Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Пояснювальна записка

до магістерської кваліфікаційної роботи

магістр

(освітньо-кваліфікаційний рівень)

на тему: Розробка системи моніторингу корпоративних серверів

Виконав: студент II курсу
групи 1ПІ-18 м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Колос Д.В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Коваленко О.О.
(прізвище та ініціали)

Рецензент: к.т.н., доц. каф. ПЗ Богач І.В.
(прізвище та ініціали)

ВНТУ – 2019

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Освітньо-кваліфікаційний рівень – магістр
Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
"____"_____2019 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Колосу Дмитру Володимировичу

1. Тема роботи – розробка системи моніторингу корпоративних серверів.

Керівник роботи: Коваленко Олена Олексіївна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від "__"_____2019 року №___

2. Строк подання студентом роботи

_____

3. Вихідні дані до роботи :

Мова програмування: Java

Технологія розробки: IntelliJ IDEA

Браузери (або ОС): Windows

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз методів створення систем моніторингу серверів; проектування системи моніторингу; розробка програмного засобу для моніторингу корпоративних серверів; тестування роботи системи; економічна частина; висновки; список використаних джерел; додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
моделі системи; інтерфейс; архітектура системи моніторингу; алгоритм роботи авторизації системи; архітектура системи; приклад вигляду розробленої програми.

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видано	завдання прийнято
1–4	к.т.н. Коваленко О.О., доцент кафедри ПЗ		
5	к.е.н. Бальзан М. В., доцент кафедри ЕПВМ		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Техніко-економічне обґрунтування доцільності розробки системи моніторингу корпоративних серверів	07.10.2019 – 27.10-2019	Вик.
2	Розробка модулів системи моніторингу корпоративних серверів	28.10.2019 – 8.11.2019	Вик.
3	Програмна реалізація додатку	9.11.2019 – 20.11.2019	Вик.
4	Тестування роботи системи	21.11.2019 – 3.12.2019	Вик.
5	Економічне обґрунтування розробки програмного продукту	4.12.2019 – 7.12.2019	Вик.

Студент _____ **Колос Д.В.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Коваленко О.О.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У магістерській кваліфікаційній роботі виконано аналіз стану проблеми сучасних систем моніторингу серверів та комп'ютерних мереж. Розглянуті загальні характеристики поширених систем моніторингу: CACTI, MRTG, Nagios і Zabbix, HP OpenView і IBM Tivoli.

Запропоновано удосконалення методу моніторингу кількості онлайн-користувачів корпоративного серверу в реальному часі та візуалізація їх навантаження. Одержала подальший розвиток модель системи моніторингу корпоративних серверів, яка на відміну від існуючих, сформована на основі методу моніторингу об’єктів з використанням інструментів просторового територіального моделювання, що дозволяє розширити функціонал процесів моніторингу. Набув подальшого розвитку метод збору метрик, який на відміну від існуючих що дозволяє сформувати систему показників моніторингу, що здійснюють спостереження за показниками поточних активних процесів; завдань, що завершені та процесів, які прогнозуються до виконання з можливістю визначення класу реалізації та кількості часу для виконання..

Розроблено інтерфейс основної частини системи моніторингу та додаткових модулів: веб-сервісу зв'язку, клієнтів, локальних серверів та баз даних. Розроблено алгоритм, діаграми класів та програмного коду.

Створено програмне оточення з розробки основних метрик системи моніторингу.

Визначені можливості використання системи Prometheus, описані кроки встановлення цієї системи з метою проведення тестування. Описано тестування системи моніторингу за допомогою Grafana. Виконано тестування файлів конфігурацій системи моніторингу, наведені основні кроки та протоколи роботи програмного засобу.

ABSTRACT

The master's qualification work analyzes the state of the problem of modern monitoring systems for servers and computer networks. Common features of common monitoring systems are discussed: CACTI, MRTG, Nagios and Zabbix, HP OpenView and IBM Tivoli.

It is suggested to improve the method of monitoring the number of corporate server online users in real time and visualizing their load. The corporate server monitoring system model, which, unlike the existing ones, has been further developed based on the object monitoring method using gamification tools and spatial territorial modeling, which allows to extend the functionality of the monitoring processes. The method of collecting metrics, which, unlike the existing ones, has been further developed, which allows for the formation of a system of monitoring indicators that monitor the indicators of current active processes; completed tasks and processes that are predicted to execute with the ability to determine the class of implementation and the amount of time to complete ..

The interface of the main part of the monitoring system and additional modules has been developed: web service of communication, clients, local servers and databases. Algorithm, class diagrams and program code are developed.

A software environment has been created to develop the basic metrics of the monitoring system.

Possibilities of using Prometheus system are identified, steps for installation of this system for testing purposes are described. Testing the monitoring system using Grafana is described. Tested the configuration files of the monitoring system, outlines the basic steps and protocols of the software.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Основним завданням системи моніторингу є надання актуальної інформації про інформаційні ресурси серверів і комп'ютерних мереж для аналізу стану IT-інфраструктури та швидкого виявлення виниклої несправності і її оперативне усунення [1,2].

Системи моніторингу дозволяють IT-фахівцям вчасно помітити зниження продуктивності і визначити «вузькі місця» в IT-інфраструктурі. Постійний моніторинг допомагає уникнути простоїв в її роботі, підтримувати всі IT-сервіси в робочому стані і зберігати необхідний рівень їх якості, а також спланувати її модернізацію.

Моніторинг в інформаційній структурі потрібен, щоб системні адміністратори були сповіщені про поломки і проблеми в інфраструктурі раніше або хоча б одночасно з користувачами. Актуальність даної роботи полягає в необхідності прогнозування, а тим самим і запобігання ситуацій простою, оповіщення про них і зберігання інформації про стан систем і служб в будь-який IT системі.

Раніше роль моніторингу здійснювали адміністратори, а інформація про стан систем в кращому випадку збиралася ними в будь-яких неспеціалізованих програмах, в гіршому випадку – взагалі ніяк не накопичувалась. Всі відомості про систему були прив'язані до практичного досвіду роботи з інфраструктурою у конкретного фахівця і повністю втрачались при його звільненні.

Зараз з'явилося безліч наполовину або повністю автоматизованих систем для моніторингу, які аналізують стан систем, збирають інформацію в колекції, які теж згодом можна збирати і вивчити при необхідності для подальшого аналізу.

Для зберігання отриманої інформації зазвичай використовується конфігураційна база даних під різними системами управління базами даних (СУБД): інформація про об'єкти моніторингу представлена, як набір

конфігураційних одиниць. Кожен сервер, кожен мережевий пристрій - це окрема одиниця, дані щодо якої обробляються та зберігаються в централізованій базі даних. Такий підхід дозволяє інтегрувати систему моніторингу з використанням інструментів візуалізації.

Системи моніторингу можуть бути орієнтовані на споживачів різного рівня. Для великих систем зазвичай використовується величезна кількість різноманітних функцій, для невеликих використовують спільний аналіз вузлів та відправку сповіщень.

Завдяки наявності засобів для реалізації функцій адміністратора більше не потрібно перевіряти вручну стан кожної складової системи, проблеми вирішуються і поломки усуваються більш оперативно, діагностика здійснюється точно.

Гіпотеза дослідження полягає в тому, що програмний додаток системи моніторингу ІТ-інфраструктури може мати розширений функціонал відповідно до вимог користувачів та аналізу ризиків несправностей та зменшення продуктивності роботи кожного елементу та всієї ІТ-інфраструктури в цілому.

Мета та завдання дослідження. Метою дослідження є розширення функціоналу системи моніторингу об’єктів ІТ-інфраструктури.

Основними задачами дослідження є:

- аналіз відомих методів та інструментарію моніторингу об’єктів ІТ-інфраструктури;

- розробка загального переліку вимог до систем моніторингу;

- дослідження існуючих рішень (аналогів) і обґрунтований вибір основи майбутнього моніторингу;

- розробка та аналіз моделей процесів моніторингу об’єктів ІТ-інфраструктури;

- розробка списку конкретизованих вимог до майбутнього моніторингу та відповідних їм методів та інструментів;

- практична реалізація системи моніторингу з розширеним функціоналом;

- тестування розробленого програмного додатка;

- аналіз організаційних процесів впровадження в експлуатацію та перевірка ефективності розробки.

Об’єктом дослідження є процес розробки програмного додатку з управління моніторингом інформаційних та технологічних ресурсів серверів.

Предметом дослідження є методи та інструменти моніторингу серверів та інфраструктури.

Методи дослідження включають в себе: методи теорій алгоритмів, моделювання, моделювання; алгоритмізації і програмної реалізації математичних моделей; оцінювання ризиків та прогнозування результатів моніторингу.

Наукова новизна отриманих результатів.

1. Одержала подальший розвиток модель системи моніторингу корпоративних серверів, яка на відміну від існуючих, сформована на основі методу моніторингу об’єктів з використанням інструментів просторового територіального моделювання та комплексної обробки даних індикаторів вбудованих та зовнішніх систем моніторингу, що дозволяє розширити функціонал процесів моніторингу.

2. Набув подальшого розвитку метод збору метрик, який на відміну від існуючих що дозволяє сформувати систему показників моніторингу, що здійснюють спостереження за показниками поточних активних процесів; завдань, що завершені та процесів, які прогнозуються до виконання з можливістю визначення класу реалізації та кількості часу для виконання.

3. Запропоновано удосконалення методу моніторингу кількості онлайн-користувачів корпоративного серверу в реальному часі та візуалізація їх навантаження.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень розроблено програмний засіб моніторингу корпоративних серверів.

Структура та обсяг роботи. Магістерська кваліфікаційна робота складається зі вступу, п’яти розділів, висновків, списку літератури, що містить 25 найменувань, 3 додатків. Робота містить 17 ілюстрацій, 5 таблиць.

Робота складається з 5 розділів. У першому розділі проаналізовано сучасний стан питання теми розробки, проведено порівняння аналогів і обґрунтовано вибір засобів реалізації мобільного додатку, сформульовано задачі дослідження.

Другий розділ містить аналіз структури, принципів та особливостей розробки мобільних додатків, особливостей розробки бази даних, а також здійснено розробку методу і засобів реалізації автоматизованої системи.

У третьому розділі розроблено інтерфейс, базу даних та проведена програмна реалізація мобільного додатку.

Четвертий розділ показує результати тестування розробленого мобільного додатку.

