МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук і технологій
Кафедра комп'ютерних систем та мереж

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
Жуков Ігор Анатолійович
"_____"_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)
ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ «БАКАЛАВР»
ЗА СПЕЦІАЛЬНІСТЮ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

Тема: «Розробка та дослідження системи моніторингу технічного стану серверного обладнання»

Виконавець: Никитенко Даниїл Леонтійович

Керівник: д.т.н., професор Гільгурт Сергій Якович

Нормоконтролер: Журавель С.В.
(підпис)

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет комп’ютерних наук і технологій

Кафедра комп’ютерних систем та мереж

Освітній ступінь: «Бакалавр»

Галузь знань, спеціальність, освітньо-професійна програма:

12 «Інформаційні технології», 123 «Комп’ютерна інженерія», «Комп’ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Жуков Ігор Анатолійович

“_____”_____2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи

Никитенка Даниїла Леонтійовича

(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. **Тема кваліфікаційної роботи:** «Розробка та дослідження системи моніторингу технічного стану серверного обладнання» затверджена наказом ректора №591/ст від 01.05.2023 р.
2. **Термін виконання роботи**: 15.05.2023 – 25.06.2023
3. **Вихідні дані до роботи**: дані про системи моніторингу, документація, засоби та технологія створення систем моніторингу.
4. **Зміст пояснювальної записки**: вступ, аналіз предметної області, проектування системи моніторингу огляд використаних технологій в ході розробки системи моніторингу, процес розробки системи моніторингу.
5. **Перелік обов'язкового графічного (ілюстративного) матеріалу:** презентація *Power Point*.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Проаналізувати джерела за темою кваліфікаційної роботи та зробити аналіз предметної області.	15.05.2023-19.05.2023	Виконано
2	Написати перший розділ кваліфікаційної роботи.	22.05.2023-23.05.2023	Виконано
3	Провести аналіз вимог користувача та сформулювати опис екранів застосунку, обрати інструменти розробки.	24.05.2023-26.05.2023	Виконано
4	Написати другий розділ кваліфікаційної роботи.	29.05.2023-02.06.2023	Виконано
5	Розробити дизайн екранів музичного застосунку та створити музичний застосунок.	03.06.2023-05.06.2023	Виконано
6	Написати третій розділ кваліфакційної роботи та зробити висновки.	05.06.2023-08.06.2023	Виконано
7	Оформлення та друк пояснювальної записки кваліфікаційної роботи.	12.06.2023-13.06.2023	Виконано
8	Підготовка презентації та доповіді для виступу.	14.06.2023-16.06.2023	Виконано
9	Захист кваліфікаційної роботи.	19.06.2023-23.06.2023	Виконано

7. Дата видачі завдання: «15» травня 2023 р.

Керівник кваліфікаційної роботи _____ Гільгурт С. Я.

(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Никитенко Д. Л.

(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи “Розробка та дослідження системи моніторингу технічного стану серверного обладнання ”: 43 с., 19 рис., 10 літературних джерел, 1 додаток.

СИСТЕМА МОНІТОРИНГУ, СЕРВЕР, МОНІТОРИНГУ СЕРВЕРУ.

Мета кваліфікаційної роботи – проаналізувати основні методи моніторингу серверів та розробити програмний додаток для моніторингу технічного стану серверного обладнання.

Об’єкт проєктування – Моніторинг технічного стану серверного обладнання.

Предмет проєктування – Система моніторингу технічного стану серверного обладнання на базі *Telegram*-бота.

Метод проєктування – визначення основних методів моніторингу технічного стану серверного обладнання.

Прогнози припущення щодо розвитку об’єкта дослідження – створення робочого зразка програми та використання його в серверних мережних комп’ютерних системах за умови можливості запуску програми на сервері.

Результати кваліфікаційної роботи рекомендується використовувати при розробці нових програмних засобів, які надають можливість моніторити технічний стан серверного обладнання.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ,ТЕРМІНІВ 7
ВСТУП 8
РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ 9
1.1.Визначення концепції моніторингу технічного стану серверів. 9
1.2 Огляд існуючих техінічних рішень 10
1.2.1Nagios 12
1.2.2. Prometheus 15
1.3Висновки до 1 розділу 16
2.1Вимоги до системи моніторингу 18
2.1.1Функціональні вимоги 18
2.1.2Нефункціональні вимоги 19
2.2 Архітектура системи моніторингу 20
2.3 Види архітектур 21
2.3.1 P2P архітектура 21
2.3.2 Клієнт-клієнтська архітектура 23
2.3.3 Клієнт-серверна архітектура 24
2.4Вибір архітектури 25
2.5 Технології для створення системи моніторингу 26
2.5.1 Python 26
2.5.2PostgreSQL 27
2.5.3 API та бібліотеки 27
2.6Висновки до 2 розділу 29
РОЗДІЛ 3 РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ ТЕХНІЧНОГО СТАНУ СЕРВЕРНОГО ОБЛАДНАННЯ 31
3.1Розробка БД 31
3.2 Створення телеграм бота 33
3.3Створення функцій роботи з БД 34
3.3 Створення функцій моніторингу 35
3.4Тестування системи 38
3.5 Висновки до 3розділу 40
ВИСНОВКИ 41

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ..... 43

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ,ТЕРМІНІВ

SSH – безпечна оболонка (Secure Shell)

VPN – віртуальна приватна мережа (Virtual Private Network)

IP – інтернет протокол (Interner Protocol)

API – інтерфейс програмного забезпечення(Aplication Programing Interface)

БД – база даних

ВСТУП

У сучасному інформаційному суспільстві комп’ютерні системи є невід’ємною частиною багатьох сфер діяльності, зокрема у сфері бізнесу, науки, освіти та адміністрування. Серверне обладнання, що використовується для забезпечення роботи цих систем, є основною складовою успішного функціонування сучасних інформаційних технологій.

Однак серверне обладнання вимагає постійного контролю технічного стану, оскільки некоректна робота обладнання може призвести до значних втрат у вигляді призупинення роботи, втрати даних та погіршення репутації організації. У зв’язку з цим, розробка та тестування системи моніторингу технічного стану серверного обладнання є актуальною та важливою задачею, яка допоможе забезпечити стабільну та безперебійну роботу інформаційних систем. Системи моніторингу дозволяють оперативно контролювати стан обладнання, виявляти проблеми на ранніх стадіях та приймати вчасні заходи для їх усунення.

Ефективна система моніторингу допомагає знизити час простою системи, покращити її надійність та забезпечити неперервну роботу інформаційної інфраструктури.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1.Визначення концепції моніторингу технічного стану серверів.

Моніторинг став загальним терміном, чий зміст сильно залежить від контексту. В широкому сенсі це відноситься до процесу стеження за станом системи. Це робиться двома способами: проактивним і реактивним. Перший включає спостереження за візуальними показниками, такими як часові ряди та панелі управління, і іноді саме цим адміністратори мають на увазі, коли говорять про моніторинг. Другий включає автоматизовані способи повідомлення операторам для привернення їхньої уваги до серйозних змін у стані системи; це зазвичай називається сповіщенням. [1]

Система моніторингу об’єднує в собі ці 2 способи. Це програмне забезпечення або набір інструментів, які використовуються для постійного контролю та відстеження стану серверів. Вона слідкує за різними аспектами роботи серверів, такими як навантаження процесора, використання пам'яті, дисків, мережі, а також доступність веб-сайтів або служб, що запускаються на серверах.[4]

Система моніторингу серверів може виявляти проблеми, такі як відмова обладнання, перевищення ресурсів, помилки програмного забезпечення, зупинення служб і забезпечувати операторам або адміністраторам серверів сповіщення про такі події. Вона також може збирати статистику про роботу серверів, допомагати у виявленні тенденцій і плануванні масштабування інфраструктури[9]

Основні принципи моніторингу технічного стану серверів

1. Збір даних: Моніторинг технічного стану серверів передбачає постійний збір різноманітних даних про сервери, включаючи апаратне забезпечення, операційну систему, мережеві підключення, навантаження та інші параметри, які можуть вказувати на потенційні проблеми

2. Аналіз даних: Зібрані дані підлягають подальшому аналізу для виявлення аномалій, прогнозування відмов або ідентифікації проблемних зон. Аналітичні алгоритми та правила можуть застосовуватися для виявлення відхилень від норми, порівняння з попередніми даними, ідентифікації трендів та специфічних проблем. Результати аналізу можуть бути використані для генерації сповіщень, створення звітів та прийняття рішень щодо управління серверами.
3. Відображення та візуалізація: Одним із важливих аспектів концепції моніторингу технічного стану серверів є можливість відображення зібраних даних у зрозумілій формі. Графіки, діаграми, панелі управління та інші інтерактивні інструменти дозволяють операторам та адміністраторам серверів швидко оцінити стан системи, виявити аномалії та прийняти необхідні заходи. Ефективна візуалізація сприяє зрозумілості та швидкому реагуванню на проблеми.
4. Сповіщення та реагування: Моніторинг технічного стану серверів повинен мати можливість надсилати сповіщення про виявлені проблеми або аномалії. Це можуть бути електронні повідомлення, *SMS*-повідомлення, веб-повідомлення або інші канали комунікації. Крім того, моніторинг може надавати можливості автоматичного реагування на проблеми, такі як перезавантаження сервера, запуск резервного копіювання або зміна конфігурації.
5. Інтеграція та розширення: Концепція моніторингу технічного стану серверів повинна бути гнучкою та легко розширюваною. Вона повинна мати можливість інтегруватися з існуючими інфраструктурними компонентами, які можуть забезпечувати додаткову інформацію або функціональність. Розширення може включати нові типи метрик, алгоритми аналізу, інтеграцію з іншими системами моніторингу тощо.

