Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних IT-комплексів”

Виконав здобувач вищої освіти
II курсу, групи КІ-25М
ОПП «Комп’ютерна інженерія»
спеціальності F7 «Комп’ютерна інженерія»
_____ Удудяк Б.О.
«____» _____ 2026 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«____» _____ 2026 р.

Нормоконтроль проекту
_____ Коваленко А.С.
«____» _____ 2026 р.

Рецензент _____
_____

Підписано КЕП
Центральноукраїнський національний технічний університет
Удудяк Богдан Олексійович
Сертифікат: 514B5C86A1E5DA11040000005458A2004B14B005
Дійсний з 09.05.2026 15:36:48 по 09.05.2027 15:36:47

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _F_____*"Інформаційні технології"*_____

Спеціальність F7 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 1 » вересня 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Удудяку Богдану Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних IT-комплексів*

2. Керівник роботи *Смірнов Сергій Анатолійович, кандидат технічних наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 41-13 від 17.08.2026 року

3. Строк подання студентом роботи до захисту *2.12.2026 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних IT-комплексів*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*5. Впровадження системи в промислову експлуатацію*
2. Перегляд аналогічних існуючих систем.	*6. Наукова новизна.*
3. Опис і обґрунтування проектних рішень.	*7. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*8. Висновки.*

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Марченко К.М., к.т.н., доцент	06.10.2026	16.11.2026

8. Дата видачі завдання « 1 » вересня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2026 р.	
3.	Розробка моделі компонента	20.10.2026 р.	
4.	Розробка структур даних	25.10.2026 р.	
5.	Розробка алгоритмів системи	30.10.2026 р.	
6.	Програмування алгоритмів	10.11.2026 р.	
7.	Розрахунки з охорони праці та техніки безпеки	15.11.2026 р.	
8.	Оформлення ПЗ	17.11.2026 р.	
9.	Попередній захист роботи	2.12.2026 р.	

Дата видачі завдання
« 1 » вересня 2026 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 1 » вересня 2026 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Удудяк Б.О. Дослідження та програмна реалізація системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів. F7 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

Метою розробки є дослідження та програмна реалізація системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

Об'єктом дослідження є процес спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

Предметом дослідження є методи спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

Методи дослідження базуються на методах операційних систем, методах теорії комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, моніторинг, ІТ-комплекс

ABSTRACT

Ududiak B.O. Research and Software Implementation of an Observability and Health Monitoring System for Hybrid Cloud-Enterprise IT Infrastructures. F7 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

This master's-level final qualification thesis presents the development of software designed for an observability and health monitoring system for hybrid cloud-enterprise IT infrastructures.

The objective of this project is the research and software implementation of an observability and health monitoring system for hybrid cloud-enterprise IT infrastructures.

The object of the research is the process of observability and health monitoring for hybrid cloud-enterprise IT infrastructures.

The subject of the research consists of methods for the observability and health monitoring of hybrid cloud-enterprise IT infrastructures.

The research methodology draws upon principles from operating systems, computer network theory, mathematical statistics, and software development.

The outcome of the work is a software implementation of an observability and health monitoring system for hybrid cloud-enterprise IT infrastructures.

During the development of the software model, an analysis of existing hardware and software tools was conducted. All components of the developed software are described in detail.

A user-friendly interface has been developed, and instructions for operating the software tools are provided. The program can be used on PCs running Windows 10/11.

The program was developed in a Python environment.

Keywords: computer engineering, monitoring, IT system

ВІДОМІСТЬ ПРОЕКТУ

		Текстові документи			
1	А4	Пояснювальна записка	87	–	
2	А4	Робоча програма	67	–	Додаток А
		Графічні документи			
3	А4	Наукова новизна	1	–	
4	А4	Структурна схема	1	–	
5	А4	Функціональна схема роботи системи	1	–	
6	А4	Діаграма взаємодії процесів	1	–	
7	А4	Блок-схема основної програми	1	–	
8	А4	Блок-схема роботи підпрограми	1	–	

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 9
ВСТУП..... 10
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 12
1.1 Призначення системи..... 12
1.2 Область застосування 12
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 15
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 15
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 22
2.3 Розгорнута постановка завдання 23
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 25
3.1 Опис функціонування системи 25
3.2 Розробка структурної схеми..... 31
3.3 Розробка функціональної схеми 45
3.4 Розробка діаграми процесів 48
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 49
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 49
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 61
6 НАУКОВА НОВИЗНА 66

7 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 67
7.1 Вступ 67
7.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста ... 67
7.3 Розробка заходів з поліпшення стану охорони праці 71
7.4 Техніка безпеки та протипожежна профілактика 72
7.5 Розрахункова частина. Розрахунок штучного освітлення 73
8 ОСНОВНІ ВИСНОВКИ 77
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 79

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

DMA – (Direct memory Access) – прямий доступ до пам‘яті;

ISA – 16 бітна шина персонального комп’ютера;

PC – (Personal Computer) – персональний комп’ютер;

PCI – 32 бітна шина персонального комп’ютера;

Socket – порт взаємодії з мережею;

ВІС – велика інтегральна схема;

ЕОМ – електронна обчислювальна машина;

ЗП – зовнішній пристрій;

ЗЗП – зовнішній запам‘ятовуючий пристрій;

ІС – інформаційна система;

ІТ – інформаційні технології;

КПДП – контролер прямого доступу до пам‘яті;

КС – комп’ютерна система;

МП – мікропроцесор;

НГМД – накопичувач на гнучких магнітних дисках;

НМД – накопичувач на магнітних дисках;

ОЗП – оперативний запам‘ятовуючий пристрій;

ОС – операційна система;

ПДП – прямий доступ до пам‘яті;

ПЕОМ – персональна електронно-обчислювальна машина;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

СЦМК – сервіс централізованого моніторингу та контролю.

ВСТУП

Актуальність теми. Інструменти моніторингу інфраструктури розроблені для контролю стану, продуктивності та доступності компонентів IT-інфраструктури. Це може включати будь-що: від маршрутизаторів і комутаторів до серверів, програм і баз даних.

Ці інструменти збирають та аналізують різні показники для раннього виявлення потенційних проблем, що дозволяє IT-персоналу реагувати проактивно, перш ніж проблема виникне. Деякі інструменти моніторингу інфраструктури також виконують аналіз першопричин, що дозволяє IT-персоналу швидко визначати джерело проблеми та вживати коригувальних заходів, коли вона виникає.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних IT-комплексів.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем спостережуваності та моніторингу стану гібридних хмарно-корпоративних IT-комплексів.

– Дослідження системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних IT-комплексів.

– Програмна реалізація системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних IT-комплексів.

Об’єктом дослідження є процес спостережуваності та моніторингу стану гібридних хмарно-корпоративних IT-комплексів.

Предметом дослідження є методи спостережуваності та моніторингу стану гібридних хмарно-корпоративних IT-комплексів.

Методи дослідження базуються на методах операційних систем, методах теорії комп’ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

– Розроблено вітчизняний продукт спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVIII науково-технічній конференції здобувачів вищої освіти «Наука-виробництву», основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №17.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Рішення для моніторингу інфраструктури – це центральна нервова система ваших IT-операцій. Це платформа, призначена для спостереження, відстеження та аналізу стану та продуктивності всього вашого технологічного стеку.

Ці рішення виходять за рамки простих перевірок «вгору чи вниз». Вони контролюють:

– Системи зберігання даних: продуктивність та ємність SAN, NAS та хмарних сховищ.

– Обчислювальні ресурси: фізичні сервери, віртуальні машини (ВМ) та контейнери.

– Мережі: використання пропускної здатності, затримка та підключення.

– Хмарні середовища: гібридні та мультихмарні архітектури.

Забезпечуючи видимість у режимі реального часу, ці інструменти дозволяють адміністраторам сховищ та інженерам виявляти вузькі місця, запобігати збоям та забезпечувати критично важливі програми необхідними ресурсами.

1.2 Область застосування

Для інженерів зі зберігання даних та директорів з IT-інфраструктури видимість – це різниця між проактивним управлінням та реактивним гасінням пожеж:

– Забезпечення надійності системи. Простої дорого коштують. Вони шкодять репутації та зупиняють продуктивність. Рішення для моніторингу

дозволяють виявляти аномалії, такі як сплеск затримки на критичному томі, перш ніж вони призведуть до повноцінних збоїв.

– Оптимізація витрат на зберігання. Надмірне виділення ресурсів – поширена та дорога звичка, що виникає через брак даних. Гарне рішення для моніторингу допомагає оптимізувати використання сховища, виявляючи навантаження типу «зомбі» та холодні дані, які можна перемістити на дешевші рівні, що допомагає досягти економічно ефективної роботи.

– Скорочення середнього часу до вирішення проблеми (MTTR). Без єдиного підходу до усунення несправностей виникає ситуація, коли команди мережі, сховища та розробників додатків обмінюються провинами. Комплексний моніторинг зіставляє дані з усіма компонентами стеку, миттєво виявляючи першопричину, щоб ви могли її виправити та повернутися до роботи.

Щоб по-справжньому оптимізувати та покращити IT-операції, переконайтеся, що обрана вами платформа має такі можливості:

– Гетерогенна підтримка: можливість моніторингу сховища та обчислень від будь-якого постачальника в одному поданні.

– Топологічне відображення: Візуалізація залежностей є критично важливою. Вам потрібно точно знати, який масив сховища підтримує яку програму, щоб ефективно виправляти несправності.

– Інтелектуальні сповіщення: Шукайте аналітичні дані на базі штучного інтелекту, які зменшують шум. Вам потрібні сповіщення про реальні проблеми, а не про статичні порушення порогових значень, які захаращують вашу поштову скриньку.

– Прогнозування потужностей: функції прогнозного планування на основі даних, які точно повідомляють вам, коли у вас закінчиться місце, виходячи з поточних тенденцій зростання.

– Інтеграція безпеки: функції, що відстежують шаблони доступу для покращення протоколів захисту даних та сприяння дотриманню вимог.

Майбутні тенденції в рішеннях для моніторингу інфраструктури

Майбутнє моніторингу автономне та прогнозне:

– AIOps та автоматизація: Ми переходимо від «сповіщень» до «вирішення проблем». Майбутні платформи не просто повідомлятимуть вам про переповнення диска; вони запускатимуть автоматизовані робочі процеси для розширення тома або очищення тимчасових файлів.

– Спостережуваність проти моніторингу: Зсув відбувається в бік «спостережуваності» – розуміння внутрішнього стану системи на основі зовнішніх виходів. Це критично важливо для налагодження складних мікросервісів та розподілених систем зберігання даних.

– Моніторинг сталого розвитку: Оскільки зелені IT стають пріоритетом, рішення для моніторингу все частіше відстежуватимуть вуглецевий слід та енергоефективність поряд з показниками продуктивності.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних IT-комплексів, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

ManageEngine Applications Manager

Менеджер додатків ManageEngine – це інструмент для моніторингу цифрового досвіду та продуктивності додатків. Хоча цей інструмент орієнтований на моніторинг додатків, він також контролює IT-інфраструктуру в тій мірі, в якій інфраструктура впливає на стан справ та продуктивність додатків.

ManageEngine пропонує дві різні версії цього інструменту. Професійна версія призначена для організацій, яким потрібно моніторити менше 500 програм, тоді як версія Enterprise може обробляти понад 500 програм і масштабується до 10 000 моніторів.

Особливості

Менеджер додатків – це, перш за все, інструмент моніторингу UX, який виконує як штучний моніторинг, так і моніторинг реальних користувачів. Крім того, інструмент може відстежувати URL-адреси для виявлення помилок, які можуть виникнути, або несанкціонованих змін вмісту веб-сайту. Інструмент також здатний відстежувати кінцеві точки REST API для виявлення простоїв та відстежувати запити Simple Object Access Protocol.

Менеджер додатків може контролювати різноманітні компоненти ресурсів інфраструктури, включаючи сервери, конвергентну інфраструктуру, віртуальні сервери та контейнери. Аналогічно, інструмент може контролювати різні служби, зокрема бази даних, пакети ERP, екземпляри Hadoop та Spark, проміжне програмне забезпечення тощо.

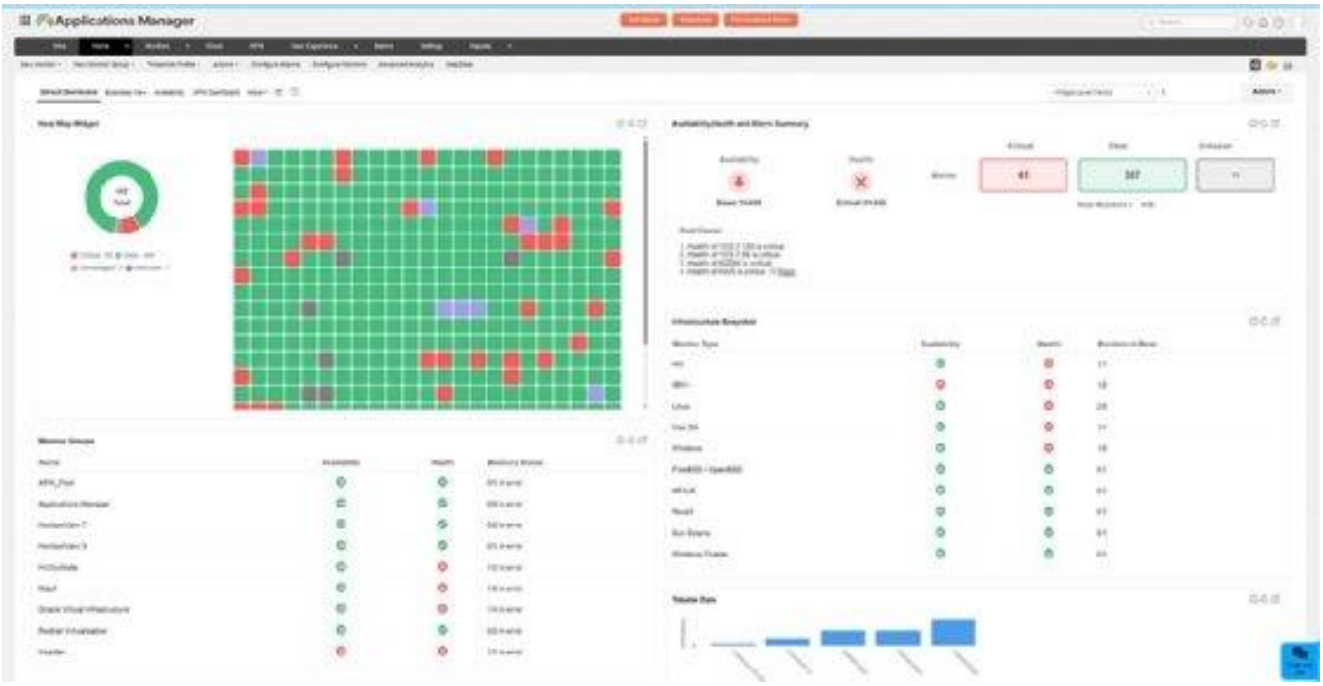


Рисунок 2.1 – Інтерфейс користувача ManageEngine Application Monitor

ManageEngine Application Monitor забезпечує комплексний уніфікований моніторинг, але його інтерфейс може мати певні труднощі з навчанням.

Переваги:

– Менеджер програм забезпечує комплексний уніфікований моніторинг.

– Він добре працює для моніторингу як традиційних, так і гібридних інфраструктур.

– Програмне забезпечення може пропонувати аналітичні дані на рівні коду, коли виникають вузькі місця.

Недоліки:

– Існує складний інтерфейс користувача з потенційно крутою кривою навчання.

– Немає стільки власних інтеграцій, скільки для деяких конкуруючих продуктів.

Варіанти використання

ManageEngine Application Manager найкраще підходить для організацій, які зацікавлені у забезпеченні справності та належної роботи своїх програм.

Інструмент може бути особливо корисним для організацій, які створюють власні програми.

Catchpoint

У той час як багато інструментів моніторингу зосереджені саме на моніторингу локальних та хмарних ресурсів організації, Catchpoint намагається контролювати весь інтернет-стек, включаючи SaaS-додатки, DNS-сервери, мережі доставки контенту, WAN, API тощо. Ідея полягає в тому, що маючи понад 3000 агентів по всьому світу, організації можуть бачити інтернет так само, як їхні віддалені користувачі та клієнти. А якщо проблема все ж таки виникне, Catchpoint може використовувати аналітику першопричин за допомогою штучного інтелекту, щоб швидко знайти джерело проблеми.

Особливості

Catchpoint розроблено для забезпечення стійкості додатків першого рівня шляхом моніторингу доставки та продуктивності додатків. При цьому він може вирішувати проблеми з підключенням як в Інтернеті, так і в корпоративних мережах. Фактично, Catchpoint дозволяє організаціям контролювати глобальну продуктивність Інтернету. Це може бути особливо корисним для організацій зі співробітниками або клієнтами, розташованими по всьому світу. Хоча Catchpoint приділяє значну увагу моніторингу Інтернету, програмне забезпечення може використовувати корпоративних агентів для моніторингу ресурсів в офісах, хмарах та центрах обробки даних.

Переваги:

– Оскільки Catchpoint моніторить Інтернет, він потенційно може виявляти проблеми з публічними мережами, які впливають на робочі навантаження.

– Catchpoint виконує як реальний, так і штучний моніторинг користувачів.

– Інструмент може контролювати робочі навантаження штучного інтелекту, такі як моделі великих мов (LLM) та чат-боти.

Недоліки:

– Початкова ціна Catchpoint становить 10 000 доларів на рік, що може бути занадто високою для стартапів.