У п’ятому розділі розраховуються витрати на розробку програмного забезпечення, експлуатаційні витрати, обсяг роботи, пов'язаної з використанням програмного забезпечення, та економічний ефект від впровадження нового програмного продукту.

У висновках наведені основні результати дослідження.

У додатках міститься технічне завдання, лістинг коду основних частин програми та ілюстративний матеріал до захисту магістерської кваліфікаційної роботи.

1 АНАЛІЗ МЕТОДІВ СТВОРЕННЯ СИСТЕМ МОНІТОРИНГУ IT-ІНФРАСТРУКТУРИ

1.1 Аналіз існуючих методів та інструментарію систем моніторингу IT-інфраструктури

Існуючі системи моніторингу умовно можна розділити на системи [1,2], що реалізують активний і пасивний моніторинг (рис. 1.1).

Рисунок 1.1 - Категорії систем моніторингу

В даному випадку під пасивним моніторингом розуміється отримання даних в режимі читання, наприклад, системи збору даних про температуру, про завантаження процесора, про споживання оперативної пам'яті. Під активним моніторингом слід розуміти моніторинг з елементами впливу на середовище (операційну систему, додатки, апаратне забезпечення). Прикладом може служити система, яка при певних зовнішніх умовах або ж при певних значеннях параметрів виконує коригувальну дію. Системи моніторингу будуються по архітектурі клієнт-сервер [3].

Взаємодія клієнта і сервера здійснюється за допомогою стандартних, або ж власних протоколів, а дані передаються через мережі передачі даних.

Сервер зберігає, використовує і модифікує поточну конфігурацію для виконання моніторингу. Власне, сервер проводить зондування системи, дає оповіщення, якщо відбулися збої, зберігає у своїй конфігурації результати зондування для подальшого виведення їх в графічному вигляді.

Сам по собі сервер не здатний графічно відображати схему мережі. Для отримання графічного зображення, а також деяких видів оповіщення про збої в системі, використовується клієнт.

Клієнт не зберігає ніякої інформації, крім розташованої в оперативній пам'яті, тобто призначеної для відображення.

Основне призначення клієнта - намалювати красиву картинку і, в разі збоїв або інших подій, відобразити це в клієнтської консолі для зручного сприйняття.

Друге призначення клієнта - це графічний інтерфейс користувача, призначений для конфігурації сервера.

Більшість сучасних активних систем моніторингу IT-інфраструктури використовують однаковий принцип: система моніторингу якимось чином опитує обладнання або програмне забезпечення, отримує результат і порівнює його або з шаблоном, або з наперед заданими гранично допустимими значеннями.

Так, для визначення доступності SMTP сервера, система моніторингу повинна підключитися до сервера на 25 порт TCP, передати рядок "hello my.monitoring.com" отримати у відповідь рядок і відключитися від SMTP сервера [2,4].

Далі перевірити, чи містить рядок відповіді сервера на початку тризначний код, що починається з цифри 2.

Якщо це так, то сервер працездатний, якщо немає, система повинна дати сповіщення. Фактично, це метод порівняння з шаблоном.

Інший приклад - перевірка завантаження процесора. Система моніторингу, найчастіше по SNMP, опитує сервер, отримує значення поточної

завантаження процесора, і порівнює його з гранично допустимим максимальним значенням, припустимо 80.

Якщо процесор завантажений більш ніж на 80%, система моніторингу повинна дати сповіщення. Це - метод перевірки гранично допустимих значень опитуваних величин.

У будь-якому випадку, в процесі конфігурації системи моніторингу необхідно ставити чіткі критерії того, що треба вважати нормальною роботою устаткування або програмного забезпечення, а що - збоєм, або ситуацією, яка може привести до збою в найближчому майбутньому.

Такий принцип моніторингу працює в більшості випадків. Однак, часом задати критерії того, що вважати збоєм, а що нормальною роботою, занадто складно або взагалі неможливо.

Крім вищезгаданого SNMP, який спеціально був розроблений для вирішення завдань передачі даних в системах моніторингу [4-7], практично у кожної системи моніторингу існують і власні реалізації протоколів обміну даними, але SNMP є найбільш популярним і затребуваним за рахунок розширюваності і відкритості інтерфейсу.

Крім цього, SNMP може бути використаний як в активному моніторингу, так і в пасивному [8-12].

Для того, щоб визначити, який тип моніторингу варто застосовувати в тому чи іншому випадку необхідно розглянути більш детально галузі використання як пасивного моніторингу, так і активного.

Пасивний моніторинг. До даного класу відносяться системи, які використовуються для відстеження виникнення несправностей або позаштатних ситуацій. Після збору інформації з джерел даних можливий ряд дій, серед яких - відображення отриманої інформації оператору, а в разі зміни параметрів за межі, визначені як «нормальні», прийняття певних кроків для усунення виниклої ситуації і нормалізації параметрів.

Формат оповіщення може бути різним: це і побудова графіків, і генерація повідомлень в пріоритетному режимі для більш оперативного відображення оператору тощо. Представниками подібних систем є MRTG (MultiRouter Traffic Grapher) [13] і CACTI [11].

Незаперечною перевагою даних програмних продуктів є їх безкоштовне використання. Розробники MRTG створили його для контролю завантаженості інтерфейсів на мережевих пристроях (комутатори і маршрутизатори). Як наслідок, MRTG стало популярним серед компаній, що працюють в галузі зв'язку [14-16].

CACTI пропонує більш зручний інтерфейс, але і вимагає великих витрат на встановлення та налаштування [11]. Однак, дані типи систем моніторингу не дозволяють в режимі реального часу відслідковувати будь-які показники, але дозволяють зберігати статистику і відображати її у вигляді графіків.

У зв'язку з тим, що в обчислювальній системі щомиті трапляються якісь події як в ОС, так і на апаратному рівні, то в разі їх незапланованого формату вони повинні бути перехоплені. Наприклад, спрацьовування датчиків перевищення температури, датчиків руху, датчиків задимленості.

Для відстеження подібних подій можна використовувати такі системи моніторингу, як Nagios і Zabbix [16]. Nagios складний в первинній настройці, але зручний при подальшому використанні. Інтерфейс Nagios орієнтований на наявність персоналу, який стежить за станом показників системи.

Дані надходять не в реальному часі, але час опитування елементів можна регулювати залежно від потреб. Дистанційні системи опитуються за допомогою програмних інтерфейсів. Відповідно при настанні будь-яких подій в інтерфейсі виводяться відповідні інформаційні повідомлення.

Даний програмний продукт орієнтований саме на спрацьовування певних подій. Zabbix має веб-інтерфейс для адміністрування та налаштування [16].

На відміну Nagios, Zabbix досить самостійний і зможе відправити повідомлення на пошту або sms за допомогою gsm-модему, або навіть спробувати самостійно підняти впав сервіс, виконавши заздалегідь певні дії.

Існує також активний моніторинг. Активний моніторинг характеризується тим, що на певні події, які відбуваються, існує заздалегідь задану дію, яке імовірно призводить до вирішення виниклої проблеми.

Таким чином, активний моніторинг характерний наявністю зворотного зв'язку. Прикладами таких систем є HP OpenView і IBM Tivoli [17,18]. Це комплексні системи, які можна в сукупності іменувати інтелектуальними системами, що генерують в залежності від виникнення подій активності у відповідь дії для відновлення необхідних показників.

У повній же мірі в цю категорію потрапляють системи формату «розумний будинок», які активно проводять моніторинг ситуації і можуть здійснювати за заданою логікою необхідні дії.

Крім того, при виборі системи моніторингу слід врахувати ступінь їх захищеності від виникнення наступних критичних чинників:

- відмова окремих елементів або системи в цілому;
- умисне шкідливий вплив на систему.

Ще однією важливою проблемою є те, що в таких системах найчастіше існує центральний сервер / диспетчер, який виконує всю обчислювальну навантаження, і, в разі виходу його з ладу, виникає загроза працездатності всієї системи [19,20].

Якісним способом усунення цієї проблеми може бути створення системи, яка самостійно стежила б за станом своїх вузлів і, в разі необхідності, перерозподіляла завдання вузла який вийшов з ладу.

При виборі, розробці, впровадженні систем моніторингу спочатку необхідно визначитися з об'єктами, які будуть піддаватися стеженню, а також критичні події і показники, які і визначать кількість повідомлень при поломці, частоту сканування і інші параметри і наслідки. Для великих інфраструктур

розгортають тестови майданчик, де можна оцінити доцільність зроблених рішень і ухвал параметрів порогових значень.

Впровадження подібних рішень особливо важливо при використанні сервісного підходу до діяльності IT-підрозділів, коли всі процеси переглядаються з точки зору що надаються підрозділом IT-сервісів. Кожен бізнес-сервіс корпоративної системи по можливості інтерпретується як IT-сервіс, задається певний рівень якості його надання. Далі він описується в системі моніторингу як набір взаємопов'язаних компонентів IT-інфраструктури.

Таким чином, можна відмітити, що в міру подальшої модернізації виробництв, ускладнення систем та збільшення частки автоматизації у технологічних процесах з'являється необхідність контролювати роботу серверів та комп’ютерних мереж. Загальна значимість систем моніторингу буде підвищуватись. Але з ростом ролі систем моніторингу необхідно приділяти більшу увагу питанням правильної їх роботи та захисту. Також, буде зростати роль систем захисту віддаленого моніторингу від зовнішніх впливів, в зв'язку з чим варто розвивати існуючі системи в сторону самодіагностики і прогнозування можливих загроз безпеки [21,22].

1.2 Порівняльний аналіз аналогів системи моніторингу

В теперішній час існує дуже велика кількість open source та комерційних систем моніторингу. Зведені відомості з найвідоміших з них показано у таблиці 2.1. Розглянемо найпоширеніші з них. Nagios - один з найвідоміших інструментів з відкритим кодом для моніторингу IT-інфраструктур, в тому числі робочих станцій кінцевого користувача, IT-сервісів і активних мережевих компонентів. Крім безкоштовної версії з відкритим кодом Nagios Core, є комерційна Nagios XI з додатковими можливостями. Ця версія має більш сучасний і простий в навігації веб-інтерфейс, який пропонує інтерактивну інформаційну панель з оглядом хостів, сервісів і мережевих пристроїв. Можливість побудови графіків тенденцій і наявність наочних

інструментів планування потужності допомагають в підготовці модернізацій інфраструктури.