1.2Огляд існуючих техінічних рішень

На сьогоднішній день існує широкий спектр технічних рішень для моніторингу технічного стану серверів. В цьому підрозділі розглянуті деякі з них, що заслуговують уваги:

1. Системи моніторингу з відкритим вихідним кодом: Існує кілька популярних систем моніторингу з відкритим вихідним кодом, таких як Nagios, Zabbix, Icinga та Munin. Ці рішення надають розширені можливості зібрання, аналізу та відображення даних про сервери. Вони підтримують широкий спектр протоколів та інтеграцію з іншими системами. Один з переваг використання систем з відкритим вихідним кодом полягає в можливості налаштування та розширення функціональності під конкретні потреби організації.
2. Комерційні системи моніторингу: На ринку також присутні комерційні системи моніторингу, такі як SolarWinds, Dynatrace, New Relic та Splunk. Ці рішення надають розширені функціональні можливості, включаючи моніторинг в реальному часі, аналітику широкого спектру даних та інтеграцію з хмарними сервісами. Комерційні системи часто мають готові рішення для конкретних сценаріїв моніторингу і надають високий рівень підтримки користувачів.
3. Хмарні сервіси моніторингу: Зростання популярності хмарних обчислень призвело до розвитку хмарних сервісів моніторингу, таких як Amazon CloudWatch, Microsoft Azure Monitor та Google Cloud Monitoring. Ці сервіси дозволяють моніторити сервери, які розгорнуті в хмарних середовищах, та надають готові інструменти для збору, аналізу та відображення даних про сервери в хмарному середовищі. Вони забезпечують високу масштабованість, надійність та інтеграцію з іншими хмарними сервісами.
4. Агент-орієнтовані системи моніторингу: Деякі системи моніторингу використовують агенти, які розміщуються безпосередньо на серверах для збору даних про їх технічний стан. Прикладами таких систем є Prometheus та Datadog. Агенти надають додаткову гнучкість та можливості збору даних на рівні операційної системи та додатків. Вони можуть забезпечувати більш детальну інформацію про роботу серверів, включаючи метрики продуктивності, журнали подій та статус служб.

1.2.1Nagios

Nagios є однією з провідних систем моніторингу серверів. Вона надає широкий спектр функціональних можливостей, зокрема моніторинг доступності серверів, використання ресурсів, реагування на незвичайні події та надсилання сповіщень. *Nagios* базується на архітектурі клієнт-сервер, де агенти забезпечують збір інформації з серверів та передають її до центрального сервера для подальшої обробки.

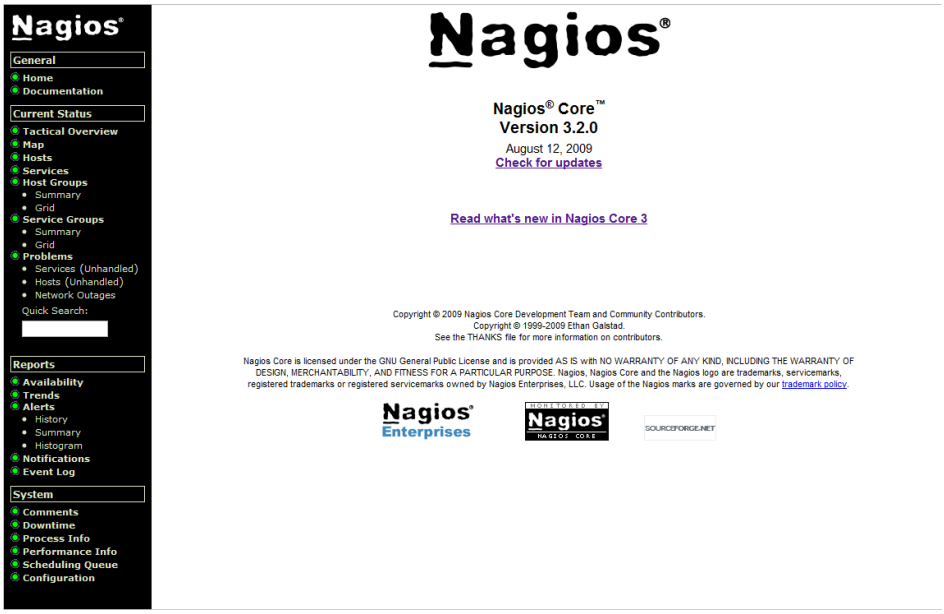

Рис1.1 Головний екран веб-інтерфейсу *Nagios*

Система моніторингу *Nagios* надає ряд функціональних можливостей для моніторингу технічного стану серверів. Основні можливості *Nagios* включають:

1. **Моніторинг доступності**: *Nagios* може перевіряти доступність серверів та мережевих пристроїв, перевіряти доступність веб-сайтів за допомогою HTTP-запитів і контролювати рівень доступності системи.
2. **Моніторинг використання ресурсів**: *Nagios* може вимірювати та моніторити використання ресурсів, таких як процесор, оперативна пам'ять, дисковий простір і мережевий трафік.
3. **Графіки та візуалізація**: *Nagios* надає можливість створювати графіки та візуалізацію метрик для аналізу та моніторингу. Це допомагає зрозуміти зміни у використанні ресурсів і виявити тренди.
4. **Конфігурація та розширення**: *Nagios* має гнучку систему конфігурації, яка дозволяє налаштовувати моніторинг для різних серверів і послуг. Він також підтримує

можливість розширення функціональності за допомогою додаткових плагінів та розширень.

5. **Журнали та історія**: *Nagios* зберігає журнали та історію моніторингу, що дозволяє відстежувати зміни стану серверів та аналізувати проблеми, які виникали в минулому.
6. **Розподілена архітектура**: Nagios може працювати в розподіленому середовищі, що дозволяє масштабувати систему та забезпечувати високу доступність.
7. **Групування та ієрархія**: Nagios дозволяє групувати сервери та пристрої в логічні групи та створювати ієрархію. Це полегшує організацію та управління моніторингом для комплексних інфраструктур.
8. **Планування**: *Nagios* дозволяє планувати моніторингові завдання і розкладувати їх залежно від потреб. Це може бути корисним для виконання специфічних моніторингових операцій в певний час або з інтервалами.
9. **Розширені засоби моніторингу мережі**: *Nagios* має можливості для моніторингу мережевих пристроїв, таких як комутатори, маршрутизатори, файрволи тощо. Він може контролювати стан портів, рівень трафіку та інші параметри мережевої інфраструктури.
10. **Перевірка конфігурації**: *Nagios* надає засоби перевірки конфігурації, що допомагають виявити помилки або неправильні налаштування. Це дозволяє забезпечити точність та надійність моніторингової системи.
11. **Інтеграція з іншими системами**: *Nagios* може інтегруватися з іншими системами, такими як системи управління подіями (SIEM), системи керування конфігураціями (CM), системи тікетів та інші. Це дозволяє забезпечити взаємодію та обмін даними між різними системами.
12. **Спостереження за розподіленими системами**: *Nagios* має можливості для спостереження за розподіленими системами та серверами, що дозволяє контролювати технічний стан серверів, що розташовані на різних місцях.

13. **Можливості розширення за допомогою API**: Nagios надає *API* для розширення функціональності та інтеграції зі сторонніми додатками. Це дозволяє розробникам створювати власні розширення та рішення, що відповідають їхнім потребам.

14. **Резервне копіювання та відновлення**: Nagios дозволяє створювати резервні копії конфігураційних файлів та історичних даних моніторингу. Це забезпечує можливість відновлення системи після випадкового збою або втрати даних.

Nagios є популярною системою моніторингу, але вона також має деякі недоліки:

- Складність налаштування: Одним з головних недоліків Nagios є складність налаштування. Ця система вимагає досвіду та знань у сфері системного адміністрування. Конфігурація і налаштування моніторингових правил можуть бути складними та часоємкими завданнями, особливо при масштабуванні інфраструктури.
- Складність розгортання: Розгортання Nagios також може бути складним процесом, особливо при інтеграції з іншими системами та налаштуванні мережевого доступу. Це може вимагати певних знань мережевої архітектури та інфраструктури.
- Обмежена масштабованість: Nagios може мати обмеження в масштабуванні при великому обсязі даних або великій кількості моніторингованих об'єктів. При збільшенні обсягу даних або кількості моніторингових об'єктів може виникнути необхідність в розгортанні багатоекземплярного середовища або інших рішень для забезпечення продуктивності.
- Відсутність реального часу: Nagios базується на понятті перевірок у заданих інтервалах часу. Це означає, що спостереження за системою відбувається не в реальному часі, а з певним затримкою. В разі виникнення проблеми, яка виникає між перевірками, Nagios може неспроможний вчасно виявити цю проблему.
- Обмежені можливості візуалізації: Вбудований інтерфейс Nagios пропонує базовий рівень візуалізації даних. Хоча можливості візуалізації можна розширити за допомогою плагінів та інтеграції з іншими інструментами, навігація та сприйняття даних за допомогою вбудованого інтерфейсу можуть бути обмеженими.

1.2.2. Prometheus

Prometheus - це потужна система моніторингу з відкритим вихідним кодом, яка широко використовується для моніторингу технічного стану серверів та інфраструктури. Основна перевага Prometheus полягає в його гнучкості, простоті налаштування та розширення, що робить його популярним вибором для багатьох організацій.[11]

Основні можливості Prometheus включають:

1. Збір даних: Prometheus надає механізми для збору даних з різних джерел. Він підтримує власний формат метрик, відомий як мова запитів Prometheus (PromQL). Крім того, він підтримує такі протоколи як HTTP, SNMP, JMX та інші, що дозволяє збирати дані з різних типів серверів та сервісів.[3]
2. Зберігання даних: Prometheus має вбудовану базу даних для зберігання зібраних даних про метрики. Це дозволяє зберігати дані від кількох годин до кількох років, залежно від налаштувань ретенції. База даних Prometheus дуже ефективна та оптимізована для швидкого доступу до даних.[3]
3. Запити та аналіз даних: За допомогою мови запитів PromQL можна виконувати потужні запити до зібраних даних. Це дозволяє аналізувати та візуалізувати дані за різними параметрами, виконувати агрегацію, фільтрацію та розрахунки. Завдяки цим можливостям, Prometheus дозволяє отримувати детальну інформацію про стан серверів та інфраструктури.
4. Візуалізація даних: Prometheus надає можливості для візуалізації зібраних даних за допомогою інтеграції з іншими інструментами, такими як Grafana. Графіки, діаграми та інші типи візуалізації допомагають спостерігати та аналізувати дані про сервери в зручному та зрозумілому форматі.
5. Сповіщення та оповіщення: Prometheus має можливості для налаштування правил сповіщень та оповіщень на основі визначених порогових значень метрик. Це дозволяє операторам системи своєчасно виявляти проблеми та приймати відповідні заходи.
6. Розширення та інтеграція: Prometheus має широкий набір інструментів, бібліотек та інтерфейсів для розширення та інтеграції з іншими системами. Це

дозволяє розробникам створювати власні інструменти, додатки та розширення, що задовольняють конкретні потреби організації.

