commun. Rev., vol. 39, no. 4, pp. 51-62, Oct. 2009.

2. C. Guo, H. Wu, K. Tan, L. Shi, Y Zhang, and S. Lu, ‘‘DCell: A scalable and fault- tolerant network structure for data centers,’’ ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 4, pp. 75-86, Oct. 2008.

3. Hung LeHong, Jackie Fenn. Key Trends to Watch in Gartner 2012 Emerging Technologies Hype Cycle

4. Cisco Systems, Inc., "Enterprise Campus 3.0 Architecture: Overview and Framework," 2008. [Електронний ресурс] - Режим доступу до ресурсу: http://www. cisco.com/en/US/docs/solutions/Enterprise/Campus/campover.html.

5. J. Case, M. Fedor, M. Schoffstall and J. Davin, "RFC1157: A Simple Network Management Protocol (SNMP)," 5 1990. [Електронний ресурс] - Режим доступу до ресурсу: http://www.ietf.org/rfc/rfc1157.txt.

6. Kochlan, M.; Hodon, M.; Cechovic, L.; Kapitulik, J.; Jurecka, M., “WSN for traffic monitoring using Raspberry Pi board,” Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on, vol., no., pp.1023,1026.

7. Sophon Mongkolluksamee, Panita Pongpaibool, Chavee Issariyapat, “Strengths and Limitations of Nagios as a Network Monitoring Solution” Proceedings of the 7th International Joint Conference on Computer Science and Software Engineering (JCSSE 2010) Vol. 1, pp. 96-101, Bangkok, Thailand, May 2010

8. Ahmed D. Kora and Moussa Moindze Soidridine, “Nagios based enhanced IT management system,” Interntional Journal of Engineering Science and Technology, vol. 4, no. 3, pp. 818-822, 2012.