За допомогою цієї програми доступні комплексне спостереження за всією IT-інфраструктурою, виявлення проблем відразу після їх виникнення, можливість ділитися отриманим при спостереженні даними із зацікавленими особами, моніторинг безпеки системи, і, як наслідок, скорочення часу простою і комерційних втрат.

Можливості Nagios:

- моніторинг мережевих служб (POP3, HTTP, SMTP, NNTP, SNMP);
- моніторинг стану хостів (використання диска, завантаження процесора, системні логи);
- підтримка віддаленого моніторингу через шифровані тунелі SSH або SSL;

Таблиця 2.1

Порівняльна характеристики важливих систем моніторингу

Особливість систем моніторингу	Nagios	Zabbix
Зберігання відомостей про конфігурацію системи	у файлах	в базі даних
Виконання операцій мониторинга	за допомогою плагінів	за допомогою агентів
Формат файлів моніторинга	простий	складний
Вбудовані засоби візуалізації	відсутні	розвинуті
Можливість встановлення конфігурації через інтерфейс	не існує	існує
Вступ змін в конфігурації системи	існує необхідність перезавантаження сервера	не існує необхідність перезавантаження сервера
Моніторинг продуктивності серверів	не існує	існує

- можливість побудови карт мереж
- проста архітектура модулів розширень (плагінів) дозволяє,

- використовуючи будь-яку мову програмування (Shell, C ++, Perl, Python, PHP, C # та інші), легко розробляти свої власні способи перевірки служб;
- паралельна перевірка служб;
- можливість визначати ієрархії хостів мережі за допомогою «Батьківських» хостів, дозволяє виявляти і розрізняти хости, які вийшли з ладу, і ті, які недоступні;
- відправка сповіщень в разі виникнення проблем зі службою або хостом (за допомогою пошти, пейджера, смс, або будь-яким іншим способом,
- певним користувачем через модуль системи);
- можливість визначати обробники подій, що відбулися зі службами або хостами для проактивного вирішення проблем;
- автоматична ротація лог-файлів;
- можливість організації спільної роботи декількох систем моніторингу з метою підвищення надійності і створення розподіленої системи моніторингу;
- включає в себе утиліту nagiostats, яка виводить загальне зведення по всім хостам, за якими ведеться моніторинг.

До недоліків системи відносяться наступні причини:

- «загальний» характер моніторингу показників;
- проблема взаємодії з серверами під управлінням Windows;
- «мережева» спрямованість моніторингу;

Інсталяція системи моніторингу Nagios досить проста, але робота з файлами для управління пристроями і тестами потребують ґрунтовного вивчення документації.

Навколо Nagios сформувалося велике і активна спільнота підтримки, учасники якого розробляють нові плагіни, що усувають недоробки основного інструменту, такі як труднощі конфігурації і відсутність можливості

автоматичного розпізнавання пристроїв. Є плагін для підтримки віртуальних середовищ.

Інша систем моніторингу - Zabbix. Zabbix - це система з відкритим кодом, що характеризується високою швидкодією при зборі даних і масштабована до корпоративного рівня. Вона дозволяє вести моніторинг серверів, мережевих пристроїв і додатків зі збором детальної статистики, що стосується продуктивності. Zabbix відрізняється простотою інсталяції, але конфігурація може викликати складності, особливо якщо потрібно налаштовувати особливі режими перевірки. Zabbix має якісний веб-інтерфейс і розвинені засоби створення звітів і побудови графіків. Все це входить в стандартний пакет, який, крім функцій моніторингу, реалізує можливості виявлення тенденцій. Повідомлення про вихід за допустимі значення параметрів система відправляє по електронній пошті і SMS. Як і у Nagios, у Zabbix є активна спільнота підтримки.

Zabbix складається з чотирьох частин:

- Сервер моніторингу (ядро) - виконує періодичний опитування і отримання даних, обробляє їх, аналізує, також здійснює запуск скриптів для розсилки повідомлень.Є сховищем, в якому зберігаються всі конфігураційні, статистичні та оперативні дані та може віддалено перевіряти мережеві сервіси. Не може розташовуватися на сервері під керуванням операційної системи сімейства Windows, а також OpenBSD.
- Проксі - збирає дані про продуктивність і доступність від імені Zabbix сервера. Він може бути використаний для розподілу навантаження одного Zabbix сервера. В цьому випадку, проксі тільки збирає дані, тим самим на сервер лягає менше навантаження на ЦПУ і на введення / виведення диска. Всі зібрані дані заносяться в буфер на локальному рівні і передаються Zabbix сервера, до якого належить проксі сервер. Zabbix проксі є ідеальним рішенням для централізованого віддаленого моніторингу місць, філій, мереж, які не мають локальних адміністраторів

- Агент - спеціальний засіб, який запускається на відслідковуються об'єктах і надає дані сервера, здійснюючи контроль локальних ресурсів і додатків (таких як жорсткі диски, пам'ять, статистика процесора і т. д.) на мережевих системах, тобто ці системи повинні працювати з запущеним Zabbix агентом (проте моніторинг можна проводити не тільки за допомогою нього, але і по SNMP версій 1, 2, 3, запуском зовнішніх скриптів, що видають дані, і кілька видів зумовлених вбудованих перевірок, таких як ping, запит по http, ssh, ftp і іншим протоколам, а так же завмер часу відповіді цих сервісів. Zabbix агенти є надзвичайно ефективними через використання вбудованих системних викликів для збору інформації про статистикою. Zabbix-агенти підтримуються не тільки на * nix операційних системах, а й на AIX і Windows.

- Веб-інтерфейс - засіб візуального представлення Zabbix, реалізований на мові програмування PHP, для запуску вимагає наявності веб-сервера

За допомогою Zabbix можна здійснювати розподілий моніторинг до 1000 вузлів, де конфігурація молодших вузлів контролюється старшими в ієрархії. Також продукт включає централізований моніторинг лог-файлів, можливість створювати карти мереж (вручну за шаблоном), виконання запитів в різні бази даних, генерацію звітів і тенденцій, виконання сценаріїв на основі моніторингу, підтримку інтелектуального інтерфейсу управління платформами (IPMI).

Zabbix надає гнучкі можливості по налаштуванню умов-тригерів, які включаються при аваріях і неполадках, і система починає моргати лампочками (насправді червоними квадратиками), оповіщаючи адміністратора про

можливої поломки. Також, при включенні тригера, веб-інтерфейс навіть починає пищати на манер будильника.

Zabbix досить самостійний і зможе відправити повідомлення на пошту, в jabber або sms за допомогою gsm-модему, або навіть спробувати самостійно

підняти впав сервіс, виконавши заздалегідь певні дії, які запускаються при спрацьовуванні певних тригерів.

Cacti - безкоштовний додаток моніторингу, що дозволяє збирати статичні дані за певні часові інтервали і відображати їх в графічному вигляді за допомогою RRDtool утиліти, призначеної для роботи з круговими базами даних (Round Robin Database), які використовуються для зберігання інформації про зміну однієї або декількох величин за певний проміжок часу. Стандартні шаблони збору включають статистику по завантаженню процесора, виділенню оперативної пам'яті, кількістю запущених процесів, використання вхідного / вихідного трафіку.

Інтерфейс відображення статистики, зібраної з пристроїв, представлений у вигляді дерева, структура якого задається самим користувачем. Як правило, графіки групують за певними критеріями, при цьому в різних гілках дерева може бути присутнім один і той же графік. Є варіант перегляду заздалегідь складеного набору графіків, і є режим попереднього перегляду. Кожен з графіків можна розглянути окремо, при цьому він буде представлений за останні день, тиждень, місяць і рік. Можливо самому вибрати часовий проміжок, за який буде згенеровано графік.

Cacti написаний в інфраструктурі Apache-PHP-MySql, дозволяє налаштовувати збір і відображення даних моніторингу на основі веб-інтерфейсу, представленого на рисунку 1.2, з юзер-френдлі організацією. Є можливість дописування власних агентів збору даних.

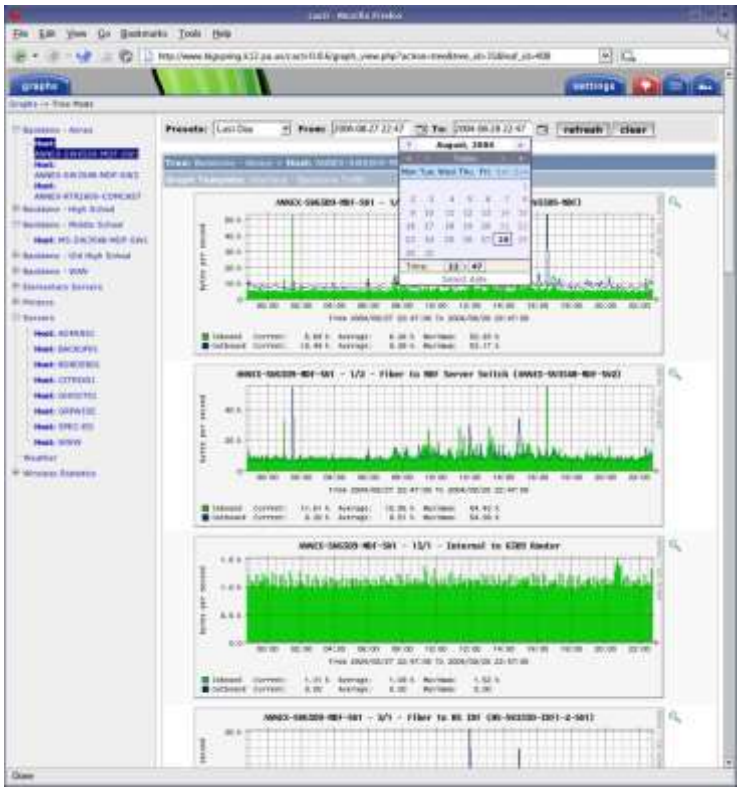

Рисунок 1.2 – Інтерфейс Cacti

Переваги Cacti:

- висока швидкість розгортання при мінімальному додатковому кодуванні;
- простота і зручність інтерфейсу перегляду графіку і їх налаштування.

Недоліки Cacti:

- досить швидке наростання кількості однотипних налаштувань в разі великого числа середовищ і серверів;
- обмежена продуктивність «нерідних» JMX рішень для Cacti;
- відсутність можливості інвентаризації.