Хоча Prometheus є потужною системою моніторингу, вона також має свої недоліки:

- Масштабованість: Prometheus може мати обмеження в масштабуванні при великому обсязі даних або великій кількості серверів. При збільшенні обсягу даних або кількості об'єктів може знадобитися розгортання кластера Prometheus для забезпечення продуктивності та швидкодії.
- Зберігання даних: Вбудована база даних Prometheus може стати обмеженням при довгостроковому зберіганні великої кількості даних. Для зберігання даних протягом тривалого періоду можуть знадобитися додаткові рішення або інтеграція з зовнішніми системами зберігання.
- Витрата ресурсів: Prometheus може вимагати значних ресурсів (процесора, пам'яті, дискового простору) для зберігання та обробки даних. При моніторингу великого масштабу інфраструктури це може призвести до необхідності масштабування ресурсів моніторингового сервера.
- Нестійкість мережі: Якщо між серверами Prometheus та моніторингованими об'єктами відсутня стабільна мережева зв'язок, це може призвести до втрати даних моніторингу. Prometheus не має механізму реплікації або резервного копіювання даних за замовчуванням, тому це може становити ризик для доступності та цілісності даних.
- Налаштування та управління: Налаштування та управління Prometheus можуть вимагати певного рівня експертизи та знань з системного адміністрування. Налаштування правил сповіщень, аналіз даних та візуалізація можуть потребувати часу та зусиль для досягнення оптимальних результатів.

1.3Висновки до 1 розділу

У даному розділі було проведено аналітичний огляд предметної області, який спрямований на вивчення основних аспектів моніторингу технічного стану серверного обладнання. Проведений аналіз дозволив отримати наступні висновки:

1. Серверне обладнання є важливою складовою інфраструктури будь-якої компанії чи організації, що забезпечує функціонування різноманітних додатків та послуг. Забезпечення безперебійної роботи серверів має вирішальне значення для бізнесу і може вплинути на продуктивність, надійність та безпеку даних.
2. Моніторинг технічного стану серверного обладнання є необхідним елементом управління і підтримки серверної інфраструктури. Відстеження параметрів, таких як температура, напруга, завантаження процесора, обсяг використаної пам'яті та інші, дозволяє оперативно виявляти проблеми та вживати відповідних заходів для їх вирішення.
3. Існує багато комерційних та відкритих систем моніторингу, які надають зручний інтерфейс для збору, аналізу та відображення даних про стан серверного обладнання. Вони можуть мати різноманітний функціонал і можуть бути адаптовані до потреб конкретної організації.
4. Розробка власної системи моніторингу може мати переваги, зокрема, повний контроль над функціоналом і адаптація до специфічних вимог організації. Проте це також може бути витратною та часозатратною задачею, вимагаючи глибоких знань інфраструктури серверів і програмування.

Загалом, проведений аналітичний огляд дозволив зрозуміти важливість моніторингу технічного стану серверного обладнання, ознайомитися з наявними системами моніторингу та виявити необхідність розробки власної системи для конкретних потреб організації. Дані висновки становлять основу для подальшої розробки та дослідження системи моніторингу технічного стану серверного обладнання в рамках дипломної роботи

РОЗДІЛ 2
ПРОЄКТУВАННЯ СИСТЕМИ МОНІТОРИНГУ СТАНУ СЕРВЕРНОГО ОБЛАДНАННЯ

2.1Вимоги до системи моніторингу

2.1.1Функціональні вимоги

1.Збір інформації про серверне обладнання

- Система повинна автоматично збирати дані про технічний стан серверного обладнання, включаючи параметри, такі як доступність, завантаженість оперативної пам’яті, завантаженість диску тощо.
- Система повинна забезпечувати регулярне оновлення даних та точність зібраних параметрів
- Система повинна здійснювати періодичний збір даних про серверне обладнання відповідно до заданого розкладу або на основі подій, таких як зміна стану апаратного забезпечення.
- Система повинна мати механізми збереження зібраних даних про серверне обладнання, які дозволять здійснювати аналіз та статистичну оброку інформації в майбутньому

2.Аналіз та сповіщення про аномалії

- Система повинна аналізувати зібрані дані та виявляти аномальні стани або помилки в роботі серверного обладнання
- Система повинна надсилати сповіщення адміністратору або відповідальній особі про виникнення проблем або незвичайних подій.
- Система повинна мати вбудовані алгоритми аналізу даних, що дозволяють виявляти аномалії та незвичайні події на серверному обладнанні

- Система повинна використовувати порогові значення та правила, щоб ідентифікувати аномальні стани і події. Користувач повинен мати можливість налаштувати ці порогові значення та правила відповідно до вимог своєї системи
- Система повинна підтримувати механізми виявлення аномалій, які можуть включати порівняння з нормальними шаблонами, виявлення викидів значень тощо.
- Система повинна підтримувати створення журналів аномальних подій, що дозволяють аналізувати та відстежувати історію виявлення проблем для подальшого вдосконалення моніторингу.

3.Віддаленний доступ та керування

- Система повинна забезпечувати можливість віддаленого доступу до інтерфейсу моніторингу з будь-якого місця та в будь-який час.
- Система повинна забезпечувати можливість віддаленого доступу з будь-якого місця за допомогою захищеного з’єднання, такого як VPN.
- Користувачі повинні мати можливість переглядати стан серверного обладнання, журнали аномальних подій, змінювати налаштування та отримувати сповіщення про аномалії.
- Система повинна підтримувати можливість керування моніторингом, таку як налаштування параметрів моніторингу, додавання або вилучення серверів, активація або деактивація сповіщень тощо.
- Віддалений доступ до системи повинен бути забезпечений з високим рівнем безпеки, включаючи аутентифікацію та шифрування даних, щоб забезпечити конфіденційність та цілісність інформації

2.1.2Нефункціональні вимоги

1.Продуктивність

- Система повинна забезпечувати швидку обробку та аналіз зібраних даних для оперативного виявлення проблем
- Система повинна мати мінімальний вплив на продуктивність серверів та мережі під час збору даних

- Час збору та оновлення інформації про серверне обладнання повинен бути мінімальним, забезпечуючи актуальність даних та миттєву реакцію на зміну стану серверів.
- Система повинна мати оптимізований механізм збереження та управління даними, щоб забезпечити швидкий доступ до історичних даних та ефективну обробку запитів користувачів.

2.Надійність

- Система повинна бути стабільною та надійною для безперебійного моніторингу серверного обладнання.
- Система повинна підтримувати механізми відновлення після відмови, щоб забезпечити неперервність моніторингу
- Система повинна мати механізми виявлення та автоматичного відновлення відмов, включаючи відновлення збоїв з'єднання, відновлення роботи після перезавантаження та інші ситуації непередбачуваної відмови.
- Для забезпечення високої доступності повинні бути передбаченні механізм розподіленого збереження даних, що гарантують збереження та доступ до даних навіть у разі відмови окремих компонентів системи

2.1.3 Вимоги до інтерфейсу

- Система повинна мати зручний та інтуїтивно зрозумілий інтерфейс для відображення стану серверного обладнання
- Інтерфейс повинен підтримувати функції налаштування, перегляду журналів, налаштування сповіщень тощо.

2.2 Архітектура системи моніторингу

Архітектура системи моніторингу визначає структуру, компоненти та взаємозв'язки між ними у системі моніторингу. Вона описує спосіб організації збору, зберігання, обробки та відображення даних моніторингу.

Архітектура системи моніторингу включає такі основні елементи:

- Джерела даних: це можуть бути сервери, мережеві пристрої, датчики або будь-які інші джерела, що надають інформацію про стан системи або ресурсів, які потрібно моніторити.
- Сховище даних: це місце, де зберігаються зібрані дані моніторингу. Це може бути база даних, сховище даних або інший механізм для зберігання та організація даних.
- Обробка даних: цей компонент відповідає за обробку зібраних даних моніторингу. Включає в себе аналіз, агрегацію, фільтрацію, нормалізацію та інші операції, що дозволяють отримати корисну інформацію з зібраних даних
- Механізми сповіщення: ці компоненти відповідають за надсилання сповіщень адміністраторам або відповідним сторонам про виявлені проблеми або події, що вимагають уваги.

2.3 Види архітектур

2.3.1 P2P архітектура

Peer-to-peer, P2P (з англ. — рівний до рівного) — варіант архітектури системи, в основі якої стоїть мережа рівноправних вузлів.[6]

Комп'ютерні мережі типу peer-to-peer (або P2P) засновані на принципі рівноправності учасників і характеризуються тим, що їх елементи можуть зв'язуватися між собою, на відміну від традиційної архітектури, коли лише окрема категорія учасників, яка називається серверами, може надавати певні сервіси іншим.[6]

В чистій мережі «peer-to-peer» не існує поняття клієнтів або серверів, лише рівні вузли, які одночасно функціонують як клієнти та сервери по відношенню до інших вузлів мережі. Ця модель мережевої взаємодії відрізняється від клієнт-серверної архітектури, в якій зв'язок відбувається лише між клієнтами та центральним сервером. Така організація дозволяє зберігати працездатність мережі при будь-якій конфігурації доступних її учасників. Проте практикується використання P2P-мереж, які все ж таки мають сервери, але їх роль полягає вже не у наданні сервісів, а у підтримці інформації з приводу сервісів, що надаються клієнтами мережі.[6]

Основні принципи:

- У піринговій архітектурі кожен вузол (комп'ютер або пристрій) може виконувати як клієнтську, так і серверну роль.
- Всі вузли підключаються один до одного та обмінюються даними без центрального сервера. Вони можуть спільно працювати для збору та обробки даних моніторингу.
- Пірингова архітектура може бути корисною для розподіленого моніторингу, де кожен вузол відповідає за моніторинг певної області чи ресурсу.