– Catchpoint, ймовірно, є зайвим для менших сайтів.

Варіанти використання

Catchpoint ідеально підходить для великих підприємств з присутністю в електронній комерції або тих, у яких віддалені співробітники по всьому світу.

Grafana Cloud

Grafana Cloud – це хмарна платформа для спостереження та моніторингу. Інструмент збирає різні показники, які поєднує з даними журналів та трасуваннями, а потім відображає інформацію на серії інформаційних панелей. Інструмент також дозволяє здійснювати синтетичний моніторинг та відстеження продуктивності. Він може використовувати штучний інтелект для отримання аналітичних даних та генерувати сповіщення за необхідності.

Особливості

Панелі інструментів є центральним елементом хмарного інтерфейсу Grafana. Програмне забезпечення може створювати різноманітні барвисті та дуже деталізовані панелі інструментів, що відображають стан інфраструктури організації. Програмне забезпечення збирає ці дані за допомогою комбінації збору журналів, телеметрії та синтетичного моніторингу.

Grafana Cloud також активно використовує штучний інтелект та машинне навчання. Ці функції допомагають виявляти аномалії, що виникають в інфраструктурі, та допомагають у плануванні потужностей. Штучний інтелект також використовується для створення прогнозів, які можна відобразити на інформаційній панелі, дозволяючи користувачам бачити, як, ймовірно, поводитиметься їхня інфраструктура.

Функція під назвою OnCall об'єднує сповіщення від Grafana, Prometheus та AlertManager, об'єднуючи все в одному місці.

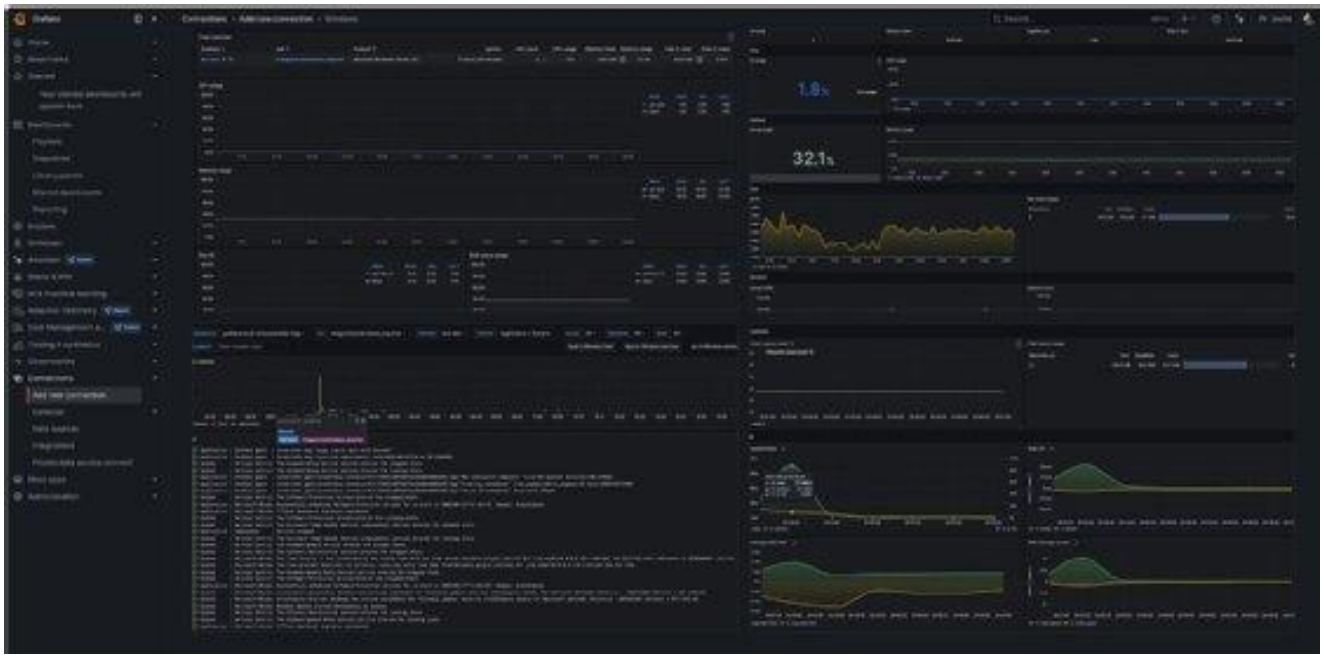


Рисунок 2.2 – Інтерфейс користувача Grafana Cloud

Grafana Cloud дозволяє користувачам переглядати різноманітні високодетальні інформаційні панелі, але вона все ще може бути менш настроюваною, ніж інші інструменти моніторингу.

Переваги:

– Хоча ціни для підприємств починаються від 25 000 доларів на рік, Grafana пропонує безкоштовний рівень для особистих проектів та професійну версію, яка починається від 19 доларів на місяць.

– Рівні корпоративного рівня підтримують єдиний вхід, інтеграцію протоколу Lightweight Directory Access Protocol та керування доступом на основі ролей.

– Grafana використовує штучний інтелект для виявлення аномалій та має LLM, який можна використовувати для допомоги у виконанні запитів.

– Існує величезна кількість підтримуваних інтеграцій.

Недоліки:

– Оскільки Grafana Cloud постачається як розміщений сервіс, він менш настроюваний, ніж інші інструменти моніторингу.

– Початковий процес налаштування може включати налаштування агентів та конвеєрів і може бути складним завданням.

Варіанти використання

Grafana Cloud може обслуговувати широкий спектр випадків використання, але найкраще він може підійти для хмарних середовищ, у яких трасування має виконуватися в розподілених системах, наприклад, коли мікросервіси використовуються інтенсивно. Він також добре підходить для команд DevOps, яким потрібні можливості глибокого моніторингу.

Kentik

Платформа мережевого моніторингу Kentik розроблена для надання консолідованого уявлення про стан та продуктивність мережі організації. Програмне забезпечення збирає ці дані, аналізуючи телеметрію потоку, хмарні журнали, дані протоколу SNMP (Simple Network Management Protocol) та синтетичні тести. Ця інформація збирається для ресурсів, що працюють як локально, так і в хмарі. Система забезпечує автоматизований аналіз та глибоке розуміння мережевих проблем, навіть у складних та розподілених мережах. Програмне забезпечення може навіть створювати візуальні карти мережі.

Kentik постачає своє програмне забезпечення на основі передплати, ціни на який починаються від 2000 доларів США на місяць за план Pro. Також існує план Premier. Однак для використання різних функцій потрібна додаткова ліцензія. Kentik пропонує безкоштовну 30-денну пробну версію свого програмного забезпечення.

Особливості

Програмне забезпечення Kentik багате на функції, а основний модуль надає такі можливості, як мережевий оглядач, оглядач даних, планувальник звітів, аналітика та сповіщення, доступ до API, планування потужностей та панель спостереження.

Ті, кому потрібні додаткові можливості, можуть використовувати додаткові модулі. Багато з цих модулів включено у вартість підписки, але деякі вимагають додаткової підписки. Наприклад, модуль Kentik Protect забезпечує

захист від DDoS-атак. Модуль Kentik Cloud допомагає з міграцією в хмару, перевіркою безпеки, оптимізацією продуктивності та зниженням витрат.

Також існує модуль під назвою Kentik Protect Advanced, який надає функції безпеки, що виходять за рамки стандартного модуля Kentik Protect. Цей модуль обробляє аналітику, виявляє ботнети та допомагає у пом’якшенні наслідків та реагуванні на інциденти.

Модуль під назвою Kentik Synthetics забезпечує моніторинг продуктивності та цифрового досвіду, а також виконує безперервне тестування. Модуль Kentik NMS надає детальнішу інформацію про мережеві інтерфейси та протоколи, тоді як модуль Kentik Edge контролює граничний трафік та виконує оптимізацію витрат.

Переваги:

– Програмне забезпечення забезпечує дисплей з однією панеллю та підтримує мережі від багатьох постачальників та гібридні мережі.

– Телеметрія може надаватися різними джерелами, включаючи SNMP, NetFlow/IPFIX, хмарні журнали та синтетичне тестування.

– Автоматизована аналітика та насичена візуалізація програмного забезпечення допомагають швидко виявляти проблеми.

Недоліки:

– Ціна програмного забезпечення може зробити його недоступним для невеликих організацій.

– Використання програмного забезпечення може бути складним завданням.

Варіанти використання

Kentik найкраще підходить для середніх та великих підприємств, яким потрібна глибока видимість у розгалужених та дуже складних розподілених гібридних мережах.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для розробки програмних модулів, збору телеметрії та реалізації підсистем спостережуваності (Observability) гібридних хмарно-корпоративних IT-комплексів обрано мову програмування Python. Це динамічна інтерпретована об'єктно-орієнтована мова загального призначення з міцною екосистемою та лаконічним синтаксисом. Офіційні дистрибутиви та документація доступні на сайті.

Завдяки мультиплатформенності та широкій стандартній бібліотеці Python дозволяє ефективно працювати з операційними системами, мережевими протоколами, базами даних і хмарними API. У межах архітектури моніторингу на базі Python створюються як легкі скрипти автоматизованої перевірки доступності вузлів, так і складні розподілені платформи агрегації метрик.

Оскільки Python є інтерпретованою мовою (де вихідний код транслюється у внутрішній байт-код для виконання інтерпретатором), його "чиста" швидкість виконання нижча порівняно з компільованими мовами на кшталт C/C++. Проте висока продуктивність розробки, гнучкість і зручність підтримки коду в сучасних хмарних рішеннях значно переважають чинник швидкодії виконання. Крім того, критичні до навантаження обчислювальні модулі (наприклад, для глибокого аналізу потоків логів чи швидкої обробки метрик) легко розширюються за допомогою вбудовуваних бібліотек, написаних на C або C++.

Процесор є ключовим компонентом обчислювальної системи, що виконує інструкції програм шляхом реалізації арифметичних і логічних операцій. Процесор оперує виключно машинним кодом (мовою низького рівня), проходячи через стандартний цикл обробки інструкцій:

1. Отримати (Fetch): завантаження наступної інструкції з пам'яті в процесор.

2. Декодувати (Decode): розпізнавання двійкового коду для визначення

типу операції.

3. Виконати (Execute): безпосереднє виконання команди центральним процесором.

Для спрощення взаємодії з апаратним забезпеченням замість складної машинного мови спочатку створювалися мови асемблера (з використанням мнемокодів), а згодом – високорівневі мови програмування (такі як Python), які абстрагують розробника від деталей архітектури центрального процесора, дозволяючи зосередитися на бізнес-логіці та алгоритмах моніторингу IT-інфраструктури.

Для проєктування, написання та відлагодження програмних компонентів системи спостережуваності використовуються різноманітні текстові редактори та середовища розробки:

– IDLE: стандартне середовище, що постачається разом із базовим пакетом Python.

– Visual Studio Code: легкий кросплатформний редактор із потужною екосистемою розширень для хмарних технологій та Python.

– PyCharm: професійне інтегроване середовище розробки (наявна повнофункціональна та урізана Community-версії), що забезпечує глибокий аналіз коду, підтримку систем контролю версій та хмарних інтеграцій.

– Geany, Notepad++, Thonny, Mu, PyScripter та Wing IDE: додаткові редактори та спеціалізовані середовища, що забезпечують зручні умови для розробки та тестування програмного забезпечення.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних IT-комплексів.

В процесі розробки випускної кваліфікаційної роботи за другим

(магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Системи моніторингу IT визначаються як інструменти, які постійно відстежують стан, продуктивність та безпеку інфраструктури та програм, щоб запобігти збоям, перш ніж вони вплинуть на роботу. Вибір правильних типів систем моніторингу IT є одним з найважливіших рішень, які приймає IT-менеджер. Комплексна програма вимагає охоплення шести основних рівнів, щоб уникнути сліпих зон, які проявляються як інциденти: сервер і хост, мережа, хмарна інфраструктура, контейнер і Kubernetes, продуктивність програм і моніторинг безпеки. Кожен рівень відповідає за окремий операційний ризик. Організації, які контролюють лише один або два рівні, наражають себе на ризик збоїв, які вони просто не можуть передбачити.

Шість додаткових рівнів моніторингу формують основу повноцінної спостережуваності в IT. Зрілі команди інженерів інтегрують щонайменше три-чотири з них, замість того, щоб покладатися на ізольовані інструменти. Кожен тип фіксує показники, які інші не можуть, тому їх поєднання забезпечує набагато кращу видимість, ніж будь-яка окрема система окремо.

1. Моніторинг серверів та хостів. Моніторинг серверів відстежує використання процесора, споживання пам'яті, операції вводу-виводу на диск та стан процесів на фізичних та віртуальних машинах. Це базовий рівень. Без нього неможливо співвіднести уповільнення роботи програм з виснаженням основних ресурсів.

2. Моніторинг мережі. Рішення для моніторингу мережі відстежують потоки трафіку, використання пропускної здатності, затримку, втрату пакетів та доступність пристроїв на комутаторах, маршрутизаторах та брандмауерах. Цей рівень відповідає на питання, чи проблеми з підключенням спричиняють збої програм, чи винен сам додаток. Найкращі методи моніторингу мережі на 2025

рік і далі приділяють значну увагу видимості на рівні протоколу, а не лише перевіркам доступності на основі ping.

3. Моніторинг хмарної інфраструктури. Сервіси моніторингу хмарних технологій відстежують віртуальні машини, керовані бази даних, сховища об'єктів та хмарні сервіси від таких постачальників, як AWS, Azure та Google Cloud. Хмарні середовища змінюються швидше, ніж локальна інфраструктура. Без спеціалізованого моніторингу хмарних технологій розростання ресурсів та неправильно налаштовані сервіси залишаються непоміченими, доки не спричинять збій або інцидент безпеки.

4. Моніторинг контейнерів та Kubernetes. Контейнери за своєю суттю є тимчасовими. Под може розкрутитися, вийти з ладу та перезапуститися за лічені секунди, що робить традиційний моніторинг на основі агентів неадекватним. Моніторинг контейнерів фіксує метрики оркестрації, стан подів, квоти ресурсів та події на рівні кластера. Цей тип не підлягає обговоренню для будь-якої організації, яка використовує мікросервіси у продакшені.

5. Моніторинг продуктивності застосунків (APM). Інструменти APM вимірюють час відгуку транзакцій, рівень помилок та карти залежностей у коді програми. Моніторинг реальних користувачів (RUM) фіксує фактичну продуктивність пристроїв кінцевих користувачів, виявляючи «повільний хвіст» найгірших сеансів, які синтетичні тести повністю пропускають. RUM та синтетичний моніторинг доповнюють один одного: синтетичні тести підтверджують доступність за розкладом, тоді як RUM показує, що насправді відчувають реальні користувачі.

6. Моніторинг безпеки. Моніторинг безпеки охоплює аналіз журналів, виявлення вторгнень, сканування вразливостей та кореляцію даних про загрози. Він розташований на всіх інших рівнях, а не під ними. Посібник із заходів IT-безпеки на 2026 рік чітко вказує, що моніторинг безпеки має бути інтегрований з даними інфраструктури, а не розглядатися як окрема функція аудиту.

Як автоматизований моніторинг мережі покращує управління IT-системами?

Ручний моніторинг мережі не спрацьовує у великих масштабах. Ручне зіставлення OID та налаштування, специфічне для певного постачальника, є основними причинами збоїв у розгортанні моніторингу мережі. Автоматизація усуває обидві проблеми.

Технологія Zero-touch discovery автоматизує класифікацію обладнання та застосовує правильні шаблони моніторингу без втручання людини. Сканування виконується двічі на день, а це означає, що щойно додані пристрої з'являються в інвентаризації моніторингу протягом кількох годин, а не чекають на наступний цикл ручного аудиту. Для організацій, які керують сотнями комутаторів і точок доступу на кількох сайтах, це різниця між актуальним і точним інвентаризацією пристроїв та тим, що постійно застаріває.

Автоматична перекласифікація пристроїв запобігає зміщенню конфігурації у великих та динамічних мережах. Коли комутатор замінюється або прошивка оновлюється, система моніторингу виявляє зміни та відповідно оновлює свої шаблони. Без цієї можливості команди успадковують застарілі конфігурації, які генерують хибні сповіщення або, що ще гірше, повністю пропускають справжні збої.

Підсистеми моніторингу SNMP, що інтегрують виявлення, класифікацію та опитування в єдиному конвеєрі, усувають потребу в окремих агентах та інтерфейсах. Це спрощує адміністрування та підвищує точність інвентаризації пристроїв. Основи автоматизації мережі тепер є необхідною умовою для будь-якої IT-команди, яка керує інфраструктурою в масштабі підприємства:

– Безконтактне виявлення усуває зусилля з ручної адаптації для нових пристроїв

– Двічі щоденне сканування підтримує актуальність інвентарю пристроїв без втручання людини

– Автоматична перекласифікація запобігає зміщенню конфігурації після змін у апаратному забезпеченні

– Інтеграція SNMP з одним конвеєром значно зменшує адміністративні витрати

– Багатогрупове успадкування застосовує узгоджені політики моніторингу для всіх сімейств пристроїв.

Яку роль відіграють спостережуваність та автономні IT у сучасних системах моніторингу?

Моніторинг та спостережуваність – це не одне й те саме. Моніторинг підтверджує, чи система працює, чи не працює. Спостережуваність забезпечує глибоке, причинно-наслідкове розуміння поведінки системи, включаючи періодичні проблеми, які перевірки стану справності ніколи не виявляють. Ця відмінність важлива, оскільки більшість виробничих інцидентів не спричинені повними збоями. Вони спричинені незначними деградаціями, які сам лише моніторинг не може пояснити.