Cacti дозволяє завести кілька користувачів і розмежувати їх права як на перегляд статистики, так і на управління системою. Логіка поділу доступу дозволяє для кожного користувача встановити загальну політику («Заборонити» або «Дозволити»), а потім зробити з неї виключення.

В результаті Cacti дозволяє малювати графіки тільки основних показників продуктивності, тоді як спроби моніторингу нестандартних

показників сильно знижують продуктивність продукту. Також проектом дуже часто необхідна інвентаризація (перерозподіл ресурсів і планування модернізації), а в даному пакеті вона відсутня.

Відомий Hyperic - проект компанії VMware, що представляє собою систему моніторингу та адміністрування для віртуальних середовищ. На практиці моніторингу систем пропонуються безкоштовна версія з відкритим кодом Hyperic HQ і комерційна vFabric Hyperic. Рішення забезпечує ефективне управління багатьма операційними системами, веб-серверами, а також серверами додатків і баз даних. Серед додаткових можливостей vFabric Hyperic - автоматизоване усунення неполадок.

Інсталяція цього програмного засобу досить проста і займає хвилини. Hyperic має простий та зручний призначений для користувача інтерфейс з продуманим дизайном. Передбачена можливість редагування інформаційних панелей - наприклад, додавання часто використовуваних графіків. Повідомлення приходять по SMS або електронною поштою, можна призначати адміністративні операції, які повинні виконуватися при отриманні тих чи інших повідомлень. Hyperic здатний автоматично розпізнавати програмне забезпечення та мережеві ресурси. Є активна спільнота підтримки.

Головний недолік Hyperic - великі, в порівнянні з іншими інструментами, потреби в ресурсах, які потрібні віртуальній машині Java.

Інший програмний засіб - SolarWinds пропонується в якості локально розміщується рішення і SaaS-сервісу. Встановлення може займати від декількох хвилин до декількох годин в залежності від складності конфігураційних даних і зазвичай не вимагає допомоги постачальника. Система легко масштабується і може використовуватися у великих організаціях. Вона також забезпечує нативную підтримку VMware. Консультації по SolarWinds можна отримати у відповідному онлайн-співтоваристві.

Його інтерфейс інтуїтивно зрозумілий, з налаштованим формами введення та можливістю доступу з мобільних пристроїв. Докладні графіки відображають мережеві збої, рівні готовності і швидкодії. Повідомлення легко налаштовуються, є можливість створювати складні послідовності операцій на основі правил. SolarWinds надає заздалегідь сконфігуровані інформаційні панелі, які можна змінювати на свій розсуд. Система генерує настроюються звіти, в тому числі автоматично за наперед заданим розкладом.

Конкурент SolarWinds - ManageEngine OpManager інсталюється швидко і легко, але конфігурувати його потрібно вручну, що може викликати труднощі. Згодом можна автоматизувати рутинні операції супроводу і усунення несправностей.

OpManager пропонує кілька інформаційних панелей, компоновку яких можна змінювати. Однак, навігація по призначеному для користувача інтерфейсу досить складна. Інструмент генерує багато видів звітів і дозволяє налаштовувати повідомляти в разі виході за порогові значення з відправкою по електронній пошті, SMS і через довільні скрипти. Є три рівні повідомлень: «попередження», «пошкодження» та «помилка». Для OpManager пропонується ряд плагінів - все як самостійні продукти.

Ще існує HP Operations Manager - це центральний компонент комплексу для системного моніторингу від HP, що представляє собою клієнт-серверне рішення, яке вимагає наявності програмних агентів на кожному вузлі. Якщо потрібно встановити кілька пакетів, початкове налаштування може виявитися непростим.

HP Operations Manager має чудовий графічний інтерфейс користувача для моніторингу стану додатків, серверів, систем і мережі. До складу рішення входять засоби планування, в тому числі інструменти аналізу прогнозе та моделювання центрів обробки даних. Повідомлення можна фільтрувати за ваги помилки і типу вузла. Є механізми попереджувального моніторингу та автоматизованої видачі повідомлень. Відомості про події супроводжуються

рекомендаціями щодо виправлення ситуації, є готові інструменти і автоматизовані операції усунення пошкоджень.

Встановлення Tivoli проста і займає кілька хвилин, але конфігурація, оновлення і тонке налаштування засобів аналітики і реагування вимагають певної кваліфікації [23].

Ця система моніторингу має інтуїтивно зрозумілий веб-інтерфейс з конфігуруються робочими просторами, забезпечена простим у використанні сховищем даних і розвиненими засобами звітності. Є аткож механізм динамічного аналізу порогових значень і продуктивності, що допомагає в запобіганні інцидентів. Існують підсистеми попереджувального моніторингу та автоматизованого керування збоями. Користуючись накопиченими даними, інструментарій реалізує функції звітності, аналізу продуктивності та виявлення тенденцій.

Відома компанія IBM надає підтримку по телефону та електронною поштою в робочий час, а також велику документацію і призначену для користувача базу знань.

Програмний продукт WhatsUp Gold встановити досить неважко - конфігурація автоматично виконується за допомогою веб-консолі і Windows-програми. Пропонується більше 200 звітів, що налаштовуються, в тому числі по тенденціям, виявленим в процесі аналізу даних за минулі періоди. Можливе формування звітів в реальному часі, що допомагає в усуненні пошкоджень [24].

На практиці доступні кілька плагінів, що розширюють можливості системи. Однак призначений для користувача інтерфейс незручний навіть при доступі до простих функцій. Наприклад, при створенні звітів за індивідуальними елементам виконуються поширені функції системи моніторингу. При виході параметрів пристроїв за порогові значення повідомлення можуть відправлятися по електронній пошті, SMS або через довільні скрипти.

1.3 Вимоги щодо побудови системи моніторингу IT-інфраструктури

Головне завдання моніторингу корпоративних серверів - це надання даних для їх подальшої обробки та аналізу з точки зору робочих бізнес-процесів і підтримання їх працездатності.

Система моніторингу корпоративних серверів та мережевого обладнання дозволяє попереджувати можливі проблеми, виявляти поточні проблеми, а також вести докладну статистику, виробляти різного роду повідомлення при настанні певних ситуацій, заздалегідь описаних в системі.

Завдяки системі моніторингу серверів, мережі та робочих станцій у разі інциденту можна зрозуміти, що є первинним чинником в порушенні роботи інфраструктури, а що - похідними факторами. Ці знання дозволяють направляти сили IT-служби на усунення саме причини проблем, а не наслідків.

До основних вимог побудови системи моніторингу відносяться такі (рис. 1.3):

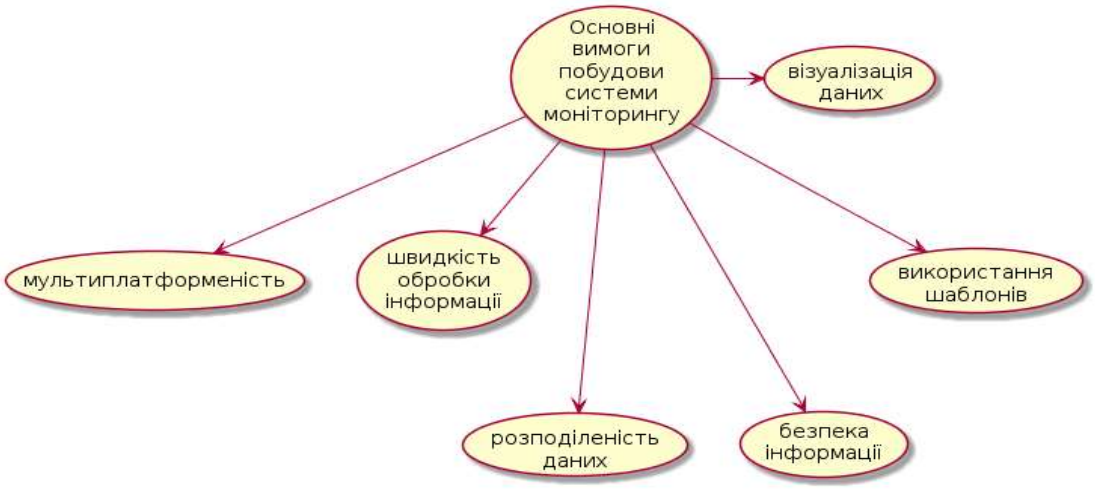

Рисунок 1.3 - Основні вимоги щодо побудови системи моніторингу

- мультиплатформеність;
- швидкість обробки інформації;
- розподіленість в отриманні даних;
- безпеку інформації;
- використання шаблонів;

- візуалізація даних.

Реалізація завдань мультиплатформенності забезпечується за рахунок таких рішень:

- моніторинг на основі платформ Linux, Unix, Solaris, BSD;

- моніторинг на основі технології SNMP v1, v2, v3 - пристроїв.

Реалізація завдань швидкість обробки інформації може бути забезпечене за допомогою таких рішень:

- тестування засобів моніторингу до 10000 пристроїв серверів;

- тестування з виконанням 100000 перевірок доступності та продуктивності обчислювальних систем;

- обробка декількох тисяч перевірок доступності та продуктивності в секунду на центральному сервері.

Рішення задач распределенности в отриманні даних може бути забезпечено за рахунок:

- централізованої конфігурації;

- централізованого доступу до необхідної інформації;

- повідомлення користувачів про критичні ситуації по електронній пошті, SMS, Jabber і т.д .;

- виконання заданих дій при будь-яких інцидентах.

Реалізація завдань в області безпеки інформації може бути забезпечене за допомогою таких рішень:

- моніторингу доступності;

- моніторингу цілісності;

- моніторингу продуктивності;

- моніторингу прав користувачів;

- аутентифікації по IP адресою або доменному імені (географічному положенню);

- захист від brute-force attack.

Гнучкість шаблонів повинно забезпечуватися за допомогою:

- можливості гнучкого налаштування шаблонів для моніторингу IT інфраструктури (моніторинг стану серверів, моніторинг мережевого обладнання);

- необхідних компонент на заданих вузлах;

- простого імпорту / експорту шаблонів.

Рішення задач візуалізації отриманих даних може бути забезпечено за рахунок:

- побудови звітів;

- створення графіків і діаграм;

- побудови мережевих карт.

До іншим завданням побудови систем моніторингу можуть належати:

- зручний Web-інтерфейс для управління системою моніторингу серверів;

- автоматичне виявлення по мережі комп'ютерів і мережевих пристроїв;

-агрегування даних на стороні агента моніторингу серверів для зменшення навантаження і трафіку на систему моніторингу;

- локалізацію (підтримка різних мов в інтерфейсі користувача).