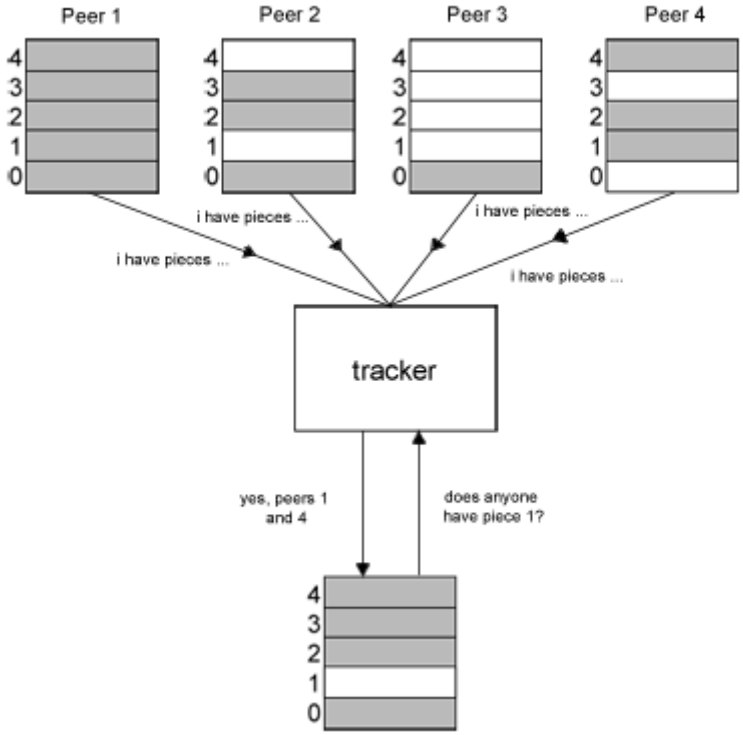

Рис.2.3.1 Схема P2P архітектури

2.3.2 Клієнт-клієнтська архітектура

У клієнт-клієнтській архітектурі, кожен пристрій виступає як самостійний агент, який збирає дані про стан серверів та надсилає їх іншим клієнтам. Клієнтські пристрої можуть виконувати роль агентів збору даних, які активно сканують сервери і отримують інформації про їх стан (наприклад, завантаженість процесора, використання пам'яті, доступність сервісів тощо).

Клієнтські пристрої обмінюються цими даними безпосередньо між собою утворюючи мережу взаємодії. Кожен клієнт може виступати як споживач та постачальник даних одночасно. Наприклад, клієнт може надсилати дані, які він зібрав, іншим клієнтам, які використовують ці дані для аналізу та прийняття рішень.

Основною перевагою клієнт-клієнтської архітектури є пряма і незалежна взаємодія між клієнтами, що дозволяє швидше передавати інформацію та ресурси між ними. Однак дана модель також має свої обмеження, наприклад, щодо безпеки або керування конфліктами між клієнтами.

Загалом, клієнт – клієнтська архітектура використовується для побудови розподілених систем, де клієнти взаємодіють безпосередньо між собою, спільно використовуючи ресурси та послуги.

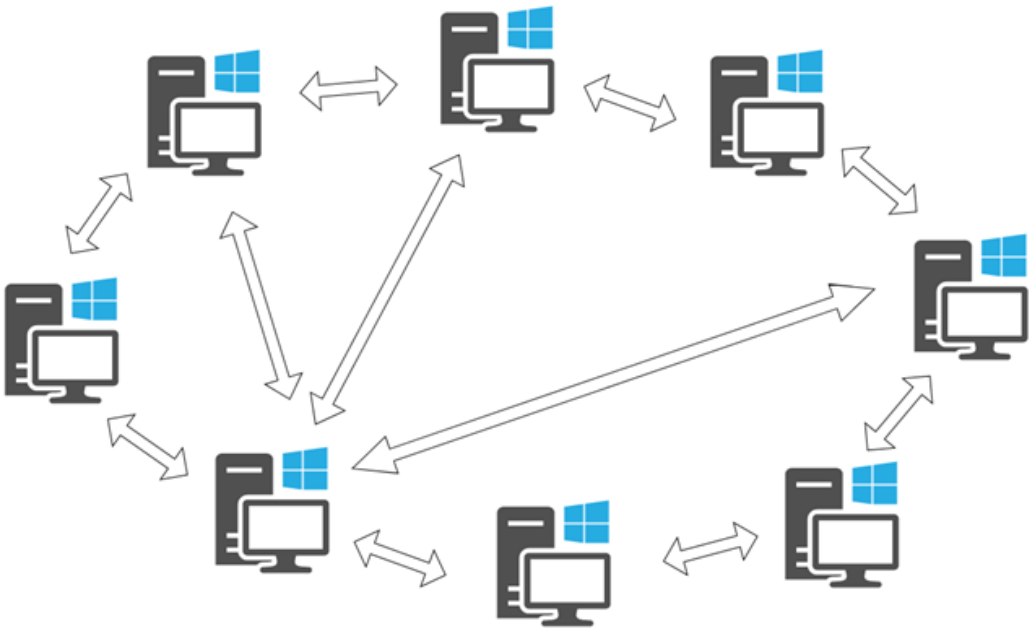

Рис.2.3.2 Схема клієнт клієнтської архітектури

2.3.3 Клієнт-серверна архітектура

Клієнт серверна архітектура використовується для організації взаємодії між клієнтськими пристроями та сервером моніторингу. У даній архітектурі сервер виконує роль централізованої системи, яка збирає, обробляє та аналізує дані про стан серверів, а клієнтські пристрої виступають в ролі джерела цих даних.

Клієнтські пристрої (наприклад, агенти моніторингу або агенти збору даних) встановлюються на кожному сервері, який підлягає моніторингу. Вони відправляють дані про стан сервера (такі як завантаженність процесора, використання пам’яті, доступність сервісів тощо) на сервер моніторингу через мережу

Сервер моніторингу приймає дані від клієнтських пристроїв, зберігає їх у базі даних та проводить аналіз. Він може генерувати сповіщення або повідомлення про будь-які проблеми або ненормальні стани серверів на основі визначених правил і порогів.

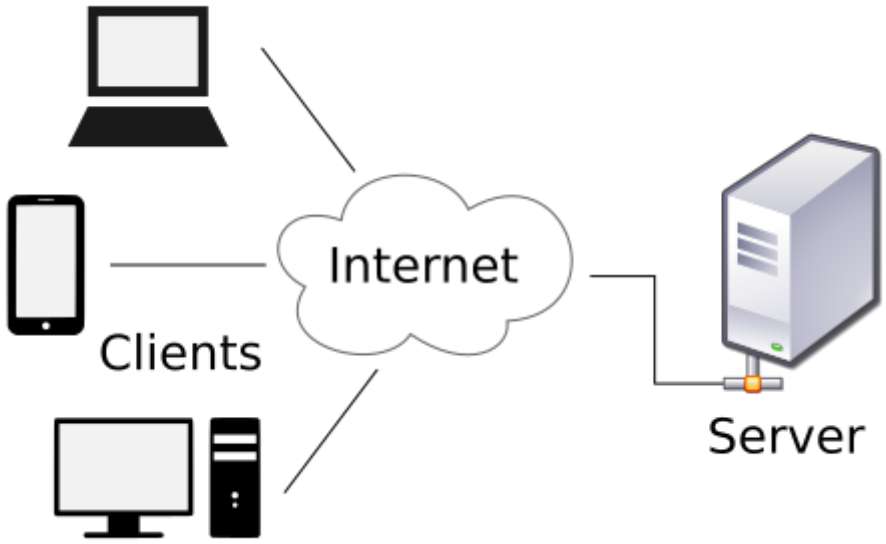

Рис.2.3.3 Схема клієнт-серверної архітектури

Клієнт-серверна архітектура забезпечує централізоване управління і контроль над системами. Сервер надає зручний інтерфейс адміністратору для перегляду даних моніторингу, виконання налаштувань та прийняття рішень щодо діагностики або вирішення проблем серверів.

Дана архітектура також дозволяє масштабувати систему моніторингу, додавати нові клієнтські пристрої або сервери, інтегрувати різні джерела даних та розширювати функціональні можливості моніторингу.

З недоліків слід звернути увагу на те що дана архітектура має обмеження в розподіленості та вимагає високої доступності та продуктивності від серверу моніторингу для забезпечення стабільної роботи системи.

Було прийнято рішення побудувати систему моніторингу на основі клієнт-серверної архітектури.

2.4Вибір архітектури

Було прийнято рішення побудувати систему моніторингу на основі клієнт-серверної архітектури з кількох причин:

1. Централізоване управління: клієнт серверна архітектура дозволяє централізовано керувати моніторингом серверів. Сервер моніторингу є централізованим вузлом, який збирає, обробляє та аналізує дані від клієнтів. Це спрощує управління системою моніторингу та дозволяє налаштовувати її в одному місці.
2. Масштабованість: архітектура дозволяє легко масштабувати систему моніторингу. На центральний сервер можна додавати нові клієнти безпосередньо, без необхідності внесення змін до існуючих клієнтів. Це спрощує процес розширення системи залежно від зростання потреб.
3. Контроль доступу: архітектура надає можливість легко контролювати доступ до системи моніторингу тому, що потрібно контролювати доступ до одного вузла – центрального серверу.
4. Збір та обробка даних: клієнт-серверна архітектура дозволяє ефективно збирати та обробляти дані моніторингу. Сервер здійснює централізовану обробку та аналіз даних, надісланих йому клієнтами. Це дозволяє швидко виявляти аномалії та генерувати сповіщення про проблеми в одному місці.