Втома від сповіщень є прямим наслідком збору необроблених показників без контексту. Коли IT-команда щодня отримує тисячі сповіщень, вона перестає довіряти системі. Критичні сповіщення залишаються без уваги серед шуму. Автономні IT-платформи вирішують цю проблему, співвідносячи сигнали інфраструктури з впливом на бізнес, перш ніж вивести сповіщення, щоб команда бачила лише те, що вимагає дій.

Застарілі фрагментовані набори інструментів створюють операційні труднощі. Інтегровані платформи об'єднують сигнали з бізнес-контекстом, що є єдиним способом ефективно масштабувати моніторинг у міру зростання інфраструктури. Мета полягає не в більшій кількості даних. Мета – у меншій кількості якісніших рішень, які приймаються швидше.

Які ключові характеристики слід порівнювати під час вибору систем IT-моніторингу?

Вибір систем моніторингу продуктивності вимагає оцінки конкретних можливостей, а не лише переліку функцій. У таблиці нижче наведено критерії, які відрізняють адекватні інструменти від справді ефективних.

Польові інструменти початкового рівня зазвичай охоплюють моніторинг серверів і мережі з базовими сповіщеннями. Корпоративні платформи додають APM, моніторинг контейнерів, інтеграцію хмарних сервісів та спостереження на основі штучного інтелекту в одному інструменті. Правильний вибір залежить від складності вашої інфраструктури, а не від розміру маркетингового бюджету постачальника.

Контрольний список мережевої інфраструктури – це практична відправна точка для визначення шарів, які охоплює ваш поточний набір інструментів, і де існують прогалини, перш ніж оцінювати нові рішення.

Як вибрати та поєднати системи IT-моніторингу для вашої організації

Зрілі інженерні команди інтегрують щонайменше три-чотири рівні моніторингу, замість використання ізольованих інструментів. Процес вибору повинен відповідати структурованому підходу, що ґрунтується на специфічній інфраструктурі вашої організації та профілі ризиків.

– Проведіть аудит вашого поточного покриття. Зіставте кожен рівень інфраструктури з шістьма основними типами. Визначте, які рівні взагалі не мають моніторингу, а які мають інструменти, що перетинаються та можуть бути об'єднані.

– Розставляйте пріоритети за критичністю для бізнесу. Якщо ваша організація залежить від веб-застосунку, орієнтованого на клієнта, APM та RUM мають вищий пріоритет, ніж моніторинг контейнерів. Якщо ви керуєте виробничою мережею, моніторинг мережі та моніторинг безпеки мають пріоритет.

– Створюйте систему шарами, а не все одразу. Спочатку розгорніть моніторинг серверів та мережі, щоб встановити базову лінію. Далі додайте моніторинг APM та безпеки. Моніторинг хмари та контейнерів слід виконувати після того, як базові шари будуть стабільними.

– Консолідуйте, де це можливо. Заходи хмарної безпеки та моніторинг інфраструктури працюють найкраще, коли вони використовують спільну

платформу даних. Фрагментовані набори інструментів створюють накладні витрати на інтеграцію та знижують якість міжрівневої кореляції.

– Встановіть порогові значення сповіщень з урахуванням бізнес-контексту. Визначте, що означає критичне сповіщення з точки зору впливу на клієнтів, а не лише з точки зору порогових значень показників. Це запобігає втомі від сповіщень з самого початку.

– Переглядайте та коригуйте щоквартально. Інфраструктура змінюється швидше, ніж моніторинг конфігурацій. Плануйте щоквартальні огляди, щоб виявити відхилення конфігурації, перш ніж це призведе до прогалин у моніторингу.

Бюджет є реальним обмеженням. Організаціям, які не можуть негайно розгорнути всі шість рівнів, слід надати пріоритет тим рівням, які найбільш безпосередньо пов'язані з послугами, що генерують дохід, та розширити охоплення, наскільки дозволяють ресурси.

Ключові висновки

Комплексний IT-моніторинг вимагає інтеграції рівнів моніторингу серверів, мереж, хмари, контейнерів, додатків та безпеки, оскільки кожен рівень виявляє збої, які інші не можуть виявити.

Чому більшість стратегій IT-моніторингу зазнають невдачі ще до їх початку

Організації інвестують в інструменти моніторингу, а потім недостатньо інвестують у процес налаштування та перевірки, який робить ці інструменти корисними. Проблема не в інструменті. Проблема в стратегії, яка його використовує. Найбільша помилка, яку я бачу, полягає в тому, щоб ставитися до моніторингу як до одноразового розгортання. Інфраструктура постійно змінюється. Комутатор замінюється, новий хмарний сервіс налаштовується, кластер контейнерів масштабується. Без автоматичної перекласифікації та регулярних перевірок конфігурації система моніторингу втрачає синхронізацію з реальністю. У цей момент у вас виникають витрати на моніторинг без його вигоди.

Друга помилка – це змішування обсягу сповіщень з якістю моніторингу. Більша кількість сповіщень не означає кращу видимість. Вони означають, що команда перестає довіряти власним інструментам. Організації, які, на мою думку, правильно це роблять, – це ті, хто інвестував час у визначення того, як виглядає змістовне сповіщення, перш ніж налаштувати єдиний поріг.

Спостережуваність – це напрямок, у якому рухається галузь, і це правильно. Але я б застеріг від того, щоб розглядати її як заміну надійного фундаментального моніторингу. Спочатку налаштуйте рівні вашого сервера, мережі та безпеки. Спостережуваність, побудована на хиткому фундаменті, створює складний шум, а не розуміння.

3.2 Розробка структурної схеми

Моніторинг – це практика збору, агрегації та аналізу заздалегідь визначеного набору сигналів від ваших систем, щоб знати, чи вони справні, та попередити людину про їхню несправність. Ключове слово – заздалегідь визначений. Ви заздалегідь вирішуєте, які сигнали мають значення (затримка запиту, коефіцієнт помилок, процесор, диск, глибина черги), створюєте панелі інструментів, які їх відображають, і встановлюєте пороги, які спрацьовують, коли сигнал перетинає навмисно намальовану вами межу. Моніторинг – це дисципліна спостереження за проблемами, які ви вже знаєте, як розпізнати.

Таке формулювання робить моніторинг скромним поруч із яскравішими ідеями, і він є скромним. Він також незамінний. Кожна надійна система в Інтернеті базується на рівні моніторингу, тому що перше завдання операцій – просто швидко та надійно дізнатися, що щось зламалося. Команда може мати найдосконалішу систему трасування та найрозумніший штучний інтелект у світі, але якщо нічого не стежить за вхідними дверима, першим сигналом збою буде розлючений клієнт. Моніторинг – це вхідні двері.

Варто точно визначити, що робить моніторинг, а що ні, оскільки наступний розділ присвячений саме цьому. Моніторинг відповідає на запитання,

яке ви вже знали: чи відбувається те, що я очікував, не так саме зараз? Він побудований навколо відомих режимів збоїв, диска, який заповнюється, залежності, яка перевищує час очікування, затримки, яка перевищує цільовий рівень обслуговування. Для кожного з них ви створюєте перевірку та поріг. Ось чому моніторинг іноді описують як обробку *відомо-невідомих* : ви знаєте категорію проблеми (дисковий простір), ви просто не знаєте, коли це станеться, тому ви постійно стежите за цим.

Люди іноді вважають, що зі зростанням спостережуваності моніторинг став застарілим. Правда навпаки. Спостережуваність дає вам можливість досліджувати проблеми, які ви ніколи не передбачали, але вона не стежить за вашими дверима і не викликає вас о 3-й ночі. Моніторинг залишається постійно увімкненою сіткою безпеки: дешевим, надійним, свідомим шаром, який стає червоним, коли виконується визначена вами умова, і виводить людину з ліжка. Ці два процеси не є суперниками. Як видно з наступного розділу, найрозумніші команди керують обома системами, і вони точно розуміють, де проходить межа між ними.

Моніторинг проти спостережливості: ключова відмінність

Саме ця відмінність бентежить більше команд, ніж будь-яка інша в операціях, тому варто зробити її абсолютно правильно. Найкоротша версія: моніторинг показує, що щось не так; спостережуваність допомагає зрозуміти чому, особливо коли причина ніколи не була на жодній інформаційній панелі. Вони вирішують різні проблеми, а вам потрібні обидві.

Моніторинг створено для відомих і невідомих ситуацій. Ви перераховуєте режими збоїв, які можете передбачити, створюєте попередньо визначену панель інструментів і попередньо визначене сповіщення для кожного з них, і постійно спостерігаєте за ними. Коли сигнал перетинає встановлений вами поріг, спрацьовує моніторинг. Перевага полягає в тому, що він дешевий, завжди доступний і миттєво зрозумілий; обмеження полягає в тому, що він може виявити лише ті проблеми, про які ви подумали заздалегідь. Якщо збій виникає внаслідок взаємодії, яку ніхто не передбачав, то на нього не чекає жодна панель

інструментів. Спостережуваність створена для невідомих. Спостережуваність – це не інструмент, який ви вмикаєте; це властивість системи. Система є спостережуваною, якщо ви можете ставити абсолютно нові запитання щодо її поведінки постфактум, не надсилаючи новий код для її інструментації. Коли всі панелі інструментів зелені, а клієнти все ще скаржаться, спостережуваність – це те, що дозволяє інженеру розбити телеметрію високої потужності за виміром, значення якого він щойно зрозумів: конкретний хеш збірки, один рівень клієнта, один регіон, і знайти причину, яку жодне попередньо визначене сповіщення не могло виявити. Щоб отримати повний огляд метрик, журналів і трас, розподіленого трасування та трьох стовпів, прочитайте спеціальний посібник зі спостережуваності; ця сторінка є фундаментальним доповненням до моніторингу.

Ці відносини є взаємодоповнювальними, а не конкуруючими. Моніторинг – це підмножина, яка обробляє все, що можна передбачити; спостережуваність – це надмножина, яка дозволяє вам обробляти все інше. На практиці вони підсилюють один одного: моніторинг говорить вам почати шукати приблизно де, а спостережуваність дозволяє вам копати, доки ви не знайдете причину. Зріла команда зберігає свої попередньо визначені панелі моніторингу та сповіщення про збої, які вона може передбачити, а також зберігає багаті дані спостережуваності під ними для тих, які вона не може передбачити. Помилка полягає в тому, щоб розглядати їх як «або/або»; правильне формулювання – це моніторинг *плюс* спостережуваність.

Однорядковий тест, щоб відрізнити їх. Якщо вам доводилося додавати нову метрику, відправляти розгортання та чекати повторення проблеми, перш ніж ви могли б її діагностувати, ви покладалися на моніторинг і досягали межі. Якщо ви могли негайно відповісти на нове запитання за допомогою вже наявної телеметрії, це була спостережуваність. Моніторинг – це те, що ви налаштовуєте до інциденту; спостережуваність – це те, що рятує вас під час інциденту, якого ви не передбачали.

Типи моніторингу та коли їх використовувати

«Моніторинг» – це охоплюючий підхід, що охоплює кілька різних дисциплін, кожна з яких спостерігає за різним рівнем стеку з різної точки зору. Досвідчені команди поєднують кілька з них, а не роблять ставку на одну. Ось практична карта:

– Інфраструктура / хост. Спостерігає за машинами: станом процесора, пам'яті, дисків, мережі та процесів на серверах, віртуальних машинах і контейнерах. Це найстаріша форма моніторингу та фундамент, на якому базується все інше. Використовуйте її, щоб виявляти виснаження ресурсів (заповнення диска, витік пам'яті), перш ніж це призведе до збою програми. Необхідно, але саме по собі це повідомляє про несправність системи, а не про те, чи постраждали користувачі.

– Застосування / APM. Моніторинг продуктивності застосунків відстежує ваш код: затримку запитів, коефіцієнт помилок, пропускну здатність та повільні транзакції на кінцеву точку. APM ближчий до користувача, ніж метрики хоста, оскільки він вимірює, що насправді робить ваш сервіс. Використовуйте його, щоб знайти повільні кінцеві точки, зростаючі коефіцієнти помилок та шлях коду, що призводить до погіршення якості. Зазвичай це найвищий рівень сигналу для раннього виявлення регресій, з якими стикається користувач.

– Мережа. Відстежує зв'язки між системами: пропускну здатність, затримку, втрату пакетів та з'єднання між хостами, регіонами та залежностями. Використовується, коли проблема виникає між компонентами, а не всередині одного, у випадку нестабільного з'єднання, перенасиченого шлюзу, хмарної мережі з шумним сусідом. Часто це прихована причина симптомів, які виглядають як повільна робота програми.

– Синтетичний (проактивний, зовні-всередину). Скриптовані зонди, які запускають критично важливі шляхи користувачів ззовні системи за розкладом, незалежно від того, чи протікає реальний трафік. Зонд входить у систему, додає до кошика та щохвилини перевіряє з кількох регіонів. Використовуйте його для

виявлення збоїв та перерваних потоків, перш ніж це зроблять реальні користувачі, а також для вимірювання доступності в години спокою. Основа моніторингу часу безвідмовної роботи та SLA.

– Моніторинг реальних користувачів (RUM). Вимірює фактичний досвід реальних людей на реальних пристроях та в мережах: завантаження сторінок, затримку взаємодії та помилки, коли користувачі їх використовують. Там, де синтетичні методи є послідовними та заснованими на сценаріях, RUM фіксує справжній, нечіткий розподіл продуктивності, включаючи повільний хвіст, який неможливо написати на сценаріях. Використовуйте його, щоб зрозуміти, як продукт насправді поводиться в польових умовах, особливо в найгірших 5% сеансів.

– Моніторинг журналів та часу безвідмовної роботи. Моніторинг журналів відстежує потоки журналів на наявність відомих шаблонів помилок та кількості певних подій. Моніторинг часу безвідмовної роботи або доступності відповідає на найпростіше питання: чи взагалі доступний сервіс, зазвичай із зовнішніх контрольних точок. Використовуйте моніторинг журналів для відомих сигнатур помилок та перевірок часу безвідмовної роботи як найпростіший можливий сигнал про те, що щось працює.

Жоден окремий тип не є достатнім. Моніторинг хоста без моніторингу програм показує, що сервер зайнятий, але не повідомляє про проблеми з клієнтами. Синтетичний моніторинг без RUM показує, що працює сценарний шлях, але не те, як реальні користувачі сприймають не сценарні шляхи. Правильним початковим набором для більшості команд є моніторинг програм/APM плюс метрики хоста плюс синтетичні перевірки критичних шляхів, потім RUM та решта в міру зростання системи.

Що слід контролювати: золоті сигнали, USE та RED

Найскладніша частина моніторингу – це не збір даних; сучасні системи випромінюють більше, ніж ви коли-небудь зможете спостерігати. Найскладніше – вибрати кілька сигналів, які дійсно відповідають проблемі користувача, та ігнорувати тисячі, які відповідають машинним дрібницям. Три

добре відомі методи забезпечують таку дисципліну.

Чотири золоті сигнали

Найпоширенішим фреймворком, згідно з практикою SRE від Google, є чотири золоті сигнали : затримка (скільки часу займають запити), трафік (скільки навантаження на систему), помилки (частота невдалих запитів) та насичення (наскільки заповнена система відносно її меж). Їхня сила полягає в тому, що вони описують сервіс з точки зору користувача. Якби ви могли спостерігати лише за чотирма показниками сервісу, зверніть увагу на ці, тому що разом вони відповідають на питання: «чи швидкі запити, успішні, і чи не закінчиться місце в системі?». Користувачі відчувають різке збільшення затримки або кількість помилок; сам графік процесора може цього не відчувати. Запускайте кожен новий сервіс із золотими сигналами та додавайте деталі звідти.

ВИКОРИСТАННЯ: для ресурсів

Метод USE від Брендана Грегга підходить для аналізу на рівні обладнання та ресурсів: для кожного ресурсу відстежуйте використання (наскільки він зайнятий), насичення (скільки роботи чекає на нього в черзі) та помилки. USE – це правильна лінза для процесора, пам'яті, диска та мережі на хості: вона вказує безпосередньо на вузьке місце ресурсу. Там, де золоті сигнали описують службу, USE описує машину під нею.

ЧЕРВОНИЙ: для послуг, що базуються на запитах

Метод RED від Тома Вілкі підходить для сервісів, керованих запитами, та мікросервісів: для кожного сервісу відстежуйте швидкість (кількість запитів за секунду), помилки (частоту невдалих запитів) та тривалість (розподіл затримки запитів). RED – це, по суті, золоті сигнали мінус насичення, налаштовані на однорідність для кожного сервісу, щоб ви могли створити один шаблон інформаційної панелі та використовувати його всюди. Це практичний шаблон за замовчуванням для парку мікросервісів.

Методи навмисно перетинаються; золоті сигнали поєднують у собі податок на ресурси USE та податок на запити RED. Дисципліна, яку вони всі

застосовують, однакова: вибирайте сигнали, що відповідають проблемі користувача, а не дрібницям машини. Високий показник процесора вартий сповіщення лише в тому випадку, якщо він уповільнює або призводить до збоїв у роботі запитів; сам по собі він є інформацією, а не інцидентом. Виберіть кілька сигналів, які, коли вони рухаються, означають, що клієнт переживає гірше, а решту нехай відображаються на інформаційних панелях, до яких ви звертаєтесь під час розслідування, а не сповіщеннями, які будять людей.

Дивіться, як ваші золоті сигнали автоматично співвідносяться з інцидентами, а їхня перша причина вже визначена.