Таким чином, система впровадження моніторингу до крупної організації являє собою складну проблему. Тому, як правило, системою моніторингу на підприємстві займається спеціальний відділ. Основні задач роботи цього відділу поділяються на технічні та організаційні вимоги. До технічних вимог відносяться наступні:

- абонентське обслуговування комп'ютерів організацій;

- сервісне обслуговування локальних мереж;

- настройка і моніторинг фізичних і віртуальних серверів;

- супровід і моніторинг серверного обладнання;

- віддалене адміністрування IT інфраструктури;

- організація та підтримка робочих місць;

- резервне копіювання даних;

- оновлення і конфігурація програмного забезпечення;

- роботи з модернізації інфраструктури;

- технічна підтримка та консультації користувачів.

До організаційних вимог роботи відділу відносяться такі:

- вибір і поставка обладнання та ліцензій;

- надання персонального менеджера;

- взаємодія з провайдерами інших послуг;

- надання гойдалки робочих станцій;

- супровід ремонту апаратури;

- цілодобова віддалена технічна підтримка користувачів.

До задач основних впровадження системи моніторингу відносяться наступні:

- отримання вхідних даних;

- аналіз вхідних даних, підготовка технічного завдання.

-узгодження технічного завдання, передбачуваного обладнання та бюджету проекту;

- підготовка обладнання;

- установка сервера моніторингу;

- настройка систем моніторингу згідно технічного завдання;

- проведення приймально-здавальних випробувань;

- усунення виявлених проблем;

- навчання персоналу замовника.

Адміністратор, який відповідає за супровід мережі та інформаційних ресурсів серверів, завжди стикається з завданням моніторингу всієї інфраструктури компанії. Раніше такі завдання вирішувалися шляхом написання сценаріїв для bat- або shell- скриптів, що працюють за розкладом і відправляють результати діагностики по електронній пошті. Однак, у такого підходу є явний ряд недоліків.

1. Наявність навичок програмування: незважаючи на те, що більшість системних адміністраторів вміють писати сценарії на мовах Perl, Python, Shell і Powershell. Написання скрипта може стати першою проблемою, яку адміністратор повинен буде подолати. Як результат - багато часу піде на вивчення мови сценаріїв.

2. Складнощі при розгортанні систем. Коли в зоні відповідальності знаходиться менше 10 серверів, перенести сценарій на кожну з машин і запустити їх в роботу за розкладом не складе труднощів. Але якщо серверів велика кількість, то можуть виникнути труднощі.

3. Різноманітність використовуваних технологій. Якщо інфраструктура цілком побудована на Linux-системах або на системах сімейства Microsoft Windows, це одне. Але якщо в компанії є і домен-контролер, реалізований на Microsoft Windows Server, файлове сховище на Freebsd і поштовий сервер Postfix на CentOS, то адміністратор змушений розробляти різні сценарії під конкретну ОС, вирішальні одну і ту ж задачу.

Тому, постановка задачі полягає у тому що необхідно виконати:

- аналіз стану проблеми сучасних систем моніторингу серверів та комп'ютерних мереж;

- постановку задачі до якої відносилось аналіз аналогів, виконання робіт що пов'язані з проектуванням, розробкою алгоритма, написання програмного коду та тестування програмного засобу;

- сформувати мету дослідження та визначити основні функції систем моніторингу;

- розглянути загальні характеристики поширених систем моніторингу: CACTI, MRTG, Nagios і Zabbix, HP OpenView і IBM Tivoli

- дослідити поширені класи програмних продуктів які використовуються у системах моніторингу серверів;

- визначити основні задачі проектування системи моніторингу;

- визначити можливості використання системи Prometheus, описані кроки встановлення цієї системи з метою проведення тестування.

Необхідно визначити:

- основні кроки проектування системи моніторингу до яких відносилось визначення предмету дослідження,

- інтерфейс основної частини системи моніторингу та додаткових модулів: веб-сервісу зв'язку, клієнтів, локальних серверів та баз даних;

- алгоритм та діаграми класів;

Необхідно створити:

- програмне оточення з розробки основних метрик системи моніторингу;

- програмний коду системи моніторингу серверів.

Необхідно виконати:

- тестування системи моніторингу за допомогою Grafana;

- тестування файлів конфігурацій системи моніторингу.

1.4 Висновки

В цьому розділі виконано аналіз стану проблеми сучасних систем моніторингу серверів та комп'ютерних мереж. Розглянуто пасивний та активний моніторинг та призначення кожного з них. Наведені приклади роботи з SNMP протоколом який поширено використовується у системах моніторингу.

Розглянуті загальні характеристики поширених систем моніторингу: CACTI, MRTG, Nagios і Zabbix, HP OpenView і IBM Tivoli. Наголошено, на що слід звертати увагу під час вибору системи моніторингу.

Виконано порівняльний аналіз аналогів: Nagios, Zabbix, Hyperic, SolarWinds, ManageEngine OpManager, HP Operations Manager. Розглянуті основні вимоги щодо побудови системи моніторингу до основних з них відносились: відносяться такі: мультиплатформеність, швидкість обробки

інформації, розподіленість в отриманні даних, безпека інформації, використання шаблонів та візуалізація даних.

Визначено, що рішення задач візуалізації полученених даних може бути забезпечено за рахунок: побудови звітів, створення графіків і діаграм, побудови мережевих карт.

Виконано постановка задачі до якої відносилось аналіз аналогів, виконання робіт що пов'язані з проектуванням, розробкою алгоритма, написання програмного коду та тестування програмного засобу.

2 ВАРІАТИВНЕ МОДЕЛЮВАННЯ СИСТЕМИ МОНІТОРИНГУ ОБ'ЄКТІВ ІТ-ІНФРАСТРУКТУРИ

2.1 Методи та інструментарій моделювання процесів моніторингу

Сутність проектування системи моніторингу полягає у створенні функціональної моделі її роботи чи в плануванні всього технологічного процесу отримання інформації про стан об'єкта від постановки завдань до видачі інформації споживачеві для прийняття рішень. Всі етапи функціонування системи моніторингу тісно пов'язані між собою, недостатня увага до розроблення будь-якого з них може стати причиною різкого зниження цінності всієї інформації.

Рисунок 2.1 – Технологія побудови системи моніторингу

Моніторинг серверів комп'ютерної мережі є однією з найважливіших задач у проектуванні, необхідних для організації повноцінного управління технологічними процесами підприємства. Процес виявлення самих пошкоджень та формування комплексу заходів може зайняти значний час і істотно вплинути на функціонування системи автоматизації підприємства в цілому.

Моніторинг працездатності серверів комп'ютерної мережі – це робота системи, яка виконує постійний нагляд за обчислювальною мережею в пошуках повільних або пошкоджених систем. Під час виявлення таких недоліків повідомляє про них адміністратору мережі за допомогою спеціальних засобів оповіщення.

Серед поширених класів програмних продуктів які використовуються у системах моніторингу серверів є наступні (рис. 2.1):

- засоби управління системою;
- системи управління мережею;
- вбудовані системи діагностики і управління;
- аналізатори протоколів;
- експертні системи;
- багатофункціональні пристрої аналізу та діагностики.

Метод моніторингу роботи корпоративного серверу можна представити як сукупність дій моніторингу показників роботи корпоративного сервер та IT інфраструктури за такою послідовністю:

1. Визначення загального навантаження серверу;
2. Визначення стану вузлів;
3. Визначення трафіку мережі;
4. Автоматичне або напівавтоматичне відключення портів, пристроїв.
5. Зміна параметрів мостів адресних таблиць, комутаторів та маршрутизаторів.

6. Визначення просторового позиціонування серверів та їх навантаження.

7. Збір інформації аналізаторів протоколів та формування комплексної оцінки трафіку мережі

8. Формування повідомлень щодо змін в мережі для адміністраторів мереж та серверів.

9. Порівняння показників систем управління та формування повідомлення щодо показників систем моніторингу.

Прикладами систем управління можуть служити популярні системи HP OpenView, SunNet Manager, IBM NetView і інші.

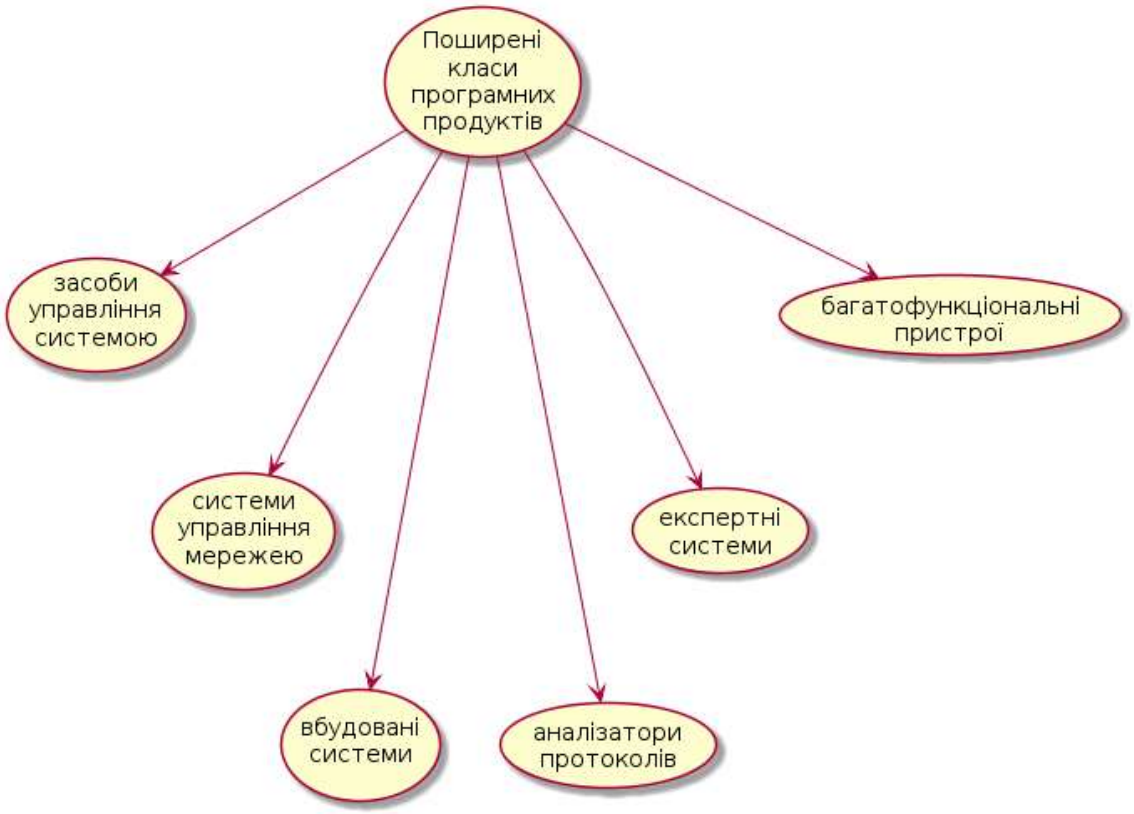

Рисунок 2.2 - Поширені класи програмних продуктів які використовуються у системах моніторингу

Вбудовані системи діагностики і управління (Embedded Systems) - це системи, що виконані у вигляді програмно-апаратних модулів, які встановлюються в комунікаційне обладнання. Також існують модулі у вигляді

програмних додатків, що вбудованих в операційні системи. Вони виконують функції діагностики і управління тільки одним пристроєм. В цьому їх основна відмінність від централізованих систем управління. Прикладом засобів цього класу може служити модуль управління концентратором Distributed 5000, який реалізує функції автосегментацією портів при виявленні пошкоджень, приписування портів внутрішнім сегментам концентратора і деякі інші. Як правило, вбудовані модулі управління додатково виконують роль SNMP-агентів, що поставляють дані про стан пристрою для систем управління.