Було прийнято рішення реалізації клієнт-серверної архітектури без використання агентів на клієнтських пристроях, через простоту реалізації. Схема реалізація виглядатиме наступним чином:

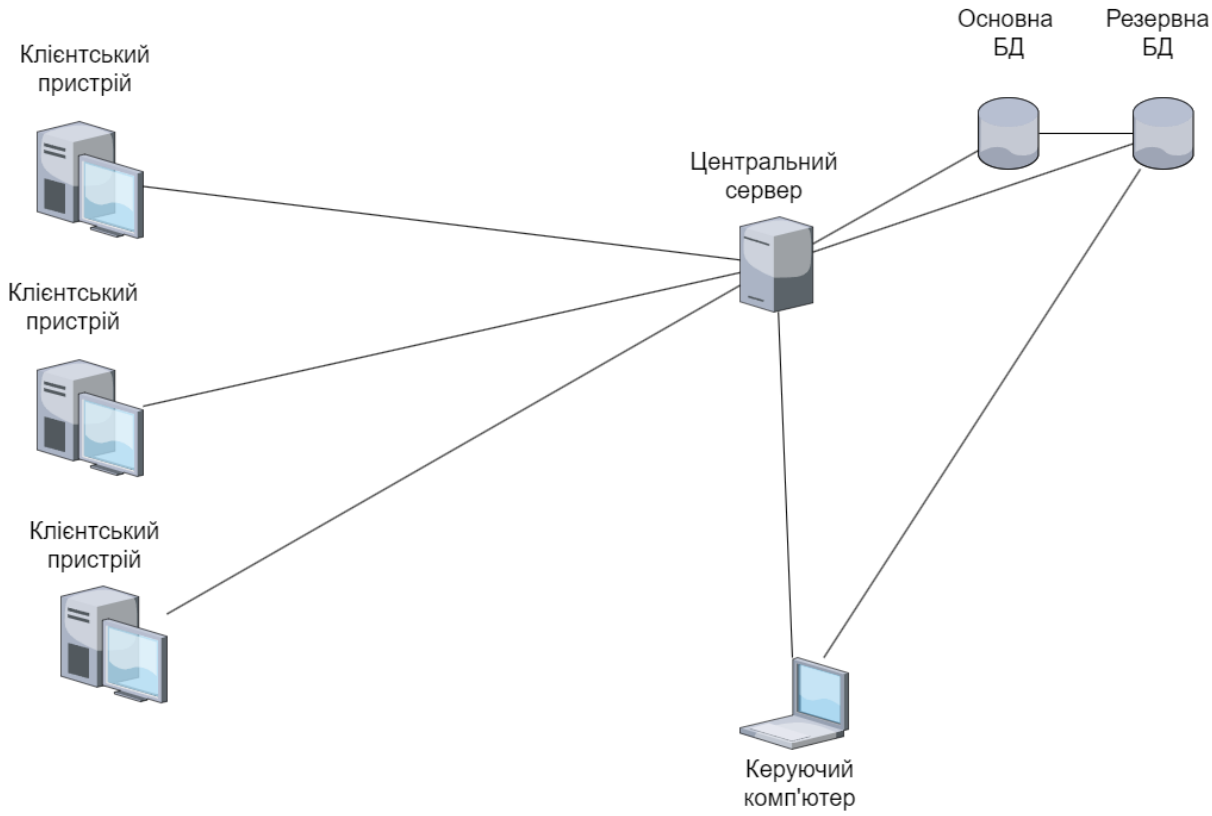

Рис.2.4.1 Схема реалізація системи моніторингу

2.5 Технології для створення системи моніторингу

2.5.1 Python

Python – інтерпретована об’єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Вона підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Вона підтримує кілька парадигм програмування: об’єктно-орієнтовану, процедурну, функціональну та аспектно-орієнтовану.[7]

Вона має такі переваги:

- Простота та зрозумілість: Python має простий синтаксис, що робить його легким для розуміння. Це дозволяє швидко створювати функціональність системи моніторингу і легко змінювати та розширювати її в майбутньому.
- Крос-платформеність: Python є крос-платформною мовою програмування, що означає, що код написаний на ній, може працювати різних операційних

системах. Це дає можливість використовувати систему моніторингу на різних серверних платформах без необхідності значних змін у коді.

- Активна спільнота: вона має велику та активну спільноту розробників, що допомагає знайти відповіді на питання та розв'язати проблеми. Існує багато онлайн ресурсів, документації, форумів та бібліотек, які спрощують розробку системи моніторингу та дозволяють використовувати найкращі практики та рішення розроблені спільнотою.

2.5.2PostgreSQL

PostgreSQL – це об'єктно-реляційна база даних з відкритим вихідним кодом розробка якої триває понад 35 років, що має репутацію надійності і продуктивності.[8]

База даних використовується для зберігання журналів системи моніторингу. Вона дозволяє зберігати повну історію подій та дій в системі моніторингу протягом тривалого періоду часу та дозволяє ефективно шукати записи за певними параметрами.

Перевагами використання саме PostgreSQL є:

- Надійність та цілісність даних:PostgreSQL відома своєю надійністю та стабільністю. Вона включає механізми транзакційності, резервного копіювання та відновлення, що дозволяють забезпечити цілісність даних та запобігти втраті інформації
- Масштабованість: вона має хороші можливості масштабування. Вона підтримує паралельне виконання запитів, розподілені запити та реплікацію для забезпечння високої продуктивності та доступності системи навіть при великому обсязі даних та високому навантаженні
- Розширена підтримка мов програмування: підтримує різні мови програмування, включаючи Python, Java, C++,Ruby та іншу.Це надає можливівсть використовувати мови програмування, з якими воний знайомі та ефективно володіють.

2.5.3 API та бібліотеки

Telebot – є бібліотекою для роботи з Telegram API (application programming interface) в середовищі Python.Вона надає змогу легко створювати та керувати Telegram-ботами. Завдяки telebot можна створювати ботів, які взаємодіють з користувачами через чати, приймають команди, обробляють повідомлення та виконують певні дії.

Переваги створення системи моніторингу із використанням telebot:

- Зручний інтерфейс комунікації:Telegram має широко поширену та зручну платформу для обміну повідомленнями. Використання telebot дозволяє взаємодіяти з користувачами в знайомому та зручному середовищі
- Миттєві сповіщення: Telegram надає можливість надсилати миттєві сповіщення на пристрої користувачів через повідомлення чату або сповіщення. Це дозволяє оперативно повідомляти про виникнення проблем та надсилати важливі сповіщення про стан серверів.
- Гнучкість налаштувань:telebot надає широкий набір налаштувань для керування ботом, включаючи обробку команд, фільтрацію повідомлень, реакцію на події та інше.

Paramiko -бібліотека для роботи з протоколом SSH в середовищі Python.Вона надає можливість здійснювати безпечне з’єднання з віддаленими серверами по протоколу SSH, виконувати команди на цих серверах і отримувати результат виконання.

Переваги використання Paramiko:

- З'єднання по протоколу SSH: SSH є широко використовуваним протоколом для безпечного з'єднання з віддаленими серверами. Використання Paramiko дозволяє встановити з'єднання з серверами по протоколу SSH, що є надійним та захищеним способом отримання інформації про стан серверів.
- Виконання команд на віддалених серверах: Paramiko надає можливість виконувати команди на віддалених серверах через SSH-з'єднання. Це дозволяє отримувати інформацію про різні параметри серверів, такі як навантаження процесора, використання пам'яті, доступність служб та інші, для подальшого аналізу та моніторингу.

- Обмін даними з серверами: Paramiko дозволяє обмінюватись даними з віддаленими серверами через SSH-з'єднання. Ви можете передавати файли, отримувати журнали, налаштування та інші дані з серверів для подальшого аналізу або зберігання.
- Гнучкість та налаштування: Paramiko надає багато функцій та налаштувань для керування SSH-з'єднаннями та виконання команд. Вона надає можливість налаштувати аутентифікацію, шифрування, ключі SSH та інші параметри для забезпечення безпеки та ефективності взаємодії з серверами.

2.6Висновки до 2 розділу

У даному розділі було проведено проєктування системи моніторингу з метою вирішення основних проблем, що виникають у контексті досліджуваної області. Аналізуючи ці проблеми, було визначено, що необхідно забезпечити моніторинг різних параметрів та стану системи з метою забезпечення її безперебійної та ефективної робот.

На основі проведеного аналізу було визначено функціональні вимоги до системи моніторингу. Ці вимоги включали збір даних, їх обробку, візуалізацію та забезпечення сповіщень у разі виявлення проблем. Крім того, були встановлені нефункціональні вимоги, такі як надійність, швидкодія, масштабованість.

Для реалізації системи моніторингу було обрано відповідні технології та інструменти. Було розглянуто можливі архітектурні варіанти та обрано найбільш підходящий для даної задачі. Також було розроблено схему взаємодії компонентів системи та визначено основні модулі.

У результаті проєктування було забезпечено можливість масштабування системи з метою пристосування до зростання обсягу даних та навантаження. Це дозволить забезпечити ефективну роботу системи навіть у разі збільшення обсягу оброблюваних даних. Крім того, були розроблені механізми для забезпечення стабільності та надійності системи, що дозволить запобігти виникненню помилок та збоїв.

Процес проєктування також передбачав врахування можливості інтеграції з існуючими системами та пристроями, що вже використовуються в досліджуваній області. Це забезпечить сумісність та взаємодію з існуючою інфраструктурою.

У результаті проєктування системи моніторингу було досягнуто встановлених цілей і вимог. Було створено детальну архітектуру системи, розроблено необхідні модулі та механізми. Очікується, що система моніторингу забезпечить ефективний контроль та управління параметрами та станом системи, що дозволить підвищити її продуктивність та надійність.

Наступним кроком у дослідженні буде реалізація розробленої системи моніторингу та її тестування на реальних даних. Це дозволить оцінити її функціональність та перевірити, наскільки вона задовольняє поставлені вимоги. Також можливі дальші покращення та розширення системи з метою її оптимізації та розвитку.