Оповіщення виконано правильно

Збір сигналів – це найлегша частина моніторингу. Вирішивши, які з них повинні розбудити людину, команди або заслужать довіру до свого пейджера, або знищать його. Погані сповіщення гірші, ніж їх відсутність, тому що команда, яка навчилася ігнорувати свої сповіщення, пропустить те, яке мало значення. Три принципи відрізняють хороші сповіщення від шуму.

Сповіщення про симптоми, а не про причини. Сповіщення на основі симптомів спрацьовує при видимому користувачем болю, зростанні затримки, зростанні рівня помилок, невдалій синтетичній перевірці. Сповіщення на основі причин спрацьовує при внутрішній умові, яка може мати значення, а може й ні: високому завантаженні процесора, переповненому кеші, зайнятому диску. Проблема зі сповіщеннями на основі причин полягає в тому, що одна й та сама причина іноді є нешкідливою, а іноді катастрофічною, тому вони постійно генерують хибні тривоги. Високе завантаження процесора під час запланованого пакетного завдання – це нормально; високе завантаження процесора, яке уповільнює перевірку, – це інцидент, а сповіщення про симптоми виявляє саме другий випадок. Переглядайте сторінку симптому; тримайте причини на інформаційних панелях, щоб ознайомитися з ними після того, як ви вже розслідуєте.

Кожне сповіщення має бути таким, що вимагає дії, та пов’язаним з дією людини. Це кардинальне правило сповіщень: якщо спрацьовує сповіщення, і

людина, до якої надсилається сповіщення, нічого не може чи не повинна з цим зробити прямо зараз, воно не повинно бути сповіщенням. Сповіщення без жодних дій – це шум, який навчає команду ігнорувати пейджер. Будь-яка інформація має бути на інформаційній панелі або в черзі заявок, а не на телефоні чергування. Перш ніж створювати сповіщення, дайте відповідь на одне запитання: що робитиме відповідальна особа, коли це спрацює? Якщо ви не можете назвати дію, у вас немає сповіщення, у вас є сповіщення, і воно не повинно нікого викликати на пейджер.

Використовуйте рівень серйозності для маршрутизації, а не для спаму. Не все заслуговує на однакову реакцію. Чітка модель серйозності, виклик зараз проти сповіщення в робочий час проти реєстрації на потім, дозволяє вам резервувати виклик о 3 ранку для речей, які дійсно не можуть чекати. Це основа здорового чергування : людей викликають лише для вирішення проблем, що впливають на користувачів, а все інше перетікає в спокійніше русло.

Якщо ви помилитеся, ви потрапите у стан втоми від сповіщень – стан, коли спрацьовує так багато малозначущих сповіщень, що служби реагування починають ігнорувати їх усі, включаючи справжні. Втома від сповіщень – це найпоширеніший спосіб, через який моніторинг на практиці не працює: не тому, що система перестала збирати дані, а тому, що люди перестали довіряти пейджеру. Виправлення полягає у безжальній дисципліні щодо того, що дозволено надсилати на пейджер. Наш повний посібник із втоми від сповіщень та способів її виправлення глибоко розглядає тактику. Принцип, який слід дотримуватися з цього розділу: кожне сповіщення, яке будить когось, має вимагати людського судження або дії, крапка.

Панелі інструментів та візуалізація

Якщо сповіщення – це спосіб моніторингу привертати вашу увагу, то інформаційні панелі – це спосіб її утримувати. Гарна інформаційна панель швидко відповідає на запитання; погана ховає відповідь під стіною графіків, які ніхто не читає. Різниця зводиться до чіткої ієрархії.

Ієрархія «від огляду до деталей». Організуйте інформаційні панелі таким

чином, щоб спеціаліст з реагування переходив від вердикту про стан системи до конкретних деталей, і ніколи не навпаки. Вершина ієрархії – це єдиний огляд, який одним поглядом відповідає на запитання «чи справна система?», зазвичай це золоті сигнали для ключових служб. Звідти спеціаліст з реагування заглиблюється в інформаційну панель служби, потім у підсистему, а потім у певну метрику. Ієрархія означає, що нікому не потрібно сканувати все; вони йдуть шляхом від симптому до деталі. Це структура, яка перетворює купу метрик на зручний діагностичний інструмент.

Панель керування сервісом – це найважливіший елемент. Кожна служба повинна мати одну основну панель керування, яка приблизно за десять секунд відповідає на запитання «чи справна ця служба?», розташовану зверху вниз від золотих сигналів до допоміжних деталей. Це панель керування, яку чергування відкривається першою, коли з'являються сторінки цієї служби. Створіть її один раз як шаблон багаторазового використання (метод RED спрощує це, оскільки кожна служба використовує однакові три сигнали), і кожна служба в автопарку стає читабельною таким самим чином.

Антишаблон: стіна графіків. Найпоширенішим недоліком інформаційної панелі є стіна графіків: десятки або сотні панелей, набитих на одному екрані, оскільки кожна метрика «може колись стати в нагоді». В результаті одна панель, яка має значення під час інциденту, губиться в шумі, і служби реагування витрачають дорогоцінні хвилини на сканування замість діагностики. Інформаційна панель – це не архів даних; це допоміжний засіб для прийняття рішень. Якщо панель не допомагає комусь прийняти рішення під час інциденту, вона має бути в дослідницькому вигляді, а не на основній інформаційній панелі. Надайте перевагу невеликій кількості панелей з високим рівнем сигналу в чіткій ієрархії, а не вичерпній сітці, яку ніхто не може прочитати під тиском.

Десятисекундний тест. Відкрийте основну панель інструментів сервісу та запитайте: чи може респондент, який ніколи раніше її не бачив, визначити, чи сервіс справний, протягом десяти секунд? Якщо так, то ієрархія працює. Якщо їм доведеться переглянути двадцять панелей, щоб сформувати вердикт, панель

інструментів являє собою стіну графіків, і це коштуватиме вам хвилин MTTR на кожен інцидент. Створіть верхню частину кожної панелі інструментів так, щоб спочатку виводити вердикт про справність, а деталі розмістіть під ним.

Перехід до 2026 року: від статичних порогів до інтелектуального моніторингу

Все, що досі описує моніторинг, як він працював протягом двох десятиліть: люди вибирають сигнали, малюють статичні пороги, створюють інформаційні панелі та спостерігають. Ця модель є основоположною і нікуди не зникне, але у 2026 році вона буде доповнена таким чином, що змінить значення «моніторингу» у повсякденній практиці.

Межа статичних порогів. Статичний поріг – це фіксована лінія: сповіщення, якщо затримка перевищує 500 мс, сповіщення, якщо завантаження процесора перевищує 80%. Статичні пороги ідеально підходять для справжніх жорстких лімітів, диск, який заповнений, є заповненим, сертифікат, термін дії якого минув, є минувшим, і ви повинні залишати їх там. Але для всього, що змінюється залежно від навантаження, одне фіксоване число є занадто точним. Рівень процесора, який є абсолютно нормальним о 14:00, є тривожним о 3:00, а плоский поріг не може слідувати сезонним, тижневим або щоденним закономірностям. Результатом є знайомий біль статичних порогів: хибні тривоги, коли нормальне навантаження ненадовго перевищує лінію, та пропущені інциденти, коли реальна проблема залишається трохи нижче неї.

Динамічне базове визначення та виявлення аномалій. Перша частина зміни полягає в заміні фіксованих ліній вивченими базовими рівнями. Замість одного числа система вивчає, як виглядає норма для кожного сигналу в різні часи доби, дні тижня та нещодавню тенденцію, і позначає справжні відхилення від цієї вивченої закономірності. Це область виявлення аномалій : вона зменшує шум статичних порогів (оскільки щоденний пік розпізнається як норма) та виявляє проблеми, які пропустила б рівна лінія (оскільки невелике, але аномальне відхилення виділяється на тлі вивченої базової лінії). Для систем, навантаження яких змінюється, динамічні базові лінії просто краще підходять,

ніж число, яке людина вгадала один раз і більше ніколи не переглядала.

Від виявлення до автономного реагування. Глибший зсув відбувається після того, як сигнал виглядає неправильним. У класичній моделі кожне цікаве відхилення стає окремим сповіщенням, людина співвідносить їх вручну, відкриває інформаційні панелі, знаходить причину та діє. Напрямок 2026 року полягає в тому, щоб стиснути цей цикл: співвіднести пов'язані сигнали з усього середовища в один інцидент, автоматично визначити ймовірну першопричину та вирішити добре зрозумілі випадки в межах політики, щоб людина бачила лише справді нові.

Розроблюваний у даній роботі комплекс збирає ваші дані моніторингу з AWS, GCP, Azure, Linux та Windows, співвідносить багато окремих сигналів в один інцидент, а не потік непов’язаних сповіщень, визначає ймовірну першопричину з її походженням та автоматично вирішує рутинні інциденти в межах визначеної вами політики. В результаті моніторинг припиняється, що означає «людина дивиться на інформаційні панелі, чекаючи, коли щось стане червоним», і починає означати «менше, кращих, вже діагностованих сторінок». Ваші інформаційні панелі, ваші золоті сигнали та ваші сповіщення залишаються; Nova перетворює їхній вихід на корельовані, першопричинені та часто вже вирішені інциденти. Щоб дізнатися ширшу категорію, до якої це належить, див. AIOps, а для практик, які він використовує, – скорочення MTTR та управління інцидентами.

90-денний план впровадження та контрольний список із 10 пунктів

Практична послідовність для моніторингу, яка забезпечує цінність на ранніх етапах, не заглушаючи команду шумом. Принцип, що проходить повз увагу: почніть із сигналів, що відповідають за проблеми користувачів в одній важливій послузі, запровадьте дисципліну сповіщень з самого початку, а потім розширте її.

Дні 1-14: Інструментуйте одну послугу за допомогою золотих сигналів

Виберіть один важливий сервіс і отримайте його чотири золоті сигнали (затримка, трафік, помилки, насиченість), а також показники хоста для машин під ним. Створіть одну основну панель інструментів сервісу, яка відповідає на запитання «чи справний він?» за десять секунд. Мета: довести, що конвеєр працює, і команда може з першого погляду оцінити стан сервісу. Не намагайтеся контролювати все; один добре контрольований сервіс навчить більше, ніж десять напівконтрольованих.

Дні 15-45: Додавання синтетичних перевірок та дисциплінованих сповіщень

Додайте синтетичні зонди на критично важливому шляху користувача, щоб виявляти поломки раніше, ніж це зроблять користувачі, та встановлюйте перевірки безвідмовної роботи як основний сигнал. Потім напишіть свої перші сповіщення та пишіть їх суворо: на основі симптомів, практичні, кожне з яких пов’язане з визначеною дією людини, з чіткою моделлю серйозності. Не піддавайтеся спокусі видавати сповіщення за кожною метрикою. Дисципліна сповіщень, яку ви встановите зараз, запобігає втомі від сповіщень пізніше.

Дні 46-75: Розширення охоплення та додавання неоново-цистного препарату (RUM)

Розгорніть шаблон golden-signals по всьому шляху критичного запиту, щоб кожна послуга була зрозумілою однаково. Додайте моніторинг реальних користувачів, щоб зрозуміти фактичний польовий досвід, включаючи повільні хвостові синтетичні перевірки, які не бачать. Створіть ієрархію панелі інструментів від загального огляду до деталей, щоб реагуючі переходили від вердикту верхнього рівня до деталей. Перегляньте кожне створене на даний момент сповіщення та видаліть ті, які ніколи не призводили до дії.

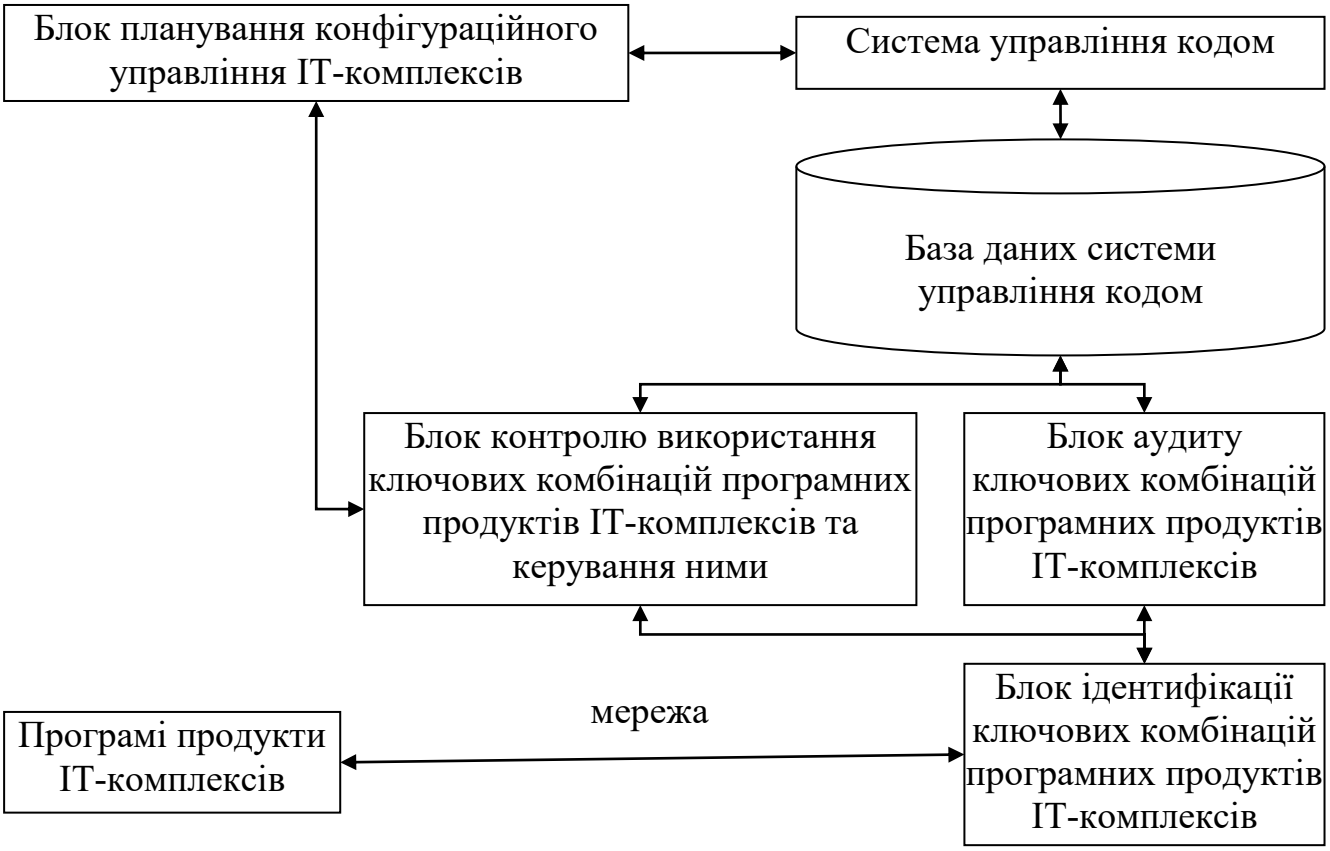


Рисунок 3.1 – Структурна схема системи

Дні 76-90: Перехід від статичних порогів до інтелектуального моніторингу

Замініть статичні порогові значення для сигналів, що змінюються в навантаженні, динамічними базовими рівнями та виявленням аномалій, щоб сповіщення відстежували реальні відхилення, а не довільні лінії. Зв'яжіть сигнал з кореляцією та реагуванням: саме тут платформа, така як Nova AI Ops, використовує дані моніторингу для співвіднесення сповіщень з інцидентами, пошуку першопричини та автоматичного вирішення рутинних випадків, щоб інвестиції в моніторинг перетворилися на меншу кількість сторінок та швидше вирішення, а не просто на більшу кількість графіків. Задокументуйте кількість сторінок до/після та MTTR, щоб обґрунтувати розширення охоплення.

Контрольний список зрілості з 10 пунктів для моніторингу

Чесно оцініть себе. Кожне «так» – це рівень зрілості; прогалини – це ваша дорожня карта.

– Чи мають ваші критично важливі сервіси золоті сигнали? Затримка, трафік, помилки та насиченість для кожного важливого сервісу, а не лише показники хоста.

– Ви також здійснюєте моніторинг з боку користувача? Синтетичні перевірки критичних шляхів та моніторинг реальних користувачів, а не лише показники інфраструктури зсередини.

– Ваші сповіщення базуються на симптомах? Вони спрацьовують на видимому користувачем болю, а не на внутрішніх причинах, які можуть бути нешкідливими.

– Чи кожне сповіщення має бути дієвим? Кожне сповіщення, яке надсилає людині повідомлення, пов’язане з чіткою дією; інформаційні сигнали відображаються на інформаційних панелях, а не на пейджері.

– У вас є модель серйозності? Надсилати сторінку зараз, сповіщати чи реєструвати, щоб сторінка о 3 ранку була зарезервована для того, що дійсно не може чекати.

– Чи має кожна служба одну чітку основну інформаційну панель? Десятисекундний вердикт про стан здоров'я, а не стіну графіків, яку ніхто не читає.

– Чи структура вашої інформаційної панелі є повністю деталізованою? Респонденти переходять від вердикту найвищого рівня до деталей, а не сканують все.

– Чи вийшли ви за рамки статичних порогів там, де це допомагає? Динамічні базові лінії та виявлення аномалій на сигналах, що змінюються в навантаженні, статичні пороги лише для справжніх жорстких обмежень.