Аналізатори протоколів (Protocol Analyzers) являють собою програмні або апаратно-програмні системи. На відміну від систем управління, ці системи виконують лише функції моніторингу і аналізу трафіку в мережах. Якісний аналізатор протоколів може захоплювати і декодувати пакети великої кількості що застосовуються у мережах протоколів - зазвичай кілька десятків. Аналізатори протоколів дозволяють встановити деякі логічні умови для захоплення окремих пакетів і виконують повне декодування захоплених пакетів. Тобто вони показують в зручній для фахівця формі вкладеність пакетів протоколів різних рівнів один в одного з розшифровкою змісту окремих полів кожного пакета.

Експертні системи акумулюють людські знання про виявлення причин аномальної роботи мереж і можливі способи приведення мережі в працездатний стан. Експертні системи часто реалізуються у вигляді окремих підсистем різних засобів моніторингу та аналізу мереж: систем управління мережами, аналізаторів протоколів, мережевих аналізаторів. Найпростішим варіантом експертної системи є контекстно-залежна help-система. Більш складні експертні системи являють собою так звані бази знань, що володіють елементами штучного інтелекту.

Багатофункціональні пристрої аналізу та діагностики. В останні роки в зв'язку з повсюдним поширенням обчислювальних мереж виникла необхідність розробки недорогих портативних приладів, які суміщають

функції декількох пристроїв: аналізаторів протоколів, кабельних сканерів і навіть деяких можливостей програмного забезпечення мережевого управління.

Запропонований метод передбачає збір інформації всіх працюючих на сервері та пристроях зовнішніх та вбудованих систем моніторингу з подальшою обробкою даних показників моніторингу з використанням експертних систем, багатофункціональних пристроїв аналізу та діагностики.

2.2 Модель системи моніторингу корпоративних серверів

Проектування системи моніторингу є важливою частиною в розробці програмного засобу. Таке проектування передбачає уточнення того, які ресурси повинні бути досліджені, що має оцінюватися, ким, як і коли. Ці питання повинні вирішуватися на етапі планування або, найпізніше, на самому початку реалізації.

Ґрунтовний аналіз проблеми та її контексту повинні бути виконані в рамках розробки та планування стратегії побудови системи і можуть служити відправною точкою для подальшого моніторингу та оцінки роботи інформаційних ресурсів корпоративних серверів. На рис. 2.3 представлена модель моніторингу.

Рисунок 2.3. – Удосконалена модель моніторингу IT-інфраструктури

Якщо такий аналіз не проведено, важливо провести його на більш пізній стадії і внести необхідні зміни в заплановані заходи.

Системи моніторингу розрізняються за рівнем складності від простого запису на аркуші паперу, в декількох блокнотах або файлах до електронної системи обліку і баз даних. Але найважливіше полягає не в рівні складності системи, а в тому, зібрана інформація, необхідна для прийняття рішень, перевірена вона і використовували її для необхідних змін і доробок.

Запропонована модель моніторингу серверу базується на методі моніторингу, розглянутому в попередньому пункті та гіпотезі використання повного інструментарію моніторингу з подальшою обробкою інформації.

До основних кроків проектування системи моніторингу відносяться такі (рис. 2.4).

1. Предмет дослідження: що необхідно враховувати?

У цьому пункті необхідно визначити мету і охоплення системи моніторингу. Мета може включати в себе такі питання, як звітність перед організаціями, партнерами та користувачами; інформування про стратегічні напрями та внесення змін в разі потреби; інформування про оперативні напрямках і внесення змін в разі потреби; розширення можливостей ключових партнерів.

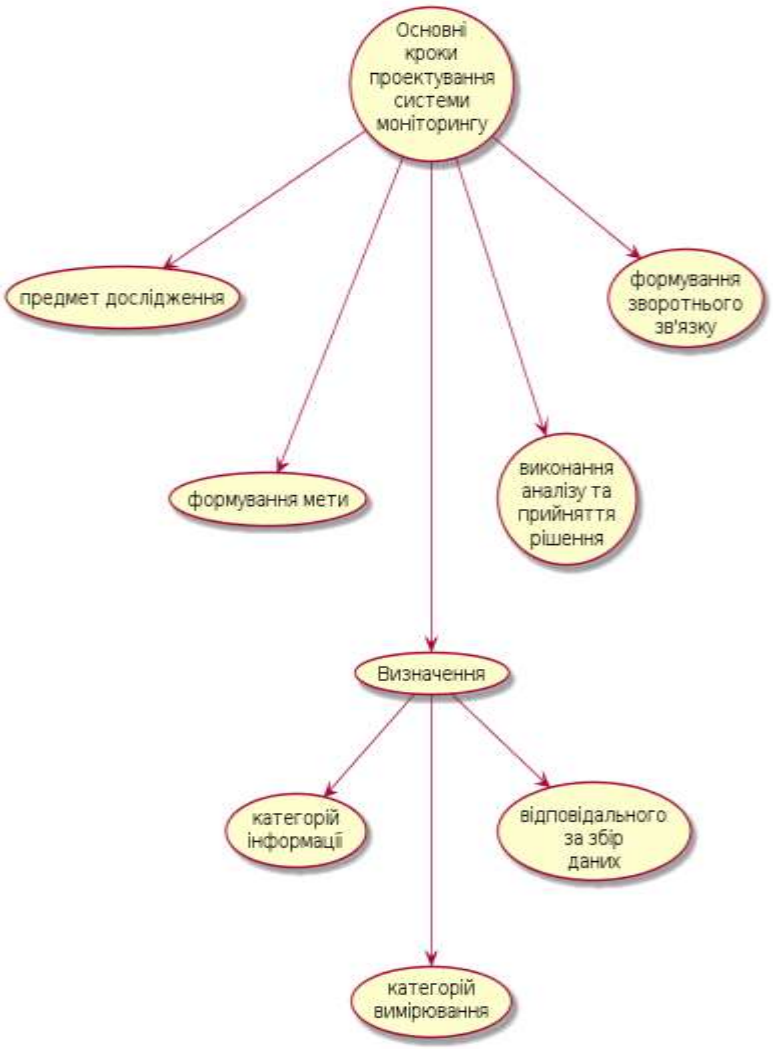

Рисунок 2.2 - Основні кроки проектування системи моніторингу

Кожна мета по-різному впливає на процес проектування системи. Наприклад, якщо мета полягає в розширенні можливостей партнерів, процес буде колективним і більше орієнтованим на навчання. Під охопленням мається на увазі необхідний рівень деталізації, рівень участі партнерів і рівень наявного фінансування (наприклад, необхідно виконати колективну діяльність, але наявні кошти обмежують можливість залучення партнерів.

2. Формування мети: чого необхідно досягти?

У цьому пункті необхідно переглянути концепцію і завдання проекту. Тут потрібно поставити такі питання: в чому сенс даних заходів втручання в роботу інформаційних систем? Яка в основі їх лежить теорія дій? Які прогнози?

3. Визначення категорій інформації: кому і які дані потрібні?

У цьому пункті необхідно оцінити основні потреби сторін в інформації. Серед основних питань: що і коли повинні знати керівники, інші співробітники, користувачі і інші зацікавлені сторони?

4. Визначення категорій вимірювання: на що необхідно звернути увагу при вимірюванні досягнень? Де ми знаходимося сьогодні щодо наших цілей проектування і експлуатації інформаційних систем? Коли і чого ми хочемо досягти?

В цьому випадку, необхідно розробити список критеріїв для вимірювання ефективності та результативності протікання процесів в системі, а також визначити тип даних (кількісні та якісні), які знадобляться для реалізації даного виміру. Наприклад, "кількість користувачів які звертаються на веб сервера" і "причини, за якими їм вдавалося повідомлення про помилки". Якість показників потім буде оцінено відповідно до деякими критеріями.

5. Визначення відповідального за збір даних: хто відповідає за збір даних? В якій послідовності?

У цьому пункті необхідно призначити відповідальних за збір та аналіз даних. Потім, необхідно відповісти на наступні питання: які методи можна

використовувати, щоб гарантувати збір і аналіз вірних даних? Хто буде за це нести відповідальність? Методи можуть бути якісними / кількісними, індивідуальними / груповими, колективними / традиційними. Як різні боки користувачів будуть залучені до цього процесу?

6. Виконання аналізу та прийняття рішення: як ми чинимо по відношенню до цілей? Чому? Чого досягли, і що потрібно зробити?

У цьому пункті необхідно сформувати критичне осмислення заходів і процесів, що відбуваються в інформаційних системах.

Критичне осмислення означає, що повинні задаватися не тільки питання про те «що сталося» і «чому», а й «що це означає» і «що ми будемо з цим робити». Це допомагає у вивченні і управлінні впливом за допомогою засобів моніторингу.

7. Формування зворотного зв'язку: як можна гарантувати, що інформація поширюється систематично і використовується для винесення уроків?