РОЗДІЛ 3
РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ СТАНУ СЕРВЕРНОГО ОБЛАДНАННЯ

3.1Розробка БД

Для розробки бази даних для зберігання даних системи моніторингу технічного стану серверного обладнання, використали послідовність кроків з використанням PostgeSQL, Python, бібліотеки psycopg2 для з'єднання з базою даних. Нижче описано процес розробки БД:

1. Створюємо БД на хмарному сервері

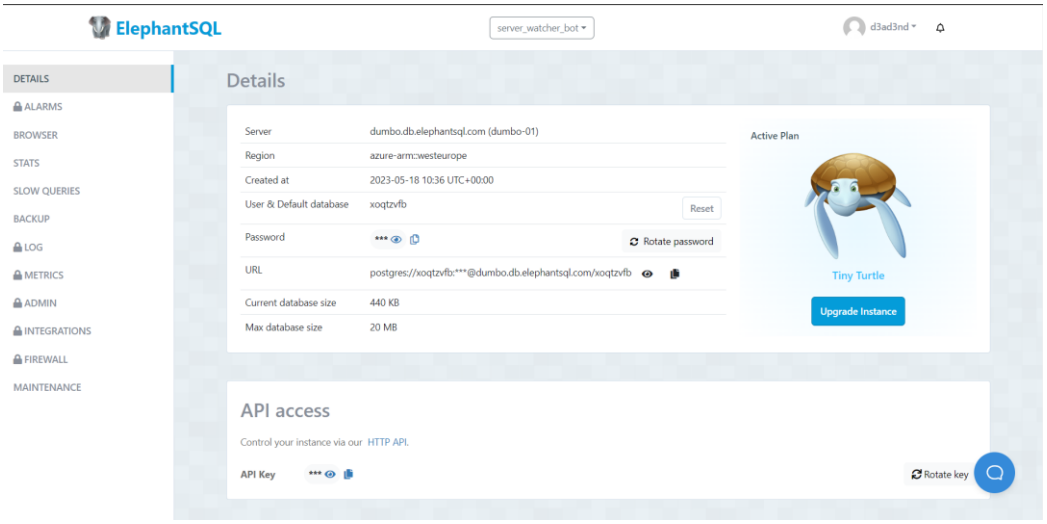

Рис3.1Вікно панелі адміністратора БД

2. Встановлюємо *psycopg2*

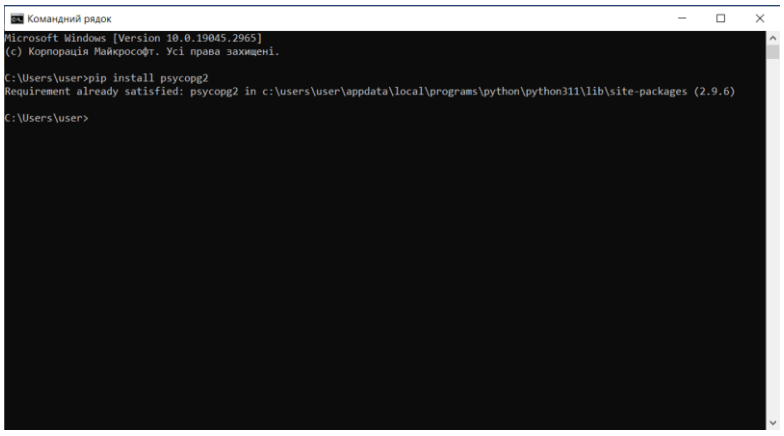

Рис 3.2 Вікно встановлення *psycopg2*

3. Підключаємося до БД

Для підключення до БД необхідно розбити посилання на БД на компоненти зрозумілі для *psycopg2*. Для цього створюємо таку функцію та передаємо їх посилання на БД, функція повертає курсор для роботи з БД.

```python
def conect_to_BD():
    connection_url = urlparse(db_url)
    username = connection_url.username
    password = connection_url.password
    database = connection_url.path[1:]
    hostname = connection_url.hostname
    port = connection_url.port
    connection = psycopg2.connect(
        database = database,
        user = username,
        password = password,
        host = hostname,
        port = port
    )
    db_cursor = connection.cursor()
    return db_cursor,connection
```

Рис3.3 Функція підключення до БД

4. Створюємо таблиці для зберігання даних системи моніторингу

Таблиці створюємо з умовою *CREATE IF NOT EXIST*, це надає змогу перевіряти цілісність таблиць у випадку перезапуску системи, а також полегшує розгортання системи моніторингу.

```python

def create_tables():
    # Створення таблиці servers
    DB_CURSOR, CONNECTION = conect_to_BD()
    DB_CURSOR.execute('''
        CREATE TABLE IF NOT EXISTS servers (
            id SERIAL PRIMARY KEY,
            ip VARCHAR(15) NOT NULL,
            ssh_login VARCHAR(50),
            ssh_password VARCHAR(255))
    ''')

    # Створення таблиці availability
    DB_CURSOR.execute('''
        CREATE TABLE IF NOT EXISTS availability (
            id SERIAL PRIMARY KEY,
            server_id INTEGER REFERENCES servers(id) ON DELETE CASCADE,
            is_available BOOLEAN,
            timestamp TIMESTAMP DEFAULT NOW()
        )
    ''')

    DB_CURSOR.execute('''
        CREATE TABLE IF NOT EXISTS disk_usages (
            id SERIAL PRIMARY KEY,
            server_id INTEGER REFERENCES servers(id) ON DELETE CASCADE,
            ram_usage INTEGER,
            timestamp TIMESTAMP DEFAULT NOW()
        )
    ''')

    DB_CURSOR.execute('''
        CREATE TABLE IF NOT EXISTS ram_usages (
            id SERIAL PRIMARY KEY,
            server_id INTEGER REFERENCES servers(id) ON DELETE CASCADE,
            ram_usage INTEGER,
            timestamp TIMESTAMP DEFAULT NOW()
        )
    ''')

    CONNECTION.commit()
    DB_CURSOR.close()
    CONNECTION.close()
```

Рис3.4 Функція створення таблиць в БД

Після вищенаведених маніпуляцій маємо таку структуру таблиць в БД:

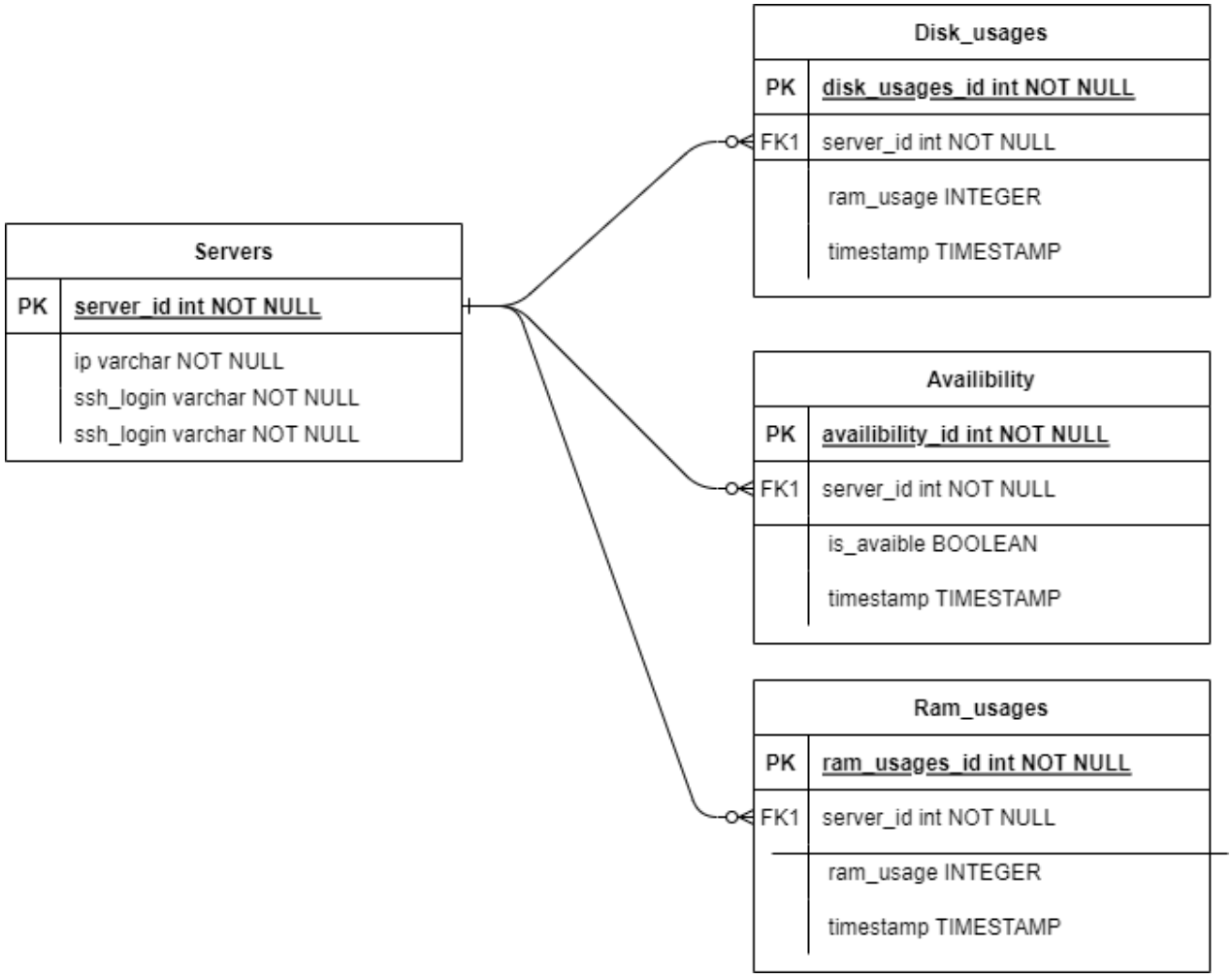

Рис.3.5 Діаграма залежностей таблиць в БД

3.2 Створення телеграм бота

Для створення бота нам необхідно звернутися до BotFather.Через нього можна зробити базові налаштування бота, обрати назву та отримати API-ключ, який ідентифікуватиме бота серед інших. Після отримання API- ключа переходимо до створення об’єкту бота, що дасть змогу працювати з ботом за допомогою *Python* коду. Також одразу створюємо стартову функцію, яка виводить привітання та пояснення щодо подальших дій користувача.

```
@bot.message_handler(commands=['start'])
def start(message):
    bot.send_message(message.chat.id, "Hello, I'm bot for monitoring servers status.Type /help to see all commands")
```