– Чи пов’язані сповіщення пов’язані з інцидентами? Один інцидент з контекстом, а не потік незв’язаних сповіщень, які людина зшиває вручну.

– Чи сигнал спонукає до дій? Моніторинг забезпечує кореляцію, аналіз

першопричин та, в ідеалі, автономне усунення недоліків, а не лише панелі моніторингу, за якими ніхто не стежить.

Більшість команд мають приблизно п'ять або шість таких команд. Проміжок між шістьма та десятьма – це той момент, коли моніторинг перестає бути стіною графіків та галасливим пейджером і починає вимірювано виявляти проблеми на ранній стадії та скорочувати час їх вирішення.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно. Розроблений комплекс спостережуваності та моніторингу забезпечує безперервний контроль доступності, продуктивності та безпеки критично важливих елементів гібридної IT-інфраструктури підприємства. Система інтегрує інструменти глибокого аналізу мережевих потоків, віртуалізованих середовищ, серверного обладнання та прикладних сервісів.

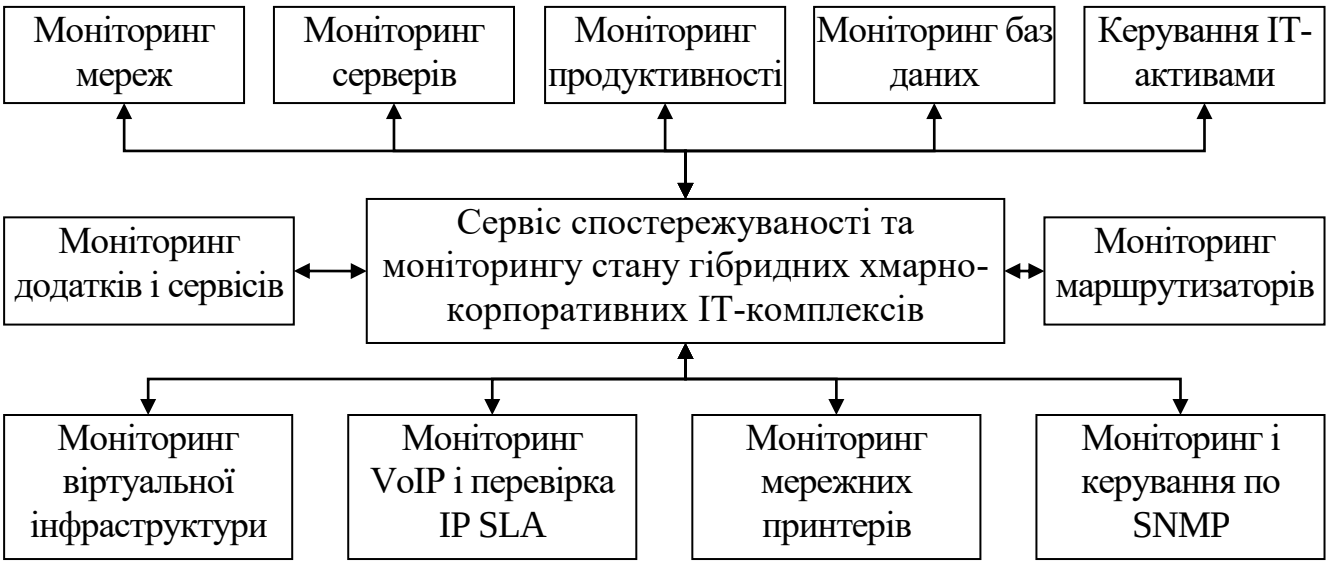


Рисунок 3.2 – Функціональна схема системи

Основні функціональні модулі та напрями моніторингу

1. Моніторинг мереж та мережного обладнання:

– Комплексне сканування: Автоматичне опитування серверів, робочих станцій, комутаторів, маршрутизаторів та периферійних пристроїв без необхідності встановлення додаткових агентів на кінцеві вузли (безагентний моніторинг за допомогою SNMP, WMI, Telnet/SSH).

– Контроль трафіку та пропускної здатності: Збір метрик у реальному часі, підтримка протоколів NetFlow, sFlow і IPFIX для виявлення аномалій та джерел підвищеного навантаження.

– Підтримка обладнання провідних вендорів: Інтеграція з пристроями Cisco, 3Com, Juniper, Alcatel тощо, включно зі спеціалізованими рішеннями на базі нестандартних MIB-описів (збір даних про навантаження процесорів, температуру та статус резервного живлення).

– Аналіз якості VoIP-з'єднань: Контроль параметрів QoS за технологією Cisco IP SLA (вимірювання затримок, втрат пакетів, джиттера та обчислення інтегрального показника якості MOS).

2. Моніторинг серверів та віртуальної інфраструктури:

– Кросплатформеність: Підтримка операційних систем Windows, Linux/Unix, macOS, Solaris, AIX та HP-UX.

– Віртуалізація (VMware ESX/ESXi): Однаково глибокий моніторинг фізичних і віртуальних серверів, контроль стану гостьових ОС, віртуальних машин (ВМ) та їхніх ресурсів (процесор, пам'ять, дискові підсистеми, мережевий ввід-вивід).

– Контроль апаратних ресурсів: Моніторинг завантаження CPU (включно з окремими ядрами), використання оперативної пам'яті, дискового простору (з прогнозуванням вичерпання ємності за допомогою лінійної регресії) та стану систем охолодження.

3. Моніторинг прикладних сервісів і баз даних:

– Додатки та вебсервери (Apache, IIS): Перевірка доступності, виконання процедур авторизації, аналіз часу відгуку, перевірка вмісту сторінок за ключовими словами та моніторинг TCP/UDP портів.

– Поштові системи: Комплексна перевірка протоколів POP3, IMAP, SMTP та наскрізне тестування надсилання й прийому тестових повідомлень.

– Промислові бази даних (Oracle, MS SQL, MySQL, PostgreSQL, Firebird): Виконання динамічних SQL-запитів через драйвери JDBC/ODBC, перевірка структури даних, діагностика збоїв і контроль часу виконання транзакцій.

4. Інтелектуальна обробка подій, тривоги та звітність:

– Централізований журнал подій: Збір та агрегація сповіщень через Syslog, SNMP Traps / Informs та системні журнали Windows із можливістю гнучкої фільтрації.

– Підсистема тривог (Alerting): Генерація сповіщень на основі подій або порогових значень метрик. Налаштування багатоканальної доставки (Email, SMS, візуальні/звукові сигналізації) та запуск автоматичних коригувальних дій (виконання скриптів, перезапуск сервісів, команди Wake-on-LAN).

– Автоматизоване звітування: Побудова звітів за розкладом у форматах PDF, HTML, Excel тощо, а також наявність інтерактивних панелей моніторингу (у тому числі віджетів «Топ 10» за показниками продуктивності).

Керування IT-активами (ITAM)

Система автоматизує процеси інвентаризації та супроводу елементів інфраструктури:

– облік апаратної конфігурації робочих станцій, серверів та мережного обладнання;

– моніторинг стану витратних матеріалів (наприклад, рівня тонера та зносу компонентів мережних принтерів) із можливістю автоматичного формування заявок на їх заміну.

3.4 Розробка діаграми процесів

На рисунку 3.3 наведено діаграму потоків даних розробленої системи, детальний розгляд якої дозволяє простежити логіку взаємодії її функціональних компонентів. У процесі проєктування використовується структурний підхід та графічне представлення інформаційних потоків у системі (Data Flow Diagram, DFD), що дає змогу ефективно візуалізувати процеси збору, обробки та передачі даних у межах комплексу спостережуваності та моніторингу гібридних хмарно-корпоративних IT-комплексів.

Згідно з інженерною методологією, розробка розпочинається з побудови контекстної діаграми (діаграми нульового рівня), яка визначає глобальні межі системи та її інтерфейси із зовнішнім середовищем. Надалі ця модель підлягає ітераційній декомпозиції шляхом деталізації базових процесів та потоків телеметричних даних з метою повного відображення архітектури розроблюваного програмно-апаратного рішення.

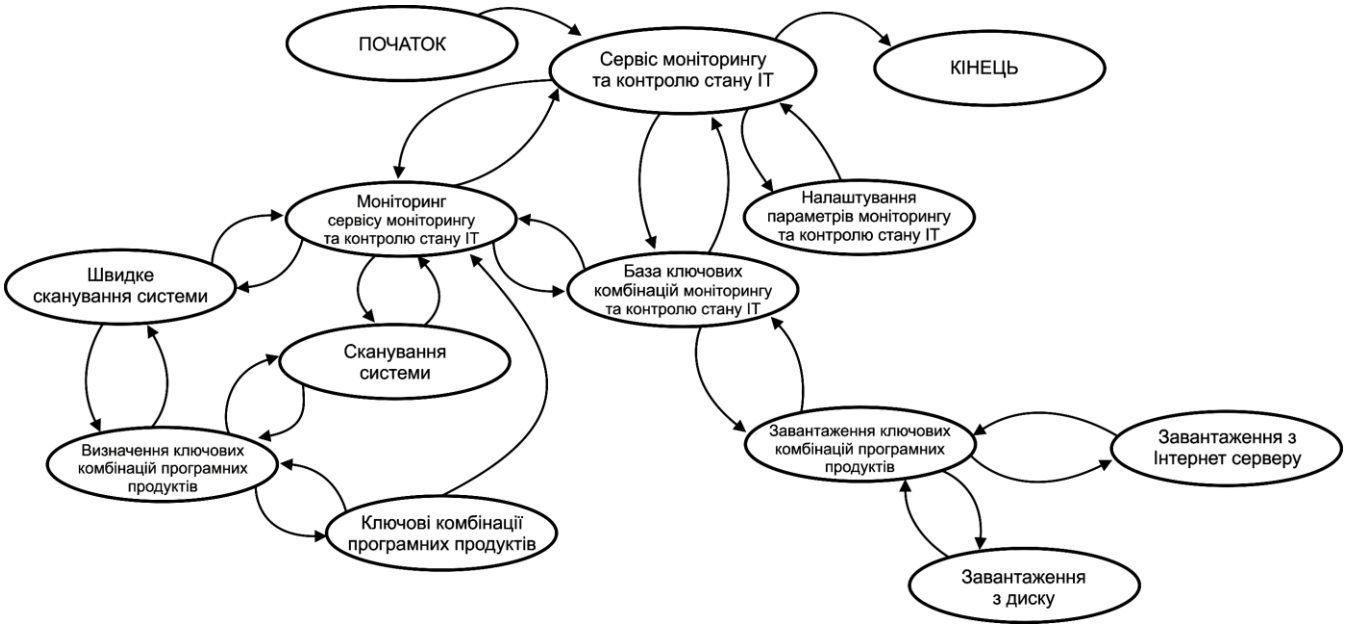


Рисунок 3.3 – Діаграма взаємодії процесів

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Функціонування системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних IT-комплексів ґрунтується на безперервному отриманні телеметричних даних, аналізі характеристик обчислювальних ресурсів і встановленні взаємозв'язків між подіями різних компонентів інфраструктури. Основним призначенням алгоритмічного забезпечення є своєчасне виявлення відхилень у роботі серверів, віртуальних машин, контейнеризованих застосунків, мережевого обладнання та хмарних сервісів. Для цього необхідно організувати узгоджене збирання метрик, журналів подій і розподілених трасувань. Отримані дані використовуються для оцінювання поточного стану компонентів, визначення можливих причин деградації та формування повідомлень про порушення працездатності. Загальну блок-схему функціонування програмного комплексу представлено на рисунку 4.1.

Особливістю гібридної інфраструктури є одночасне використання локальних обчислювальних ресурсів і сервісів публічних або приватних хмар. Такі компоненти можуть працювати під керуванням різних операційних систем, платформ віртуалізації та засобів адміністрування. Частина ресурсів контролюється безпосередньо організацією, тоді як інша доступна через програмні інтерфейси хмарних провайдерів. Це ускладнює створення єдиного механізму моніторингу. Тому запропонована архітектура передбачає використання проміжного рівня телеметричних агентів і колекторів, які забезпечують уніфікацію даних незалежно від місця розміщення обчислювальних компонентів.

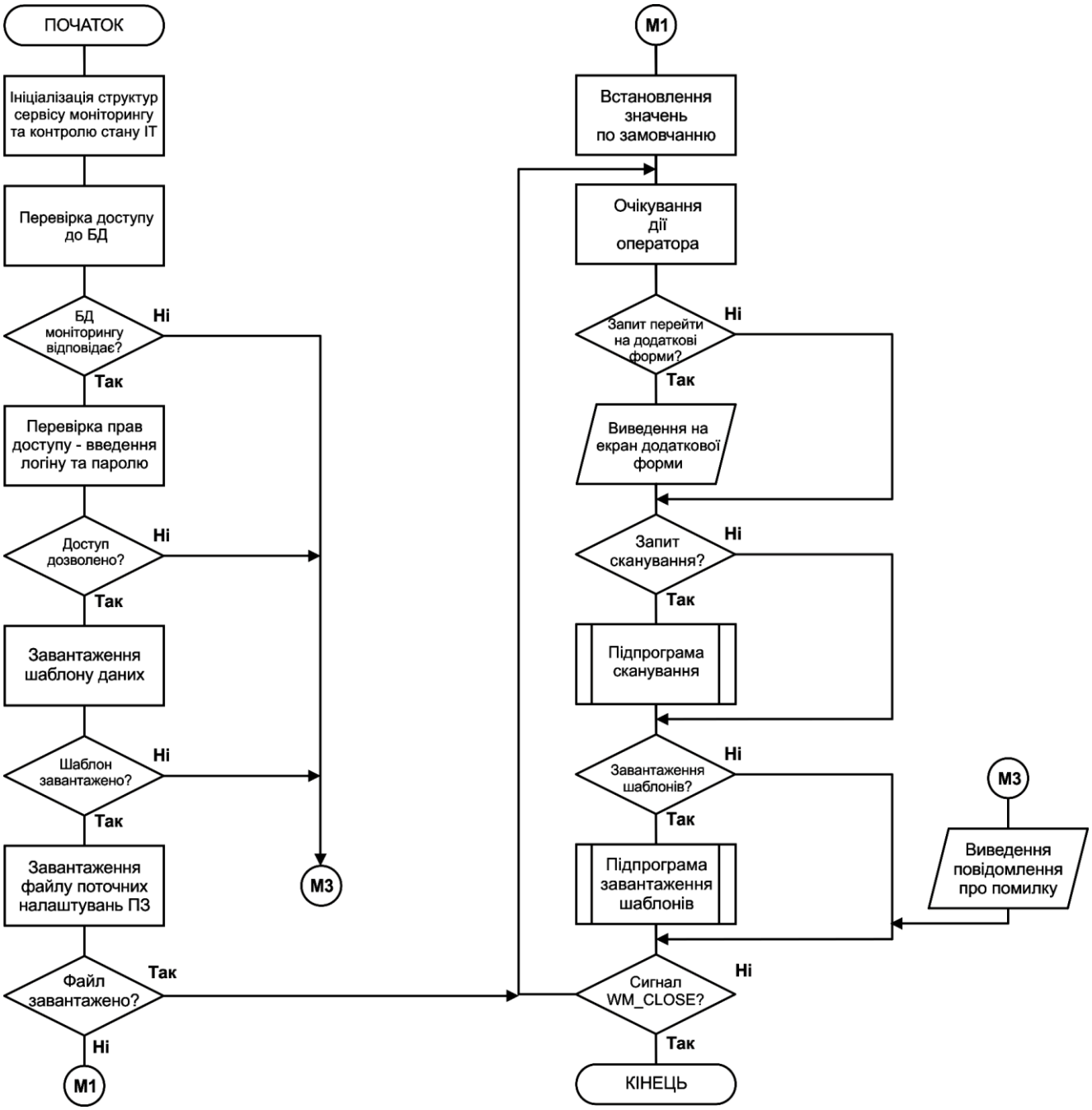


Рисунок 4.1 – Блок-схема основної програми

Основу спостережуваності становлять три взаємодоповнювальні категорії телеметричних даних. Метрики характеризують кількісні показники функціонування системи за визначені часові інтервали.

Журнали подій містять інформацію про окремі операції, помилки та зміни стану програмних компонентів. Розподілені трасування відображають послідовність проходження запиту через взаємопов'язані сервіси. Спільне

використання цих даних дозволяє перейти від простого контролю доступності до аналізу причин порушення роботи складного інформаційного комплексу.

Для забезпечення наскрізного контролю необхідно пов'язати телеметричні повідомлення з відповідними обчислювальними ресурсами. Кожному серверу, віртуальній машині, контейнеру або прикладному сервісу ставиться у відповідність унікальний ідентифікатор. Він використовується під час зберігання метрик, пошуку журналів і зіставлення розподілених трасувань. Додатково враховуються належність ресурсу до локального або хмарного сегмента, його функціональне призначення та зв'язки з іншими компонентами. Алгоритм оброблення телеметрії повинен підтримувати актуальність цих характеристик за умов динамічного створення й видалення віртуалізованих ресурсів.

Центральним елементом аналітичної підсистеми є модуль оцінювання стану інформаційних сервісів. Він отримує підготовлені показники, перевіряє їхню актуальність і визначає наявність відхилень від допустимих значень. Результати окремих перевірок зіставляються з даними про залежності між компонентами.

Це дозволяє формувати узагальнену оцінку працездатності сервісу та встановлювати можливі джерела деградації. Алгоритмічну послідовність функціонування відповідної підпрограми аналізу стану відображено на рисунку 4.2.