У цьому пункті необхідно розробити процес комунікації і звітів. Необхідно вирішити, хто потрібен для зв'язку з окремими підрозділами та кому звітувати під час процесу моніторингу та оцінки, і як це робити.

Різні партнери і користувачі мають різні потреби в інформації і різні вимоги до звітності. Тому результати оцінки повинні бути представлені в доступному форматі, відповідному для систематичного поширення в рамках організації і далі - для освітніх цілей і наступних действій.В світлі уроків, отриманих в процесі оцінки, визначаються треті сторони, зацікавлені в проекті. Вони стають одержувачами інформації для розширення можливостей застосування відповідних відкриттів проекту.

8. Прогнозування розвитку системи моніторингу: які види потенціалу потрібні і повинні бути посилені?

У цьому пункті необхідно оцінити потенціал і умови для реалізації системи моніторингу. Потрібно чітко уявляти свої потреби в сенсі кадрових ресурсів, матеріального заохочення, структур, процедур і фінансів. Також

потрібно оцінити потенційні ризики в процесі збору даних, реєстрації та повідомлення про них в звітах.

Навчання співробітників методам колективного збору даних допоможе підвищити кадровий потенціал і буде для них стимулом при залученні їх до колективного процес моніторингу та оцінки інформаційних процесів.

Низький рівень підтримки проекту буде мати наслідки для можливостей і умов, необхідних для проведення моніторингу, оцінки роботи інформаційних ресурсів і оцінки наслідків.

2.3 Система моніторингу ігрового серверу

Сучасні вимоги щодо проектування серверів та комп'ютерних мереж вимагають більш точного і гнучкого підходу до систем моніторингу. Від коректної роботи Web-серверів і серверів баз даних може залежати працездатність внутрішньокорпоративних додатків і важливих зовнішніх сервісів для клієнтів. Збої і порушення роботи маршрутизаторів можуть порушувати зв'язок між різними частинами організації і її філіями.

Сервери внутрішньої пошти і мережевих месенджерів, автоматичних оновлень і резервного копіювання, принт-сервери - будь-які з цих елементів можуть страждати від програмних і апаратних збоїв. Завдання системи моніторингу - це попередження, так як перерви в роботі мережі в цілому впливають на авторитет організації, комерційні організації втрачають заробіток при непрацездатності комп'ютерної мережі. Тому більшість комерційних організацій використовують різноманітні засоби і програмні продукти для власних систем моніторингу.

Оптимальна програма для моніторингу навантаження на сервер повинна відображати звіти і графіки таких показників:

- завантаження процесора;
- мережевий трафік;
- кількість вільного місця на дисках;
- запущені процеси;

- витрата споживаної пам'яті і т.д.

У разі виявлення відхилень від нормальних показників додаток повинен повідомити про неполадки і запропонувати виконання дій, спрямованих на коригування несправності. Ви дуже просто зможете відстежити і з'ясувати причину виникнення підвищеного навантаження, адже спеціальний софт зберігає записи про кожну подію моніторингу.

Під час проектування інструментів моніторингу систем що досліджують потужні інфраструктури необхідно враховувати ряд факторів. Перш за все - це оцінити відповідність функціоналу необхідним технічним і бізнес-вимогам. Потім, потрібно розглянути які важливі на етапі проектування систем моніторингу:

-особливості розгортання та супроводу систем моніторингу, щоб підібрати інструмент;

- провести аналіз інформаційних та технологічних ресурсів;

-визначити рівень компетенції команди IT-спеціалістів, що будуть обслуговувати систему моніторингу;

-обчислити загальну вартість системи моніторингу що буде впроваджуватись.

Проектування інструментів моніторингу систем передбачає такі категорії дослідження (рис 2.3):

- функціональність системи;
- інтерфейс користувача;
- система повідомлень;
- інформаційна структура;
- інсталяція та супровід системи.

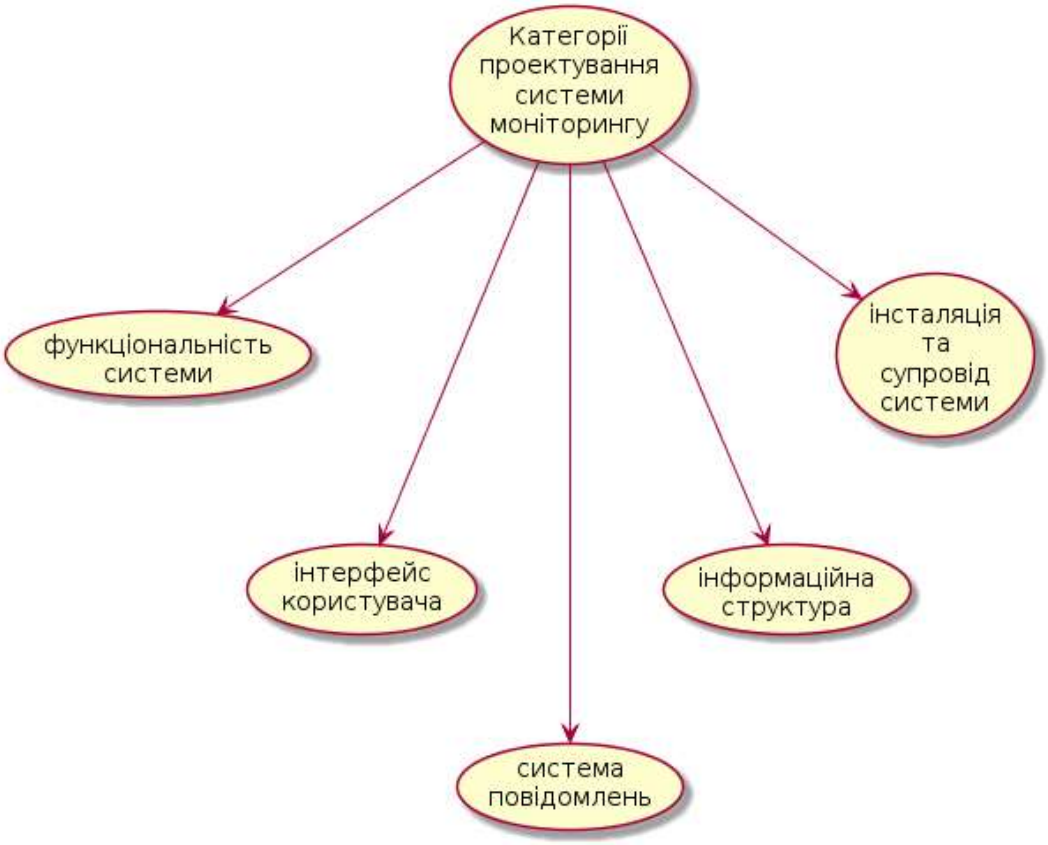

Рисунок 2.3 - Важливі категорії проектування системи моніторингу

Для проектування майбутньої функціональності потрібно визначити потреби різних користувачів: розробників, експлуатаційних персоналу та інших. Наприклад, відповідальний за прийняття бізнес-рішень може бути зацікавлений в звітах про виконання договорів про рівень обслуговування. Такі документи, в свою чергу, можуть виявитися цінними і для технічних спеціалістів у якості допоміжних засобів при виявленні проблем з продуктивністю інформаційних ресурсів. Оцінити необхідно найрізноманітніші аспекти. Інструмент моніторингу систем повинен:

- підтримувати моніторинг як серверної, так і інтерфейсної частини середовища;

- бути здатним розпізнавати широке коло проблем (від зниження швидкості та аварійних завершень до витоків пам'яті);

- мати розвинуту систему ведення протоколів роботи програмного засобу;

- мати розвинуту систему безпеки.

Під час проектування інструментів моніторингу велике значення має майбутній інтерфейс користувача. Інструменти моніторингу інфраструктури існують вже давно, однак їх користувацькі інтерфейси не завжди спішать за сучасними вимогами та тенденціями. Тому, під час проектування системи необхідно оцінити, чи задовольняє інтерфейс користувача потребам роботи фахівців. В залежності від спеціальності користувача та його посадових обов'язків, можливо, буде обов'язковим наявність веб-інтерфейсу або мобільного додатка - для універсальної функціональності.

Повідомлення, інтеграція з службою підтримки та можливість автоматизації є важливими аспектами під час проектування системи моніторингу. Відомо, що призначення системи моніторингу полягає в тому, щоб допомогти якомога швидше реагувати на проблеми, наприклад, на погіршення якості обслуговування периферійних пристроїв або інформаційних (обчислювальних) ресурсів. У цьому випадку, можливо, пріоритетною вимогою під час проектування буде налаштування системи видачі попереджень.

Тому, під час вибору інструментарію слід звернути увагу на наступні фактори:

- підтримка різних способів повідомлення (по SMS, електронною поштою, за допомогою будь-яких скриптів;

- об'єм конфігураційних файлів, який буде необхідний для програмного середовища;

- засоби підтримки операційних систем;

- можливість інтеграції з системою підтримки користувачів.

Під час проектування необхідно визначити, що за набором відомостей про інфраструктуру підприємства можливо автоматизувати виконання різних завдань під час настання різних подій. Це надасть більше контролю під час появи пошкоджень.

Розвертання та супровід також являють важливі чинники під час проектування моніторингу систем. Перш всього, спосіб розгортання інструментів повинен узгоджуватися з корпоративними політиками. Крім того, обраний інструмент повинен підтримувати різні технології програмування, бути сумісним з інфраструктурою підприємства та відповідати компетенції IT-спеціалістів.

Необхідно також провести оцінку методів збору показників з точки зору можливості вилучення цінних відомостей під час роботи серверів. Необхідно врахувати, що вибір способу моніторингу продуктивності серверів залежить від того, що є джерелом діагностичної інформації. Наприклад, вона може вийматись з коду, з журналів операцій (протоколів) і мережевого обладнання.

Також, під час проектування необхідно оцінити вартість інсталяцій та супроводів: будь-які інструменти моніторингу доведеться адаптувати до конкретного середовища. Тому, роль процесів їх встановлення та налаштування в проектах важко переоцінити. При цьому, необхідно звернути увагу на простоту розгортання, на можливість автоматичного розпізнавання топології прикладних програм і співвіднести це з наявними навиками та ресурсами.

Під час встановлення інструментарію потрібно орієнтуватися на його швидке повернення коштів. Для оцінки загальної вартості системи моніторингу потрібно, наприклад, провести порівняння подібної версії системи з альтернативою.