Рис.3.6 Стартова функція телеграм боту

Також одразу створюємо команду *help* після виконання якої користувач матиме змогу ознайомитись з усіма існуючими функціями:

```
@bot.message_handler(commands=['help'])
def help(message):
    bot.send_message(message.chat.id, '''List of commands:
    /help - help.
    /start_monitoring - start automatic server monitoring.
    /stop_monitoring - stop automatic server monitoring.
    /add_server_ip - add server to automatic monitoring.You need to stop automatic monitoring before adding a server.
    /add_server_ssh - add remote server ssh data for login and monitoring
    /remove_server - remove server from automatic monitoring.You need to stop automatic monitoring before remove a server
    /server_list - displays a list of monitored server
    /ram_list - displays ram_usage log
    /availability_list - displays servers availability log
    ''')
```

Рис 3.7 Функція *help*

3.3Створення функцій роботи з БД

Для роботи системи моніторингу з БД нам необхідно описати функції для запису журнальних даних в БД, отримання їх у структурі даних, зрозумілій для мови програмування, формі, та для оновлення уже існуючих даних, для кожної таблиці потрібно зробити окрему функцію, через доволі велику різницю між ними. Нижче наведено приклад для певних таблиць:

```
def add_server(ip_address):
    DB_CURSOR, CONNECTION = conect_to_BD()
    DB_CURSOR.execute('''
    INSERT INTO servers(ip) VALUES (%s)
    ''',(ip_address,))
    CONNECTION.commit()
    DB_CURSOR.close()
    CONNECTION.close()

def remove_server(ip_address):
    DB_CURSOR, CONNECTION = conect_to_BD()
    DB_CURSOR.execute('''
    DELETE FROM servers WHERE ip = %s
    ''',(ip_address,))
    CONNECTION.commit()
    DB_CURSOR.close()
    CONNECTION.close()
```

Рис.3.8 Функції додавання та видалення серверів з БД

```python
def update_available_status(is_available,server_id):
    db_cursor,connection = conect_to_BD()
    db_cursor.execute('INSERT INTO availability(is_available,server_id)  VALUES(%s,%s)', (is_available, server_id))
    connection.commit()
    db_cursor.close()
    connection.close()
```

Рис.3.9 Функція оновлення даних

```python
def get_ram_usages():
    db_cursor,connection = conect_to_BD()
    db_cursor.execute('''SELECT servers.ip, ram_usages.*
    FROM ram_usages
    JOIN servers
    ON ram_usages.server_id = servers.id LIMIT 100''')
    ram_usages = db_cursor.fetchall()
    db_cursor.close()
    connection.close()
    return[{'ip':s[0], 'id':s[1], 'server_id':s[2],'ram_usage':s[3],'timestamp':s[4]} for s in ram_usages]
```

Рис3.10 Функція отримання даних з БД

Функції наведенні вище виконують запити до бази даних, отримують відповідь у вигляді *SQL* даних та повертають значення з якими можна працювати в *Python*.

3.3 Створення функцій моніторингу

Для реалізації основної задачі системи моніторингу технічного стану серверного обладнання реалізовано функції моніторингу доступності, зайнятості оперативної пам’яті та інші. Це реалізовано за допомогою протоколів мережі *ICMP* та *SSH*. *ICMP* потрібен для функції перевірки доступності серверів.*SSH* використаний для під’єднання до клієнтів та зняття показників, та перевірки відповіді на *SSH* запити.

```python
def connect_with_ssh(ip_address,login,password):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    try:
        ssh.connect(hostname=f'{ip_address}', username=f'{login}', password=f'{password}')
        return ssh, True, "Ok"
    except paramiko.AuthenticationException:
        return None,False, f"Server {ip_address} Authentication error: " + str(paramiko.AuthenticationException)
    except paramiko.SSHException:
        return None,False, f"Server {ip_address} SSH error: " + str(paramiko.SSHException)
    except Exception:
        return None,False, f"Server {ip_address} Error: " + str(Exception)
```

Рис.3.11 Функція підключення до серверів через *SSH*

Функція *connect_with_ssh(ip_address, login, password)* виконує наступні основні задачі:

1. Створення SSH-з'єднання: За допомогою *paramiko.SSHClient()* створюється об'єкт SSH-клієнта для встановлення з'єднання з віддаленим сервером за допомогою протоколу SSH.
2. Налаштування політики додавання нових хостів: Встановлюється політика автоматичного додавання нових хостів до списку довірених хостів за допомогою *ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy()).*Це дозволяє уникнути підтвердження ручного додавання хоста під час першого з'єднання.
3. Встановлення SSH-з'єднання з сервером: Викликається *ssh.connect()* з параметрами *hostname* (IP-адреса сервера), *username* (логін для авторизації) та *password* (пароль для авторизації). Якщо з'єднання успішно встановлено, повертається об'єкт SSH-клієнта ssh, прапорець True та повідомлення "Ok".
4. Обробка винятків автентифікації: Якщо виникає помилка автентифікації (*AuthenticationException*), повертається *None*, прапорець *False* та повідомлення про помилку автентифікації.
5. Обробка винятків SSH: Якщо виникає помилка *SSH (SSHException),* повертається *None*, прапорець *False* та повідомлення про помилку *SSH*.
6. Обробка інших винятків: Якщо виникає будь-яка інша помилка, повертається *None*, прапорець *False* та повідомлення про помилку.

Функція дозволяє встановлювати SSH-з'єднання з віддаленим сервером за допомогою вказаної IP-адреси, логіну та пароля. При успішному з'єднанні повертається об'єкт SSH-клієнта, що дозволяє виконувати команди на сервері. У випадку помилки автентифікації або SSH, або інших винятків, повертаються відповідні значення для подальшої обробки в коді.

Всі функції для моніторингу окремих показників підключаються до головної, що надає змогу керувати ними в одному місці в одному місці та додавати нові функції для моніторингу та зняття показників з серверного обладнання.

```python
def monitor_servers_params(message):
    while monitoring_enabled:
        monitor_server_list = db_functions.get_servers()
        response_time = 0
        for server in monitor_server_list:
            if response_time is not None:
                db_functions.update_available_status(True, server['id'])
            else:
                db_functions.update_available_status(False, server['id'])
                bot.send_message(message.chat.id,f"Server {server['ip']} is down")

        memory_percent = ram_usage_monitoring(monitor_server_list)
        if (memory_percent != None) and (memory_percent > ram_threshold):
            db_functions.update_ram_status(memory_percent,server['id'])
            bot.send_message(message.chat.id,f"Server {server['ip']} RAM CRITICAL {memory_percent}%")

        time.sleep(cooldown)
```

Рис 3.12 Головна моніторингова функція

Функція monitor_servers_params(message) виконує наступні основні задачі:

1. Отримання списку серверів для моніторингу: За допомогою функції db_functions.get_servers() отримується список серверів, які підлягають моніторингу.
2. Перевірка доступності серверів: Для кожного сервера з отриманого списку виконується перевірка доступності шляхом виконання певного тесту (наприклад, пінгування або виконання команди SSH). Якщо сервер відповідає, викликається функція db_functions.update_available_status(True, server['id']), яка оновлює статус доступності сервера в базі даних. У противному випадку, якщо сервер не відповідає, викликається функція db_functions.update_available_status(False, server['id']), а також надсилається повідомлення користувачу про недоступність серверу за допомогою bot.send_message().
3. Моніторинг використання пам’яті: Функція ram_usage_monitoring виконує моніторинг використання оперативної пам’яті на серверах зі списку monitor_server_list.Повертає відсоток використання пам’яті. Якщо цей відсоток не є нульовим і не перевищує заданий поріг викликається функція db_functions.update_ram_status, яка оновлює статус використання пам’яті в базі даних.Також надсилається повідомлення користувачу про критичне використання пам’яті на сервері за допомогою bot.send_message

4. Моніторинг дискового простору: Шляхом виклику певних функцій отримується інформація про використання дискового простору на кожному сервері зі списку monitor_server_list. Якщо використання дискового простору перевищує заданий поріг, викликається відповідна функція для оновлення статусу використання дискового простору в базі даних та надсилається повідомлення користувачу через bot.send_message()

3.4Тестування системи

Для перевірки системи моніторингу технічного стану серверного обладнання створимо віртуальну машину, яка знаходяться в тій самій мережі що і хост на якому розміщена система моніторингу. Налаштуємо DHCP так щоб у хоста та віртуальної машини був статичний IP-адрес. Додаємо віртуальну машину до списку серверів для моніторингу та поки не запускаємо її для перевірки сповіщень про доступність