Для програмної реалізації розглядається архітектура, яка використовує телеметричні колектори, централізовані сховища даних і серверний модуль аналітичного оброблення. Як можливу технологічну основу доцільно застосувати OpenTelemetry для уніфікованого отримання телеметрії, Prometheus для роботи з часовими рядами та Grafana для візуалізації стану інфраструктури. Для зберігання журналів і трасувань можуть використовуватися спеціалізовані компоненти, зокрема Loki та Tempo.

Програмну логіку додаткового аналізу можна реалізувати мовою Python. Зазначений набір технологій характеризує рекомендовану конфігурацію, а не підтверджений склад фактично реалізованого програмного забезпечення.

Проєктне рішення щодо організації наскрізної телеметрії гібридного комплексу

На початковому етапі необхідно визначити структуру контрольованої інформаційної інфраструктури. Вхідними даними є перелік фізичних серверів, віртуальних машин, контейнерів, мережевих вузлів, хмарних служб і прикладних сервісів. Для кожного компонента встановлюються доступні способи отримання телеметрії. Додатково визначається необхідна частота вимірювань та рівень деталізації. Результатом є конфігурація джерел даних, яка використовується під час організації безперервного спостереження.

Для локальних серверів доцільно використовувати програмні експортери або агенти, які отримують системні показники. До них належать завантаження процесора, обсяг використаної оперативної пам'яті, характеристики дискових операцій та статистика мережевого обміну. Для віртуалізованих середовищ необхідно додатково контролювати стан гіпервізора, використання спільних ресурсів і доступність віртуальних машин. У контейнеризованих системах важливими є показники використання ресурсів окремими контейнерами, кількість перезапусків і стан розгортання прикладних компонентів.

У хмарному середовищі телеметрія може отримуватися через вбудовані служби моніторингу провайдера та підтримувані програмні інтерфейси. При цьому необхідно враховувати різницю між системними метриками, доступними безпосередньо з віртуальної машини, та показниками керованого сервісу, які надає хмарна платформа. Наприклад, для керованої бази даних можуть бути доступні метрики продуктивності, але відсутній прямий доступ до операційної системи сервера. Тому алгоритм збирання повинен визначати доступний набір вимірювань для кожного типу ресурсу.

Для уніфікації телеметричних потоків може використовуватися OpenTelemetry Collector. Його функціональна модель передбачає компоненти приймання, оброблення та експорту телеметрії. Окремі конвеєри можуть обслуговувати метрики, журнали та трасування. Така організація дозволяє відокремити способи отримання даних від конкретних сховищ і засобів аналізу.

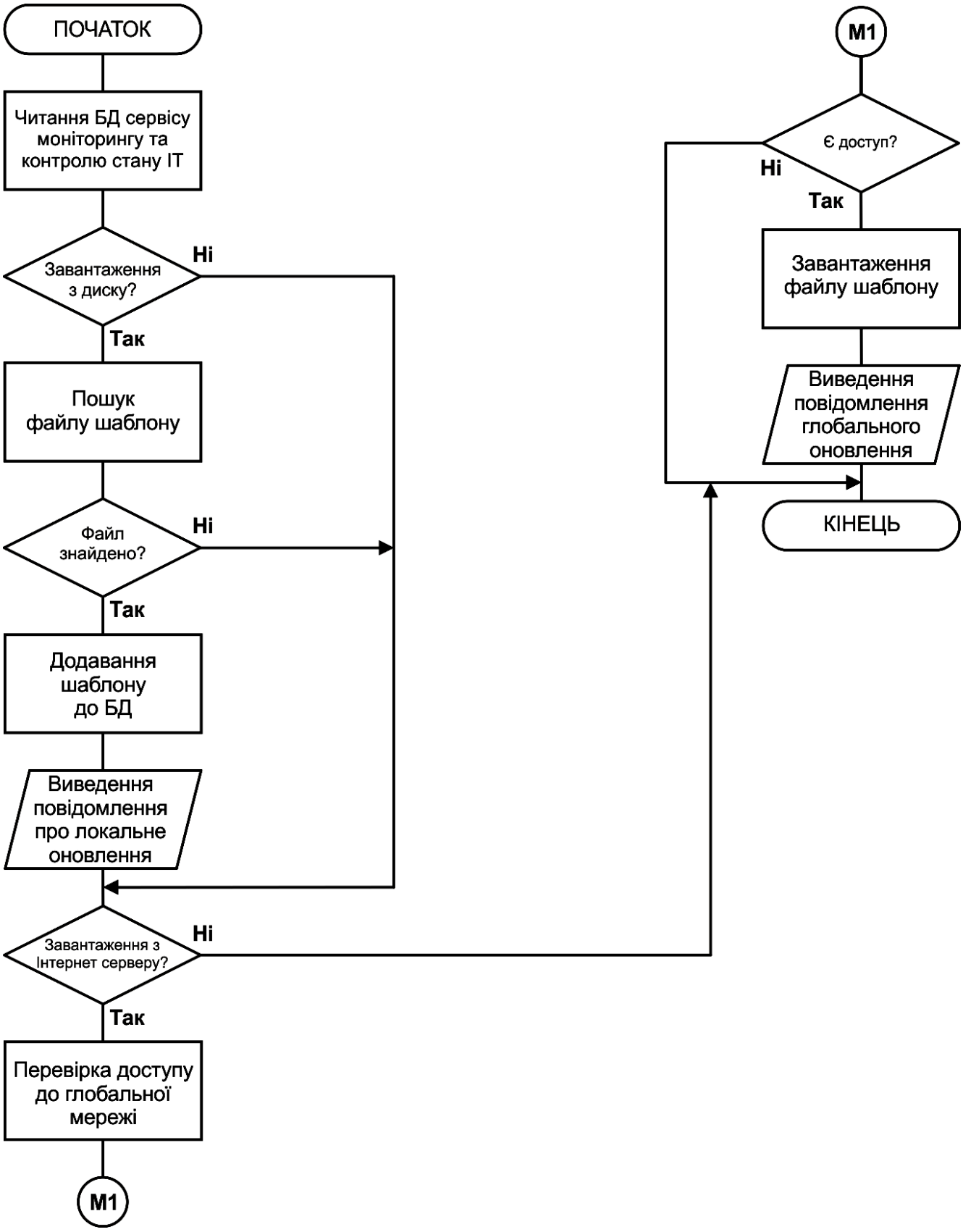


Рисунок 4.2 – Блок-схема роботи підпрограми

У гібридному комплексі доцільно розміщувати колектори безпосередньо в локальних і хмарних сегментах.

Локальний колектор отримує дані від корпоративних серверів, а хмарний обробляє телеметрію відповідних віддалених ресурсів. Після попереднього оброблення інформація передається до центральної аналітичної підсистеми через захищений канал. Такий підхід зменшує потребу в прямому доступі центрального сервера до кожного контрольованого вузла. Також він дозволяє організувати локальне буферування за тимчасової недоступності міжсегментного з'єднання.

Послідовність оброблення метрик передбачає отримання вимірювання, перевірку його структури, визначення ресурсу та приведення одиниць вимірювання до узгодженого формату. Наприклад, обсяг використаної пам'яті доцільно представляти в байтах, а часові характеристики операцій у секундах. Відносні показники повинні мати однозначно визначений діапазон. Це необхідно для правильного порівняння вимірювань, отриманих від різних операційних систем і хмарних платформ.

Для журналів подій додатково виконується визначення рівня важливості, часу виникнення та програмного компонента, який сформував повідомлення. Текст журналу може містити специфічні поля, які не використовуються іншими системами.

Такі атрибути доцільно зберігати в розширеній структурі повідомлення. Водночас необхідно забезпечити наявність основних ідентифікаторів ресурсу та часу події. Це дозволяє виконувати подальшу кореляцію журналів із метриками та трасуваннями.

Для розподілених трасувань використовуються ідентифікатори trace_id та span_id. Вони дозволяють пов'язувати операції одного запиту, навіть якщо його оброблення виконується декількома сервісами на різних серверах. Кожен сегмент трасування містить інформацію про окрему операцію, її тривалість та зв'язок із батьківською операцією. Коректне передавання контексту між сервісами є необхідною умовою відновлення повного шляху запиту.

Уніфікація вимірювань і забезпечення достовірності часових даних

Отримані телеметричні повідомлення можуть відрізнятися частотою формування, часовою точністю та способом представлення показників. Для забезпечення їхньої сумісності необхідно виконувати нормалізацію. Вона передбачає перевірку числових значень, одиниць вимірювання, часових позначок та ідентифікаторів джерел. Повідомлення з некоректними параметрами не повинні передаватися до аналітичного модуля як достовірні вимірювання.

Важливо розрізняти час виникнення події та час її отримання центральним сервером. Для локальних вузлів різниця може бути незначною, тоді як у хмарних сегментах вона залежить від затримок мережевого передавання та роботи проміжних колекторів. Усі часові позначки доцільно приводити до UTC. Для синхронізації годинників контрольованих вузлів може використовуватися NTP або інший придатний механізм. Якщо часові дані неузгоджені, система повинна позначати відповідні вимірювання як потенційно ненадійні.

Модуль нормалізації повинен враховувати різні типи метрик. Лічильники накопичують значення й зазвичай використовуються для оцінювання інтенсивності операцій. Миттєві показники характеризують стан ресурсу в певний момент часу. Гістограми містять інформацію про розподіл вимірювань, наприклад тривалості HTTP-запитів. Некоректне застосування однакових математичних операцій до різних типів метрик може призвести до помилкових висновків.

Для збереження числових вимірювань можна використовувати структуру, яка містить назву метрики, ідентифікатор ресурсу, числове значення, одиницю вимірювання та часову позначку. Наведений фрагмент Python демонструє можливий механізм перевірки та нормалізації таких даних. Код не залежить від конкретного телеметричного колектора та може використовуватися як основа серверного адаптера.

```
from dataclasses import dataclass
from datetime import datetime, timezone
from math import isfinite

# Уніфікований запис числової метрики
@dataclass(frozen=True)
class MetricSample:
    resource_id: str
```

```
    metric_name: str
    value: float
    unit: str
    timestamp: datetime

# Допустимі назви метрик та відповідні одиниці
ALLOWED_METRICS = {
    "cpu_utilization": "ratio",
    "memory_used": "bytes",
    "disk_read_bytes_total": "bytes",
    "request_duration": "seconds"
}
def normalize_metric(payload: dict) -> MetricSample:
    # Перевірка ідентифікатора джерела
    resource_id = payload.get("resource_id")

    if not isinstance(resource_id, str) or not resource_id:
        raise ValueError("Не визначено ресурс")

    # Перевірка назви та одиниці метрики
    name = payload.get("metric_name")
    expected_unit = ALLOWED_METRICS.get(name)

    if expected_unit is None:
        raise ValueError("Непідтримувана метрика")
    unit = payload.get("unit")
    if unit != expected_unit:
        raise ValueError("Невідповідність одиниць")

    # Перевірка числового значення
    raw_value = payload.get("value")

    if type(raw_value) not in (int, float):
        raise ValueError("Значення не є числовим")
    value = float(raw_value)
    if not isfinite(value):
        raise ValueError("Некоректне числове значення")
    if name == "cpu_utilization" and not 0 <= value <= 1:
        raise ValueError("Завантаження поза діапазоном")

    # Часова позначка має містити часову зону
    raw_time = payload.get("timestamp")
    if not isinstance(raw_time, str):
        raise ValueError("Відсутній час вимірювання")
    timestamp = datetime.fromisoformat(
        raw_time.replace("Z", "+00:00")
    )
    if timestamp.tzinfo is None:
        raise ValueError("Не визначено часову зону")
    return MetricSample(
        resource_id=resource_id,
        metric_name=name,
        value=value,
        unit=unit,
        timestamp=timestamp.astimezone(timezone.utc)
    )
```

Функція забезпечує перевірку базових параметрів вимірювання та відхиляє некоректні значення. У прикладі використано обмежений перелік метрик для демонстрації принципу нормалізації. Для реального комплексу необхідно передбачити конфігурований реєстр показників і перевірку допустимих значень

відповідно до їхнього типу. Окремо повинні контролюватися від'ємні значення накопичувальних лічильників, несумісні одиниці та часові позначки, що виходять за встановлений інтервал.

Після нормалізації система перевіряє актуальність вимірювання. Якщо показник надходить із затримкою, він може бути збережений для історичного аналізу, але не повинен безумовно використовуватися для визначення поточного стану ресурсу. Для кожної групи метрик доцільно встановлювати максимальний допустимий вік даних. Якщо цей інтервал перевищено, результат поточного моніторингу позначається як невизначений.

Особливу увагу необхідно приділити повторному надходженню телеметрії. За відновлення мережевого з'єднання колектор може повторно передати раніше накопичені повідомлення. Для запобігання дублюванню необхідно використовувати ідентифікатор часового ряду, часову позначку та додаткові характеристики джерела. Для журналів можуть застосовуватися унікальні ідентифікатори записів. Повторне отримання не повинно безпідставно збільшувати кількість зафіксованих подій.

Математична модель оцінювання технічного стану сервісів

Після підготовки телеметричних даних виконується оцінювання стану контрольованих ресурсів. Для цього необхідно визначити набір показників, які характеризують працездатність конкретного сервісу. Для обчислювального вузла такими показниками можуть бути завантаження процесора, використання пам'яті та характеристики дискових операцій. Для прикладного сервісу важливішими є частка помилкових запитів, тривалість оброблення операцій і доступність основних функцій. Використання однакових критеріїв для всіх компонентів не завжди дозволяє правильно визначити їхній стан.

Для базового моніторингу доцільно використовувати порогову модель. Вона передбачає порівняння поточного значення показника з допустимою межею. Якщо значення виходить за встановлений діапазон, формується ознака відхилення. Проте одноразове перевищення порогу не завжди характеризує реальну несправність. Короткочасне підвищення навантаження може бути

нормальним під час виконання ресурсоємної операції. Тому алгоритм повинен враховувати тривалість відхилення та його повторюваність.

Для оцінювання стабільності показника можна використовувати ковзне середнє, визначене формулою $\bar{x_t} = (1)/(N)\sum(i=0)^{(N-1)} x_{(t-i)}$, де N є кількістю вимірювань у вікні, а $x_{(t-i)}$ характеризує відповідне значення показника. Отримана величина дозволяє зменшити вплив короткочасних коливань. Водночас використання великого вікна збільшує затримку виявлення різких змін. Тому параметр N повинен визначатися з урахуванням необхідної оперативності моніторингу.

Для виявлення нетипової поведінки доцільно додатково застосувати статистичний аналіз історичних вимірювань. Наприклад, для показників із відносно стабільним розподілом може використовуватися стандартизоване відхилення $z_t = (x_t - \mu)/\sigma$, де μ та σ є оцінками середнього значення й стандартного відхилення за базовий період. Значне абсолютне значення z_t свідчить про нетиповість поточного вимірювання. Однак для показників із сезонністю, асиметричним розподілом або частими різкими змінами така модель може формувати надмірну кількість помилкових спрацювань.

Для підвищення стійкості до поодиноких викидів можна використовувати медіану та медіанне абсолютне відхилення MAD. Вони менш чутливі до окремих екстремальних значень. За достатньої кількості історичних вимірювань формується базовий інтервал нормальної поведінки. Поточне значення порівнюється з цим інтервалом. Якщо відхилення перевищує задану межу, система формує повідомлення про можливу аномалію.

Програмну реалізацію статистичного аналізу можна виконати мовою Python за допомогою стандартного модуля statistics. Наступний фрагмент демонструє перевірку поточного значення за історичними даними. Історичне вікно не містить самого контрольованого вимірювання, що дозволяє уникнути його безпосереднього впливу на оцінку базового рівня.

```
from statistics import median
from math import isfinite
```

```
def detect_metric_anomaly(
    history,
    current_value,
    threshold=3.5,
    min_samples=12,
    min_scale=0.01
):
    # Перевірка достатності історичних даних
    if len(history) < min_samples:
        return {
            "status": "insufficient_data",
            "score": None
        }
    values = [float(value) for value in history]
    current = float(current_value)
    if not all(isfinite(value) for value in values):
        raise ValueError("Історія містить некоректні значення")
    if not isfinite(current):
        raise ValueError("Некоректне поточне значення")
    if threshold <= 0 or min_scale <= 0:
        raise ValueError("Некоректні параметри аналізу")
    # Визначення медіанного рівня метрики
    baseline = median(values)
    # Обчислення медіанного абсолютного відхилення
    deviations = [
        abs(value - baseline) for value in values
    ]
    mad = median(deviations)
    # Масштаб визначається за каліброваною моделлю
    scale = max(1.4826 * mad, min_scale)
    # Оцінювання відхилення поточного значення
    score = abs(current - baseline) / scale
    return {
        "status": (
            "anomaly" if score >= threshold else "normal"
        ),
        "score": score,
        "baseline": baseline,
        "scale": scale
           }
```

Наведений алгоритм визначає ступінь відхилення поточного значення від історичного медіанного рівня. Коефіцієнт 1,4826 використовується для наближеного приведення MAD до масштабу стандартного відхилення за умови нормального розподілу. Параметр min_scale запобігає діленню на нуль і повинен визначатися відповідно до фізичного змісту конкретної метрики. Для показників із різними одиницями вимірювання необхідно використовувати окремі конфігурації аналізу.

У гібридних IT-комплексах важливо враховувати періодичні зміни навантаження. Наприклад, у робочий час інтенсивність звернень до корпоративної інформаційної системи може бути значно більшою, ніж уночі. Тому порівняння поточного значення з єдиним середнім рівнем за тривалий

період може спричиняти помилкові спрацювання. Для таких показників доцільно формувати окремі базові профілі відповідно до часу доби, дня тижня або характеру технологічного процесу.