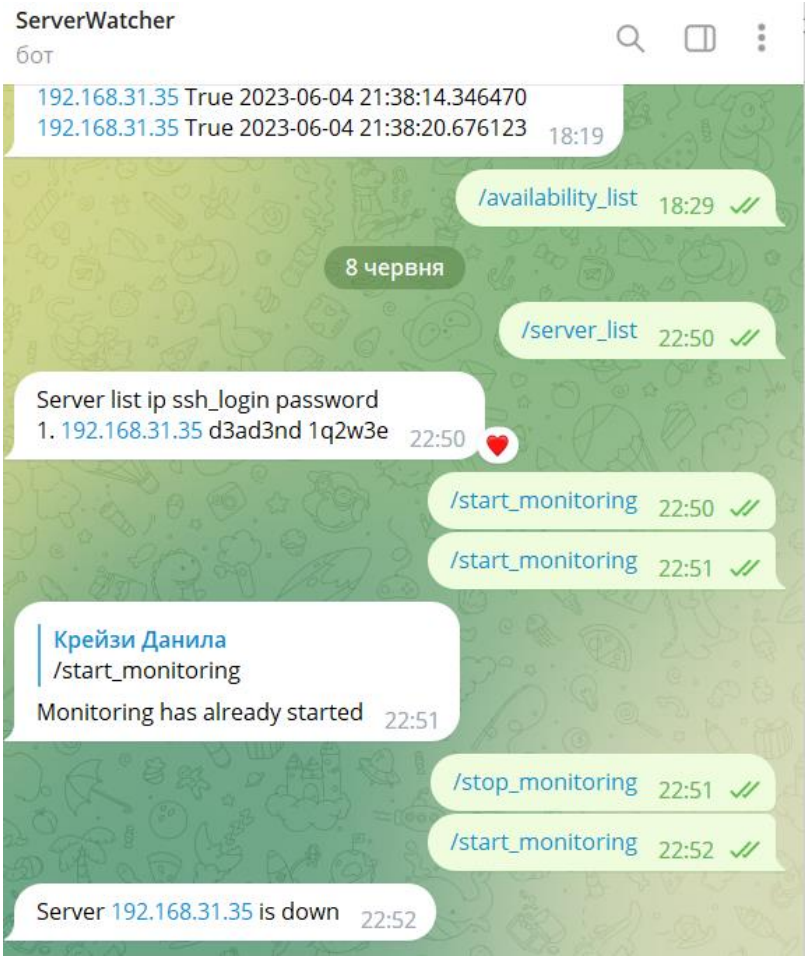

Рис.3.13 Екран успішного повідомлення про недоступність сервера

Тепер встановимо пороговий показник використання оперативної пам’яті на мінімальне значення та перевіримо повідомлення про перевищення використання оперативної пам’яті

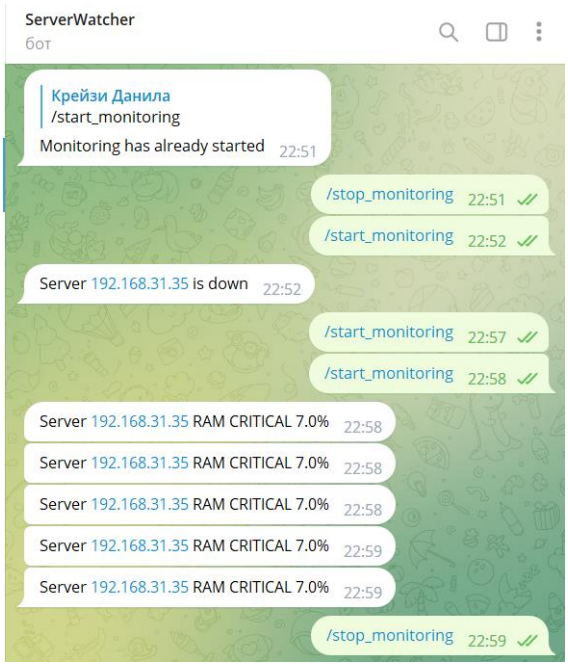

Рис 3.14 Екран успішного повідомлення про перевищення *RAM* сервера

Тепер перевіримо вивід журналів які зберігаються в БД, для цього виконаємо кілька команд що виводять журнали користувачу.

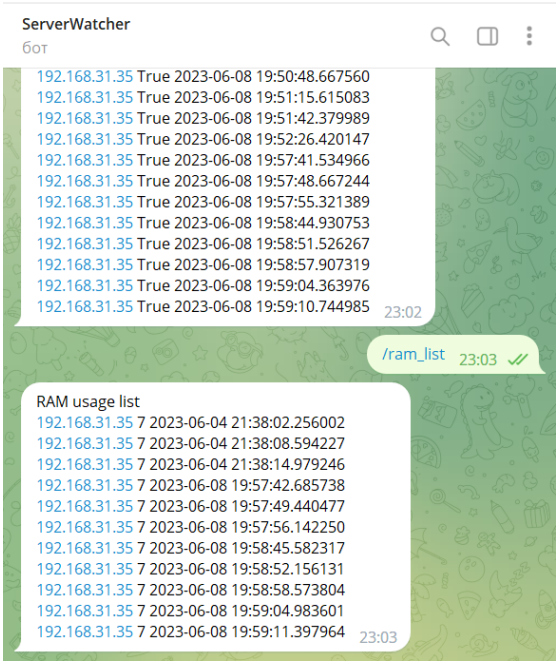

Рис.3.15 Екран з успішним виводом журналів

3.5 Висновки до Зрозділу

У даному розділі було проведено розробку системи моніторингу з метою вирішення проблем, пов'язаних з контролем технічного стану серверного обладнання. Під час аналізу цих проблем, було визначено, що необхідно забезпечити нагляд та оцінку різних параметрів технічного стану, таких як температура, навантаження, робота різних компонентів тощо.

На основі проведеного аналізу було сформульовано функціональні вимоги до системи моніторингу. Ці вимоги включали збір та обробку даних з серверного обладнання, їх візуалізацію у зручному форматі та можливість сповіщення про виявлені аномалії або проблеми.

Під час процесу розробки було обрано відповідні технології та інструменти для реалізації системи моніторингу. Було розроблено алгоритми та модулі для збору даних з серверного обладнання, їх обробки та візуалізації. Додатково, було розроблено механізми сповіщення, які дозволяють оперативно реагувати на виявлені проблеми або небажані становища.

В процесі розробки системи моніторингу технічного стану серверного обладнання було звернуто увагу на додаткові аспекти, що сприятимуть її ефективності та надійності.

Одним з таких аспектів є розробка механізмів логування та зберігання історичних даних. Це дозволить зберігати та аналізувати дані про стан серверного обладнання протягом тривалого періоду, що є корисним для виявлення трендів, ідентифікації довгострокових проблем та планування обслуговування.

У результаті розробки системи моніторингу було досягнуто основні цілі - забезпечено постійний контроль технічного стану серверного обладнання та можливість оперативного реагування на виявлені проблеми. Розроблена система дозволяє ефективно виявляти аномалії та забезпечувати безперебійну роботу серверів.

ВИСНОВКИ

У рамках дипломної роботи з назвою "Розробка та дослідження системи моніторингу технічного стану серверного обладнання" було проведено ретельне дослідження та розробку системи, спрямованої на контроль та оцінку технічного стану серверного обладнання. В результаті проведених робіт були отримані наступні висновки:

Виявлені та аналізовані основні проблеми, пов'язані з контролем технічного стану серверного обладнання, такі як перегрів, навантаження, відмови компонентів тощо. Це дозволило усвідомити необхідність розробки системи моніторингу, спрямованої на попередження та виявлення подібних проблем з метою запобігання негативним наслідкам.

Було сформульовано функціональні вимоги до системи моніторингу, включаючи збір даних, їх обробку, візуалізацію та сповіщення про виявлені проблеми. Ці вимоги послужили основою для подальшої розробки системи та визначення її компонентів та функціональних можливостей.

Було розроблено та реалізовано систему моніторингу, що забезпечує постійний контроль технічного стану серверного обладнання. Система включає модулі збору даних, їх обробки та візуалізації, а також механізми сповіщення про виявлені аномалії. Застосовані технології та інструменти були обрані з метою забезпечення ефективності та масштабованості системи. Додатково, в процесі розробки та дослідження системи моніторингу технічного стану серверного обладнання було виявлено кілька потенційних напрямків подальшого розвитку

По-перше, можливе розширення системи для включення моніторингу мережевих параметрів. Це дозволить здійснювати контроль як серверного обладнання, так і мережевого зв'язку, що може бути корисним при виявленні проблем, пов'язаних зі з'єднанням та пропускною здатністю.

По-друге, можливе розширення системи для включення аналізу журналів подій. Це дозволить здійснювати моніторинг системних подій та реагувати на потенційно небезпечні або некоректні ситуації. Інтеграція з існуючими системами журналування може допомогти у забезпеченні повної картини стану серверного обладнання.

По-третє, можливе розширення системи для включення прогнозування несправностей. Використання аналітики даних та машинного навчання може допомогти виявити паттерни та ознаки, які передували відмовам чи проблемам з серверним обладнанням. Це дозволить приймати запобіжні заходи та забезпечити безперебійну роботу інфраструктури.

Проведено дослідження та експерименти для перевірки функціональності та ефективності розробленої системи. Результати підтвердили, що система здатна ефективно виявляти аномалії та проблеми в роботі серверного обладнання, а також оперативно сповіщати про них. Дослідження також виявили можливість розширення та покращення системи для подальшої оптимізації та пристосування до змінних вимог.

Загалом, дипломна робота засвідчує успішну розробку та дослідження системи моніторингу технічного стану серверного обладнання. Результати цієї роботи можуть бути використані для покращення ефективності, надійності та безпеки роботи серверів, що має велике значення для підтримки нормального функціонування інформаційних систем та бізнес-процесів.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://books.google.com.ua/books?id=KirJSqFcWIEC&pg=PA1&hl=uk&source=gbs_toc_r&cad=4#v=onepage&q&f=false
2. Art of Monitoring James Turnbull
3. "Monitoring with Prometheus: Effective Monitoring and Alerting Practices for DevOps" Brian Brazil
4. Monitoring with Graphite: Tracking Dynamic Host and Application Metrics at Scale" Jason Dixon
5. P2P архітектура [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Peer-to-peer (Дата звернення 27.05.2023) - Назва з екрану
6. Python [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Python (Дата звернення 25.05.2023) - Назва з екрану
7. PostgreSQL [Електронний ресурс] - Режим доступу до ресурсу https://www.postgresql.org/ (Дата звернення 28.05.2023) – Назва з екрану
8. Nagios [Електронний ресурс] – Режим доступу до ресурсу: https://www.nagios.com/solutions/server-monitoring/ (Дата звернення 24.05.2023) – Назва з екрану
9. Nagios [Електронний ресурс] – Режим доступу до ресурсу: https://www.nagios.com/ (Дата звернення 23.05.2023) – Назва з екрану
10. Prometheus [Електронний ресурс] – Режим доступу до ресурсу: https://prometheus.io/ (Дата звернення 23.05.2023) – Назва з екрану