Результатом аналітичного оброблення є структурований запис про стан метрики. Він містить ідентифікатор ресурсу, назву показника, час оцінювання, отримане значення, базовий рівень і статус відхилення. Якщо для прийняття рішення недостатньо історичних даних, система повинна повертати окремий статус. Відсутність результату статистичного аналізу не повинна автоматично вважатися підтвердженням нормальної роботи ресурсу.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програмне забезпечення системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів призначене для централізованого контролю обчислювальних ресурсів, мережевих компонентів і прикладних сервісів, розміщених у локальній та хмарній інфраструктурі. Система забезпечує збирання телеметричних даних, аналіз показників продуктивності, виявлення відхилень та формування сповіщень про порушення працездатності. Для оцінювання стану ресурсів передбачається спільне використання метрик, журналів подій і розподілених трасувань. Головне вікно програми показано на рисунку 5.1.

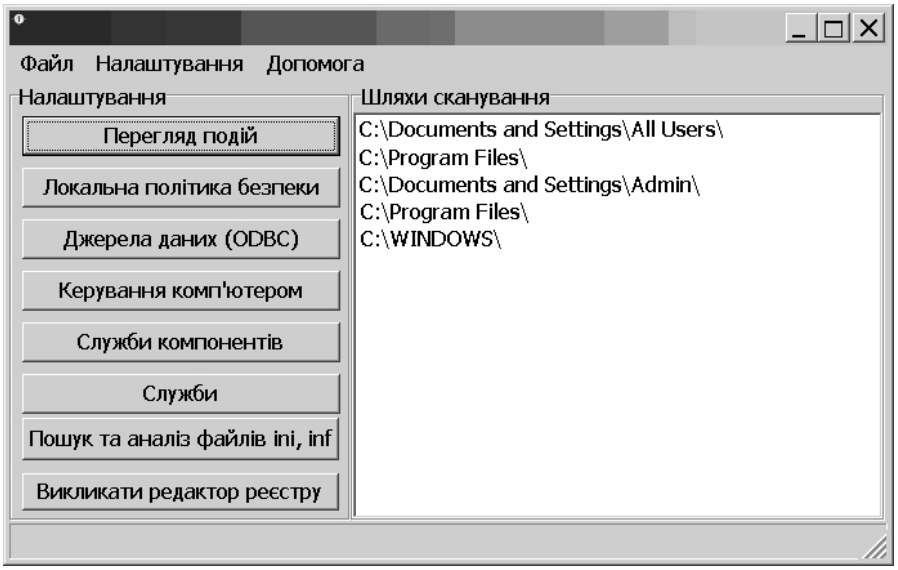


Рисунок 5.1 – Головне вікно ПЗ

Для впровадження програмного комплексу необхідно підготувати серверне середовище, яке забезпечує приймання, оброблення та зберігання телеметричних даних. Як основу розгортання доцільно використовувати серверну операційну систему Linux. Компоненти системи можуть розміщуватися на окремих

віртуальних машинах або в контейнеризованому середовищі. Розподілення сервісів між кількома вузлами дозволяє незалежно виділяти обчислювальні ресурси для збирання телеметрії, аналітичного оброблення та зберігання інформації. Конкретна конфігурація визначається кількістю контрольованих ресурсів та інтенсивністю надходження повідомлень.

Для реалізації телеметричного конвеєра пропонується використання OpenTelemetry Collector. Цей компонент забезпечує отримання даних із різних джерел, їх попереднє оброблення та передавання до відповідних систем зберігання. Збирання інфраструктурних метрик може здійснюватися через спеціалізовані експортери. Для накопичення часових рядів доцільно застосувати Prometheus. Журнали подій можуть зберігатися в Loki, а розподілені трасування в Tempo. Візуалізацію отриманих показників передбачається організувати засобами Grafana. Зазначені компоненти утворюють рекомендовану технологічну конфігурацію програмного комплексу.

Апаратні вимоги залежать від кількості джерел телеметрії, частоти опитування та строку зберігання історичних даних. Найбільше навантаження на дискову підсистему можуть створювати журнали подій і деталізовані часові ряди. Для серверів зберігання необхідно передбачити достатню продуктивність дискових операцій та можливість збільшення доступного простору. Обсяг оперативної пам'яті повинен забезпечувати оброблення поточних запитів і виконання аналітичних операцій. Для міжсегментного обміну необхідне стабільне мережеве підключення з підтримкою захищеного передавання даних.

Підготовка інфраструктури розпочинається з визначення контрольованих локальних і хмарних ресурсів. Для кожного сервера, віртуальної машини, контейнера або прикладного сервісу встановлюється спосіб отримання телеметрії. Додатково визначаються мережеві адреси, доступні програмні інтерфейси та належність до відповідного сегмента. Для забезпечення однозначної ідентифікації використовуються атрибути ресурсу, зокрема назва сервісу, ідентифікатор вузла та характеристики середовища розгортання. Ці відомості необхідні для подальшого зіставлення метрик, журналів і трасувань.

Наступним етапом є розгортання телеметричних колекторів. У корпоративному сегменті доцільно встановити окремий екземпляр OpenTelemetry Collector, який отримує дані від локальних серверів. Для хмарних ресурсів може використовуватися додатковий колектор, розміщений поблизу контрольованих сервісів. Під час налаштування визначаються джерела даних, процедури оброблення та кінцеві точки експорту. Передавання інформації до центрального середовища повинно здійснюватися через захищені з'єднання. Для тимчасової недоступності мережі доцільно передбачити буферизацію з урахуванням доступного обсягу пам'яті або дискового простору.

Для серверної частини аналітичного оброблення передбачається використання Python. Модулі цієї мови можуть виконувати нормалізацію вимірювань, статистичний аналіз показників, оцінювання стану ресурсів і кореляцію сповіщень. Перед встановленням необхідно підготувати сумісне середовище виконання та програмні залежності. Для ізоляції бібліотек доцільно створити окреме віртуальне середовище Python. Після встановлення компонентів перевіряється доступність модулів оброблення даних та можливість отримання показників із відповідних сховищ.

Підготовка Prometheus передбачає визначення джерел метрик, інтервалів опитування та строку зберігання часових рядів. Для кожного джерела налаштовується відповідна кінцева точка отримання даних. Необхідно також визначити допустимі мітки метрик, щоб запобігти надмірному збільшенню кількості часових рядів. Для Loki та Tempo налаштовуються приймання відповідних типів телеметрії, параметри сховищ і правила збереження даних. Способи передавання між OpenTelemetry Collector та цими компонентами повинні відповідати підтримуваним інтерфейсам.

Після розгортання сховищ виконується налаштування Grafana. До системи додаються відповідні джерела даних для отримання метрик, журналів і трасувань. Для контрольованих ресурсів формуються інформаційні панелі, які дозволяють переглядати завантаження процесорів, використання пам'яті, характеристики дискових операцій та мережевого обміну. Для прикладних сервісів доцільно

відображати інтенсивність запитів, частку помилок і показники затримки. Коректність налаштування перевіряється шляхом зіставлення відображених значень із даними відповідних джерел.

Окремої підготовки потребує модуль аналізу аномалій. Для нього визначаються контрольовані метрики, тривалість історичного вікна та допустимі відхилення. Параметри статистичної обробки повинні враховувати особливості кожного показника. Для ресурсів із періодичним навантаженням доцільно використовувати окремі базові профілі. Перед увімкненням автоматичного формування сповіщень необхідно перевірити алгоритм на контрольованих вимірюваннях, які містять нормальні значення та штучно сформовані відхилення.

Налаштування системи сповіщень передбачає визначення порогових значень, періодів підтвердження порушення та умов повернення до нормального стану. Для кожного правила встановлюється належність до відповідного ресурсу або прикладного сервісу. Окремо визначається поведінка за відсутності телеметрії та виникнення помилок її отримання. Такі ситуації не повинні автоматично трактуватися як підтвердження нормальної роботи. Для зменшення кількості повторних повідомлень доцільно налаштувати групування сповіщень за спільними ресурсами та залежностями.

Рекомендована послідовність запуску починається з перевірки мережевої доступності й готовності серверів зберігання. Після цього запускаються Prometheus, Loki та Tempo з урахуванням обраної конфігурації. Наступним етапом активуються локальні та хмарні колектори OpenTelemetry. Після підтвердження надходження телеметрії запускається аналітичний модуль Python. Завершальним етапом активуються Grafana та правила сповіщень. Перед переведенням системи до робочого режиму необхідно перевірити доступність усіх джерел даних і правильність їхньої ідентифікації.

Первинна перевірка працездатності виконується шляхом отримання контрольних вимірювань від локального сервера та віддаленого хмарного ресурсу. Необхідно підтвердити появу метрик у Prometheus, надходження журналів до Loki та доступність трасувань у Tempo за наявності відповідної

інструментації застосунків. Після цього перевіряється правильність відображення даних у Grafana. Для оцінювання роботи аналітичного модуля формується контрольоване відхилення одного з показників. Система повинна визначити його відповідно до заданих критеріїв та сформувати належне сповіщення.

Для ідентифікації програмного забезпечення необхідно передбачити відображення інформації про розробника та належність програмного продукту до відповідної магістерської кваліфікаційної роботи. Авторські відомості доцільно розміщувати в окремому інформаційному розділі програми. Вони повинні бути відокремлені від службової конфігурації та не містити конфіденційних параметрів моніторингової інфраструктури. Дані про авторство розробленого програмного забезпечення представлено на рисунку 5.2.

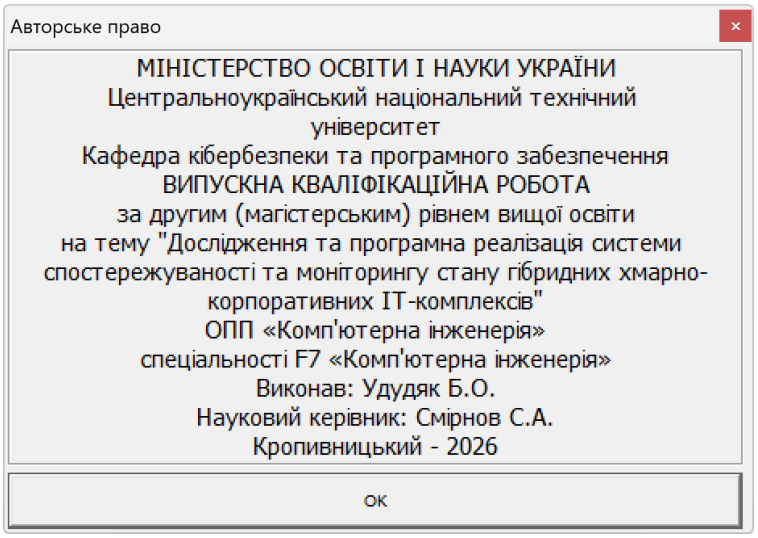


Рисунок 5.2 – Шаблона форма авторського права

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

Метою розробки є дослідження та програмна реалізація системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

Об'єктом дослідження є процес спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

Предметом дослідження є методи спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

Методи дослідження базуються на методах операційних систем, методах теорії комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

– Розроблено вітчизняний продукт спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

7. Вступ

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

Правовою основою організації робіт з охорони праці є Конституція України, Закон України «Про охорону праці», Кодекс законів про працю України (КЗпП) та державні нормативні акти з охорони праці.

Метою даного розділу магістерської роботи є аналіз умов праці інженера-програміста під час розробки веб-сервісу для аналізу даних, виявлення потенційно небезпечних та шкідливих виробничих факторів, а також розробка комплексних заходів щодо їх усунення або мінімізації до нормативних значень.

Актуальність питань охорони праці для працівників ІТ-сфери обумовлена специфікою їхньої діяльності: тривалою роботою з відеодисплейними терміналами (ВДТ), високим нервово-емоційним напруженням та гіподинамією. Створення безпечних умов праці є запорукою високої продуктивності розробника та запобігання професійним захворюванням.

7.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Об'єктом дослідження є робоче місце розробника програмного забезпечення, обладнане персональним комп'ютером. Робота виконується у приміщенні кафедри.

Характеристика приміщення

Приміщення розташоване на другому поверсі будівлі, має прямокутну форму та бічне природне освітлення через вікна. Геометричні параметри приміщення:

– Довжина (А): 5,0 м.

– Ширина (В): 4,0 м.

– Висота (Н): 3,0 м.

Розрахуємо площу (S) та об'єм (V) приміщення:

$$S = A \cdot B = 5{,}0 \cdot 4{,}0 = 20{,}0 \text{ м}^2$$

$$V = S \cdot H = 20{,}0 \cdot 3{,}0 = 60{,}0 \text{ м}^3$$

У приміщенні працює 1 співробітник.

Згідно з ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», на одне робоче місце з ВДТ повинно припадати не менше 6,0 м² площі та не менше 20,0 м³ об'єму.

Фактичні значення $20{,}0 \text{ м}^2$ та $60{,}0 \text{ м}^3$ значно перевищують нормативні, що свідчить про дотримання санітарних вимог щодо простору.

Аналіз небезпечних та шкідливих виробничих факторів

Відповідно до ГОСТ 12.0.003-74, на розробника під час роботи можуть впливати фактори, наведені в таблиці 7.1.

Робота програміста відноситься до категорії 1а (легка фізична робота), оскільки енерговитрати становлять до 139 Вт, а робота виконується сидячи і не потребує систематичного фізичного напруження. Параметри мікроклімату повинні відповідати вимогам ДСН 3.3.6.042-99. Оптимальні та допустимі норми наведено в таблиці 7.2.

Таблиця 7.1 – Характеристика шкідливих та небезпечних факторів

Група факторів	Найменування фактору	Джерело виникнення	Нормативний документ
Фізичні	Підвищена температура повітря робочої зони	Нагрів електронних компонентів ПК, система опалення	ДСН 3.3.6.042-99
	Недостатня освітленість робочої зони	Невідповідність потужності ламп, забруднення вікон	ДБН В.2.5-28-2018
	Підвищений рівень шуму	Системи охолодження системного блоку	ДСН 3.3.6.037-99
	Підвищений рівень електромагнітних випромінювань	Монітор, системний блок, периферія	ДСанПіН 3.3.2.007-98
	Небезпека ураження електричним струмом	Електрична мережа 220В, 50Гц	ПУЕ, НПАОП 40.1-1.21-98
Психофізіологічні	Нервово-емоційне перенапруження	Розв'язання складних алгоритмічних задач, дефіцит часу	ДСанПіН 3.3.2.007-98
	Перенапруження зорового аналізатора	Тривала робота з об'єктами розрізнення на екрані	-//-
	Монотонність праці та гіподинамія	Тривале перебування у фіксованій позі (сидячи)	-//-

Таблиця 7.2 – Норми мікроклімату для категорії робіт 1а

Період року	Параметр	Оптимальні величини	Допустимі величини
Холодний	Температура повітря, °С	22 – 24	21 – 25
	Відносна вологість, %	40 – 60	не більше 75
	Швидкість руху повітря, м/с	0,1	не більше 0,1
Теплий	Температура повітря, °С	23 – 25	22 – 28
	Відносна вологість, %	40 – 60	55 (при 28°С)
	Швидкість руху повітря, м/с	0,1	0,1 – 0,2

Для підтримання цих параметрів приміщення обладнане системою централізованого водяного опалення (взимку) та побутовим кондиціонером (влітку). Вентиляція – природна, через вікна та кватирки.

Аналіз виробничого освітлення

Робота з комп'ютером вимагає високої концентрації зору і належить до розряду зорових робіт III (високої точності). Згідно з ДБН В.2.5-28-2018 «Природне і штучне освітлення», у приміщенні передбачено суміщене освітлення:

1. Природне: бічне, здійснюється через два вікна розміром 1,5 · 1,8м. Коефіцієнт природної освітленості (КПО) становить е = 1,5%.

2. Штучне: загальне рівномірне освітлення забезпечується стельовими світильниками з LED-лампами.

Нормована освітленість робочої поверхні столу повинна становити не менше 300–500 лк. Також важливим параметром є обмеження прямої

блискучості джерел світла та відбитої блискучості на екрані монітора. Коефіцієнт пульсації освітленості не повинен перевищувати 5%.

Аналіз шуму та вібрації

Джерелами шуму в приміщенні є вентилятори охолодження комп'ютерної техніки (кулери процесора, відеокарти, блоку живлення) та зовнішній шум з вулиці.

Відповідно до ДСН 3.3.6.037-99, гранично допустимий рівень звуку на робочих місцях програмістів становить 50 дБА.

Фактичний рівень шуму від сучасного ПК складає близько 35-40 дБА, що не перевищує нормативних значень. Вібрація на робочому місці відсутня.

7.3 Розробка заходів з поліпшення стану охорони праці

Основним напрямком поліпшення умов праці розробника є раціональна організація робочого місця відповідно до вимог ергономіки та фізіології праці. Неправильне розміщення техніки або незручні меблі можуть призвести до швидкої втомлюваності та зниження продуктивності.

Ергономічні вимоги до організації робочого місця Робоче місце організовано за принципом функціонального зонування. Згідно з ДСанПіН 3.3.2.007-98, конструкція робочого столу та крісла повинна забезпечувати підтримання раціональної робочої пози (кут нахилу корпусу – 15-20° вперед або назад, кут згинання ліктів – 90°).

1. Робочий стіл. Використовується стіл модульної конструкції. Висота робочої поверхні становить 725 мм. Поверхня столу має матове покриття світло-сірого кольору з коефіцієнтом відбиття 0,4, що запобігає утворенню відблисків. Простір для ніг має розміри: висота – 600 мм, ширина – 500 мм, глибина – 450 мм.

2. Робоче крісло. Використовується підйомно-поворотне крісло, яке регулюється за висотою та кутом нахилу спинки.

– Висота сидіння регулюється в межах 400–550 мм.

– Ширина і глибина сидіння становлять не менше 400 мм.

– Спинка має анатомічну форму, що підтримує поперековий відділ хребта.

3. Розміщення засобів відображення інформації. Екран відеомонітора розташовано на оптимальній відстані 600-700 мм від очей користувача (але не ближче 500 мм). Верхній край екрану знаходиться на рівні очей або трохи нижче, що забезпечує кут огляду 15-20° вниз. Це дозволяє розвантажити м'язи шиї та зменшити висихання рогівки ока (зменшується площа відкритої поверхні ока).

4. Клавіатура та маніпулятор «миша». Клавіатура розміщена на поверхні столу на відстані 100–300 мм від краю, зверненого до користувача. Це створює опору для рук. Використовується килимок для миші з гелевою подушечкою під зап'ястя для профілактики тунельного синдрому.

Режим праці та відпочинку

Для запобігання перевтомі розроблено раціональний режим праці. При 8-годинному робочому дні передбачені:

– Обідня перерва – 45 хвилин.

– Регламентовані перерви тривалістю 15 хвилин через кожні 2 години роботи. Під час перерв рекомендується виконувати комплекси вправ для очей (офтальмотренаж) та фізичні вправи для зняття напруги з м'язів спини та шиї.

7.4 Техніка безпеки та протипожежна профілактика

Електробезпека Обладнання робочого місця (ПК, монітор, периферія) живиться від однофазної мережі змінного струму напругою 220 В (50 Гц). Згідно з ПУЕ (Правила улаштування електроустановок) та НПАОП 40.1-1.21-98, приміщення кафедри за ступенем небезпеки ураження електричним струмом відноситься до класу приміщень без підвищеної небезпеки (сухе приміщення, струмонепровідна підлога, відсутність агресивного середовища).

Для забезпечення електробезпеки реалізовано такі заходи:

1. Захисне заземлення. Усі металеві неструмопровідні частини обладнання (корпус системного блоку) заземлені через третій провідник (PE) в кабелі живлення та євророзетку із заземлюючим контактом. Опір заземлюючого пристрою не перевищує 4 Ом.

2. Ізоляція. Використовуються кабелі з подвійною ізоляцією. Проводиться регулярний візуальний огляд на предмет пошкоджень.

3. Захист від перевантажень. Лінія живлення обладнана автоматичними вимикачами, які розмикають ланцюг при виникненні струмів короткого замикання.

4. Організаційні заходи. До роботи допускаються особи, які пройшли інструктаж з електробезпеки (I група допуску).

Протипожежна профілактика Згідно з НАПБ Б.03.002-2007, приміщення з комп'ютерною технікою відноситься до категорії «В» (пожежонебезпечні), а за класом зони згідно ПУЕ – до класу П-IIа. Основними причинами можливої пожежі є: перегрів електронних компонентів, коротке замикання, велике перехідний опір у контактах.

Система протипожежного захисту включає:

1. Пожежна сигналізація. Приміщення обладнане автоматичними димовими сповіщувачами типу СПД-3, підключеними до загального пульту спостереження.

2. Первинні засоби пожежогасіння. У приміщенні наявний вуглекислотний вогнегасник типу ВВК-2 (або ВВК-3). Вибір вуглекислотного вогнегасника зумовлений тим, що він не пошкоджує електронне обладнання (на відміну від порошкового) і дозволяє гасити електроустановки під напругою до 1000 В.

3. Евакуаційні заходи. План евакуації розміщено на видному місці біля виходу. Ширина проходів (не менше 1 м) забезпечує вільну евакуацію персоналу. Двері відчиняються у напрямку виходу з приміщення.

Виконання цих заходів зводить ризик виникнення пожежі до мінімуму та забезпечує безпеку персоналу у разі надзвичайної ситуації.

7.5 Розрахункова частина. Розрахунок штучного освітлення

Метою розрахунку є визначення необхідної кількості світильників для забезпечення нормованого рівня освітленості на робочому місці програміста 20,0 " м"^2 60,0 $м^3$

Розрахунок проводиться методом коефіцієнта використання світлового потоку.

1. Вихідні дані для розрахунку:

– Довжина приміщення (A): 5,0 м.

– Ширина приміщення (B): 4,0 м.

– Висота приміщення (H): 3,0 м.

– Висота робочої поверхні (роб): 0,8 м (висота столу).

– Висота підвісу світильників (св): 0,2 м (від стелі).

Розрахункова висота підвісу (h):

$$h = H - h_{роб} - h_{св} = 3{,}0 - 0{,}8 - 0{,}2 = 2{,}0 \text{ м}$$

2. Визначення індексу приміщення (i):

Індекс приміщення характеризує геометрію кімнати і впливає на те, як світло відбивається від стін.

$$i = \frac{S}{h \cdot (A + B)} = \frac{20}{2{,}0 \cdot (5{,}0 + 4{,}0)} = \frac{20}{18} \approx 1{,}11$$

3. Вибір коефіцієнтів:

– Коефіцієнти відбиття: Стеля – біла 70%, стіни – світло-бежеві 50%, підлога – темна 30%.

– Коефіцієнт використання світлового потоку eta: Для індексу i = 1,1 та обраних кольорів інтер'єру, згідно з довідковими таблицями, eta = 0,45 (або 45%).

– Коефіцієнт запасу (k): Враховує запилення ламп у процесі експлуатації. Для офісних приміщень з LED-лампами k = 1,4.

– Коефіцієнт нерівномірності освітлення **(z):** Для люмінесцентних та LED ламп z = 1,1.

4. Розрахунок необхідного світлового потоку (заг):

Загальний світловий потік, який повинні випромінювати всі лампи в кімнаті, розраховується за формулою:

$$F_{заг} = \frac{E_{норм} \cdot S \cdot k \cdot z}{\eta}$$

$$F_{заг} = \frac{400 \cdot 20 \cdot 1{,}4 \cdot 1{,}1}{0{,}45} = \frac{12320}{0{,}45} \approx 27377 \text{ лм}$$

5. Вибір світильників та ламп:

Для освітлення обираємо сучасні растрові LED-світильники, що вбудовуються у підвісну стелю (тип «Армстронг»).

Обрано світильник: Philips CoreLine Panel.

– Потужність одного світильника: 36 Вт.

– Світловий потік одного світильника (F_1): 3400 лм.

6. Визначення кількості світильників (N):

$$N = \frac{F_{заг}}{F_1} = \frac{27377}{3400} \approx 8{,}05$$

Приймаємо кількість світильників N = 8 шт.

7. Перевірка фактичної освітленості (E_факт):

$$E_{факт} = \frac{F_1 \cdot N \cdot \backslash eta}{S \cdot k \cdot z} = \frac{3400 \cdot 8 \cdot 0{,}45}{20 \cdot 1{,}4 \cdot 1{,}1} = \frac{12240}{30{,}8} \approx 397 \text{ лк}$$

Фактична освітленість (397 лк) відрізняється від нормованої (400 лк) менше ніж на 1%, що є допустимим (допускається відхилення -10%...+20%).

Таким чином, для приміщення площею 20 м² необхідно встановити 8 LED-світильників (розміщення у два ряди по 4 світильники), що забезпечить комфортні та безпечні умови праці розробника.

Висновки до розділу Заходи з охорони праці та техніки безпеки

У даному розділі було проведено комплексний аналіз умов праці при розробці програмного забезпечення.

1. Встановлено, що приміщення кафедри за своїми геометричними параметрами відповідає санітарним нормам (20 м² на одну особу).

2. Проаналізовано шкідливі фактори (електромагнітне випромінювання, шум, зорове напруження) та запропоновано заходи щодо їх нейтралізації.

3. Розроблено заходи електробезпеки (захисне заземлення) та пожежної профілактики (наявність вуглекислотних вогнегасників).

4. Проведено розрахунок штучного освітлення, в результаті якого визначено необхідність встановлення 8 растрових LED-світильників для забезпечення нормованої освітленості 400 лк.

Запропоновані заходи гарантують безпечні умови праці, збереження здоров'я розробника та високу ефективність роботи над дипломним проєктом.

8 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

– Досліджена система спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

– На основі отриманих результатів досліджень створена програмна реалізація системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання спостережуваності та моніторингу стану гібридних хмарно-корпоративних ІТ-комплексів.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Удудяк Б.О. Дослідження та програмна реалізація системи спостережуваності та моніторингу стану гібридних хмарно-корпоративних IT-комплексів // Збірник праць молодих науковців ЦНТУ. – Вип. 17. – Кропивницький: ЦНТУ, 2027.
2. Priscila Heller. Automating Workflows with GitHub Actions. Packt Publishing. 2021. 216 p.
3. JJ Geewax. API Design Patterns. Manning Publications Co. 2021. 481 p.
4. Prateek Prasad. App Design Apprentice. Razeware LLC. 2020. 272 p.
5. Dawn Griffiths, David Griffiths. Head First Android Development. O'Reilly Media, Inc. 2021. 1414 p.
6. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.
7. Aaron Torres. Go Programming Cookbook Second Edition. Packt Publishing Ltd. 2019. 427 p.
8. Мелешко Є.В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: Видавець – Лисенко В.Ф., 2019. – 156 с.
9. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.
10. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.
11. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.
12. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292

р.

13. Богаченко, С., Смірнов, О., Буравченко, К., Смірнова, Т., Конопліцька-Слободенюк, О., Якименко, Н., Смірнов, С. «Обмеження ресурсів мобільних та edge-платформ при розробці програмного забезпечення для реалізації та оптимізації моделей машинного навчання». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*, 2026, т. 2 № 34, С. 677-686.

14. Смірнов, О., Ткачук, Р., Козірова, Н., Константинова, Л., Конопліцька-Слободенюк, О., Якименко, Н., Смірнов, С.. «Дослідження застосування векторних баз даних у генеративному штучному інтелекті». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*, 2026, Том 1 № 33, 667-682.

15. Вінтенко Б.Ю., Смірнова Т.В., Миронець І.В., Смірнов О.А., Буравченко К.О. «Метод оцінки функціональної стійкості комп'ютерно-орієнтованих процедур системи підтримки оперативного персоналу АЕС». *Системи управління, навігації та зв'язку*, 2026, вип. 2(84), С. 62-69.

16. Смірнова Т.В., Усік П.С., Лисенко І.А., Буравченко К.О., Смірнов О.А. «Методологія підтримки технологічних процесів у критичній інфраструктурі з забезпеченням безпеки інформації на основі хмарних технологій». *Безпека інформації.* 2025. Том 31 № 1. С. 49-60.

17. Смірнова Т.В., Буравченко К.О., Добринчук О.А., Смірнов С.А., Якименко Н.М., Смірнов О.А. «Хмарна технологія моніторингу ключових індикаторів ефективності технологічних процесів у критичній інфраструктурі». *Центральноукраїнський науковий вісник. Технічні науки.* 2025. № 12(43), ч. II. С. 36-54.

18. Лисенко, І., Минайленко, Р., Смірнов, С., Буравченко, К., Якименко, Н., Смірнов, О. «Дослідження інструментів штучного інтелекту для інтелектуального аналізу даних». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*, 2025, № 3(31), С. 227–241.

19. Vintenko, B., Myronets, I., Smirnov, O., Smirnova, T. «Enhancing the

Reliability of Input Data for Nuclear Power Plant Operator Support Systems». *International Conference on Sensors, Devices and Systems. SDaS 2025. Lecture Notes in Electrical Engineering,* 2026. vol 1617. pp 3-12 Springer, Cham.

20. Смірнов О., Заріцкий В., Буравченко К., Конопліцька-Слободенюк О., Константинова Л., Якименко Н., Смірнов С. «Оптимізація розпізнавання облич за допомогою CUDA прискореної бібліотеки dlib». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2026, Том 4, № 32, С. 573-582.

21. Kuznetsov O., Frontoni E., Kuznetsova Y., Chevardin V., Smirnov O. «Architectural foundations for adaptive security in edge computing systems». *Cybersecurity Defensive Walls in Edge Computing*, 2025. pp. 21-61.

22. Вінтенко, Б.Ю., Миронець, І.В., Смірнов, О.А., Коваленко, О.В., Усік, П.С., Буравченко, К.О., Лисенко, І.А. «Логіко-структурна модель комп'ютерно-орієнтованої процедури системи підтримки оперативного персоналу АЕС». *Кібербезпека: освіта, наука, техніка*. 2025. Том 2 № 30. С. 413-427, 2025.

23. Смірнова, Т.В. «Дослідження методів, моделей та сучасних ІТ-рішень для підтримки технологічних процесів у критичній інфраструктурі держави». *Кібербезпека: освіта, наука, техніка*. 2025. Том 2 № 30. С.195-208, 2025.

24. Al-Azzeh, J., Ayyoub, B., Mesleh, A., Smirnova, T., Gnatyuk, S., Drieiev, O., Smirnov, O., Dorenskyi, O. «Cloud-Based Information System for Evaluating Caverns in the Process of Blasting Metal Surfaces of Details». *International Review on Modelling and Simulations* 18 (1), 2025. pp. 32-42.

25. Вінтенко Б.Ю., Смірнов О.А., Миронець І.В., Смірнова Т.В., Коваленко О.В., Мацуй А.М. «Модель шляхів отримання вхідних даних комп’ютерної інтелектуальної системи підтримки оперативного персоналу АЕС». *Центральноукраїнський науковий вісник. Технічні науки.* 2025. Вип. 11(42), ч. ІІ. С.52-62.

26. Смірнов, О.А., Константинова, Л.В., Конопліцька-Слободенюк, О.К.,

Козірова, Н.В, Якименко, Н.М., Доренський, О.П., Буравченко, К.О. «Дослідження інструментів штучного інтелекту для роботи з базами даних та аналізу даних». *Кібербезпека: освіта, наука, техніка.* 2025. №3(27), С. 429–448.**)**

27. Smirnov O., Fedorov E., Neskorodieva A., Neskorodieva T. «Intellectual Classification method of Gymnastic Elements Based on Combinations of Descriptive and Generative Approache». *CEUR Workshop Proceedings* Volume 3664, 2024, Pages 11-23.

28. Вінтенко, Б., Миронець, І., Смірнов, О., Коваленко, А., Конопліцька-Слободенюк, О., Смірнова, Т., Константинова, Л. «Дослідження застосування систем підтримки оперативного персоналу об'єкту критичної інфраструктури при керуванні енергоблоком АЕС з реактором типу ВВЕР-1000». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 6-26.

29. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

30. Kuznetsov O., Ilchenko O., Kryvinska N., Buravchenko K., Smirnov O., Savchenko Iu. «An Empirical Assessment of Leading Blockchain Financial Services». *2023 IEEE 1st Ukrainian Distributed Ledger Technology Forum (UADLTF)*, Kyiv, Ukraine, 2023, pp. 1-6,

31. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

32. Malyukov V., Bebeshko B., Lakhno V., Smirnov O., Malyukova I., Mohylnyi H. «Managing the Purchase-Sale Process of Digital Currencies Under Fuzzy Conditions». *Lecture Notes in Networks and Systems*, 2023, 729 LNNS, pp. 104–112.

33. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced*

Information Systems, 2023, 7(2), pp. 49-56.

34. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings*, Volume 3530, 2023, pp. 256-265.

35. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

36. Вінтенко Б.Ю., Смірнов О.А., Коваленко А.С., Смірнов С.А., Буравченко К.О. «Дослідження вимог міжнародних стандартів IEC60880 та IEC62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 3(73), С. 155-166.

37. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

38. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 2(72), С. 170-178.

39. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». Сучасні інформаційні системи, 2023, том 7, № 2, С. 49-56.

40. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А. «Дослідження нормативної документації та стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». VI

міжнародна науково-практична конференція “Інформаційна безпека та комп’ютерні технології”, м. Кропивницький. 20-21 квітня 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 35-36.

41. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

42. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w.

43. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143

44. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». Communications in Computer and Information Science, 2021, vol 1486. Springer, Cham. pp 169-184.

45. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

46. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

47. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

48. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting

Hidden Information in FAT». International Journal of Computer Network and Information Security (IJCNIS). Vol. 12, No. 3, 2020. PP.33-43.

49. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

50. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

51. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

52. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

53. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

54. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P. 707-712.

55. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21

September 2019. P.701-706.

56. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

57. Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.

58. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» Вісник Черкаського державного технологічного університету. Технічні науки. №4. С. 103-110. 2020.

59. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

60. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

61. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

62. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий

вісник. Технічні науки. № 1(32). с. 184-194, 2019.

63. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

64. O. Smirnov, O. Kovalenko, A. Kovalenko, S. Smirnov, V. Vialkova. The mathematical model of the testing technology for Dom Xss vulnerabilities. Scientific & practical cyber security journal (SPCSJ) Vol 2 Issue 1, 22-28 pp. [Электронный Журнал]. Georgia. Tbilisi: SCSA – 2018.

65. Oleksii Smirnov, Oleksandr Kovalenko, Jamil Al-Azzeh, Anna Kovalenko, Serhii Smirnov. Qualitative risk analysis of software development. Asian Journal of Information Technology. – Volume 17(3). – Medwell Journals. – 2018. – P. 218-230.

66. Смірнов О.А., Коваленко О.В., Коваленко А.С., Смірнов С.А. Розробка методу передтестової компіляції й розподілу доступу. Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”, м. Кропивницький. 19-20 квітня 2018р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215