2.4. Методи збору та обробки інформації моніторингу серверів

Існує кілька способів отримання моніторингових даних. Одними з найпоширеніших є зовнішні бази даних, які наповнюються даними за допомогою інших систем, і підключення до сервера, наприклад, за протоколом SSH з подальшим запуском моніторингових утиліт. У задачі, для вирішення якої розробляється ця система - моніторинг в режимі реального часу, використовується спосіб з SSH-підключенням. Варто відзначити, що крім

прямого SSH-підключення необхідно забезпечити можливість збору і відправки даних через SSH-клієнт, який часто є єдино можливим способом підключення.

Щоб забезпечити підтримку декількох варіантів підключення, джерело даних спроектований як адаптер між зовнішньою системою, у якій можна отримати моніторингові дані за запитом, і моніторинговою системою.

Після збору даних проводиться їх обробка, в ході якої дані в форматі простого тексту перетворюються в формат, який приймає споживачами даних - веб-інтерфейсом і Системою зберігання і аналізу журналів. В обох випадках це JSON-формати, в яких моніторингові дані згруповані за необхідне для використання споживачем даних чином і додана необхідна службова інформація.

Для того, щоб забезпечити максимальну розширюваність системи щодо підтримки всіляких моніторингових утиліт і реалізувати можливість додавати підтримку нових утиліт без необхідності модифікувати і перекомпілювати вихідний код системи для моніторингу, було вибрано подання утилітних розширень в вигляді виконуваних скриптів. Мовою написання скриптів був визначений Java. Важливою причиною використання цієї мови для розробки послужило те, що вона поставляється разом з віртуальною машиною Java, що дозволяє використовувати його без необхідності установки інтерпретатора і легко запускати виконання скриптів.

Основне завдання утилітного розширення (скрипта) складається в перетворенні надходить на вхід моніторингової інформації в один із специфікованих форматів, найчастіше - в JSON певної структури. Основні залишають скрипта - службова інформація і головна функція, яка і повинна здійснити перетворення, будучи викликаною з Java. У випадку з написанням утилітного розширення для побудови графіків за даними однієї з моніторингових утиліт, ця функція буде містити в собі фільтрацію даних і налаштуавння їх подальшого відображення.

2.5 Висновки

В цьому розділі визначені основні задачі проектування системи моніторингу, метод та моделі моніторингу. Визначено, що важливою частиною моніторингу серверів є спостерігання за їх працездатністю. Під час виявлення будь-яких недоліків система повинна повідомляти про них адміністратору мережі за допомогою спеціальних засобів оповіщення. Також визначені поширені класи програмних продуктів які використовуються у системах моніторингу серверів та є такі: засоби управління системою, системи управління мережею, вбудовані системи діагностики і управління, аналізатори протоколів, експертні системи та багатофункціональні пристрої аналізу та діагностики.

Під час проектування інструментів моніторингу систем визначено, що сучасні вимоги щодо проектування серверів та комп'ютерних мереж вимагають більш точного і гнучкого підходу до систем моніторингу. Щоб підібрати інструмент, проведення аналізу інформаційних та технологічних ресурсів, визначення рівня компетенції команди IT-спеціалістів, що будуть обслуговувати систему моніторингу, обчислення загальної вартості системи моніторингу що буде впроваджуватись.

Проектування важливих інструментів систем моніторингу передбачала дослідження таких категорій дослідження як: функціональність системи, інтерфейсу користувача, системи повідомлень, інформаційної структури, інсталяції та супроводу системи.

3 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ДЛЯ МОНІТОРИНГУ КОРПОРАТИВНИХ СЕРВЕРІВ

3.1 Створення програмного забезпечення з розробки метрик моніторингу

Сучасні підприємства різних галузей є в більшості випадків складними автоматизованими технологічними комплексами з розвиненими автоматизованими системами управління технологічними процесами. Ефективність діяльності таких компаній безпосередньо залежить від надійності і ефективності функціонування інформаційних систем в цілому, його окремих підсистем, комплексів і компонентів, злагодженості роботи операторів, диспетчерів, інженерно-технічного персоналу, коректності їх дій при прийнятті рішень, виконання та контроль виконання рішень.

Тому однією з актуальних проблем для підприємств з розвиненими автоматизованими системами є формування системи відстеження (моніторингу) в режимі реального часу.

У різних галузях промисловості поняття моніторинг систем автоматизації тобто відстеження стану і ситуацій існує і впроваджується досить давно. Тому, призначення системи моніторингу систем управління різними технологічними та інформаційними об'єктами - автоматизація процесів контролю в режимі реального часу технічного стану елементів і технічних (або інформаційних) ресурсів.

В основному, об'єктами моніторингу є апаратні і програмні засоби інформаційних систем, встановлені в різних організаціях. Як правило, система моніторингу є складовою частиною загальної системи управління.

Тому, в першу чергу в системах моніторингу реалізовані наступні функції (рис. 3.1):

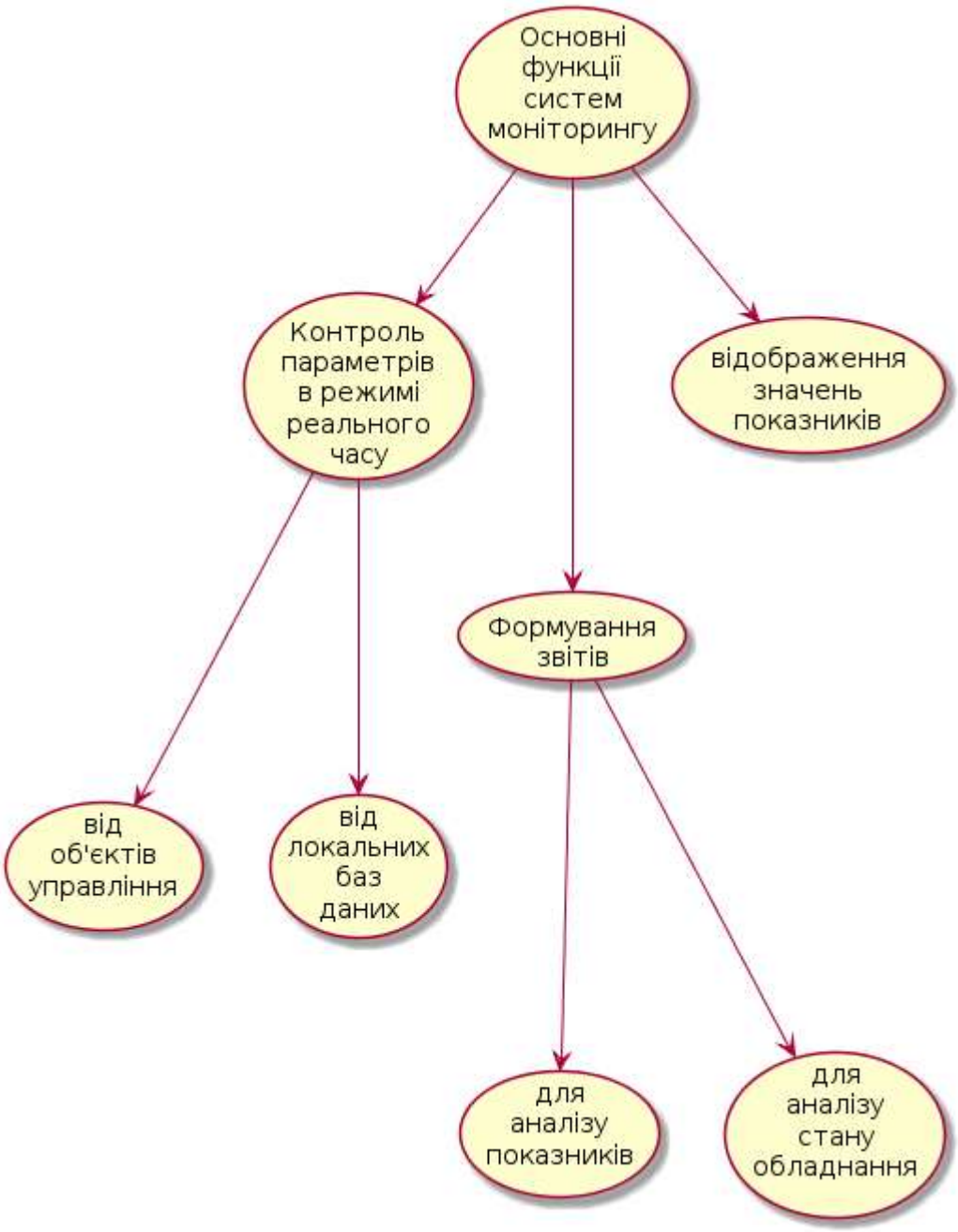

Рисунок 3.1 - Основні функції систем моніторингу

- контроль параметрів в режимі реального часу, що надходять від об'єктів управління;

- контроль параметрів в режимі реального часу локальні баз даних систем управління організації;

- відображення значень показників, що характеризують ситуацію інформаційних ресурсів на технологічних об'єктах;

- формування звітів для аналізу показників, що надходять з інформаційних ресурсів технологічних об'єктах;

- формування звітів для аналізу стану обладнання засобів вимірювання і автоматизації.

Часто користувачами систем моніторингу є фахівці інформаційних або технічних відділів автоматизації виробництва, відповідальні за працездатність засобів вимірювання і автоматики.

Як правило, система моніторингу є розвиненою територіально розподіленою системою управління. Для її контролю стану потрібна велика кількість різних показників, які налічують близько 10000 одиниць. Якщо прийняти, що перевірка кожного показника виробляється приблизно 1 раз в 5 хв, то для реалізації моніторингу потужності одного персонального комп'ютера явно недостатньо. Тому мова може йти про побудову розподіленої програмно- інформаційної системи моніторингу.

Перевагою розподіленої системи моніторингу є можливість контролю великого числа об'єктів, без пред'явлення особливих вимог до продуктивності комп'ютера, на якому здійснюється моніторинг. При цьому сама система моніторингу стає живучою за рахунок її реконфігурації при отказеспеціалізірованних комп'ютерів моніторингу або клієнтських комп'ютерів і масштабується в результаті нескладного процесу додавання в систему нових спеціалізованих комп'ютерів моніторингу та клієнтів.

3.2 Моделювання та проектування архітектури системи моніторингу

З урахуванням вимог і аналізу існуючих рішень, була розроблена архітектура системи для моніторингу серверів в режимі реального часу. Проект такої архітектури представлений на рисунку 3.2

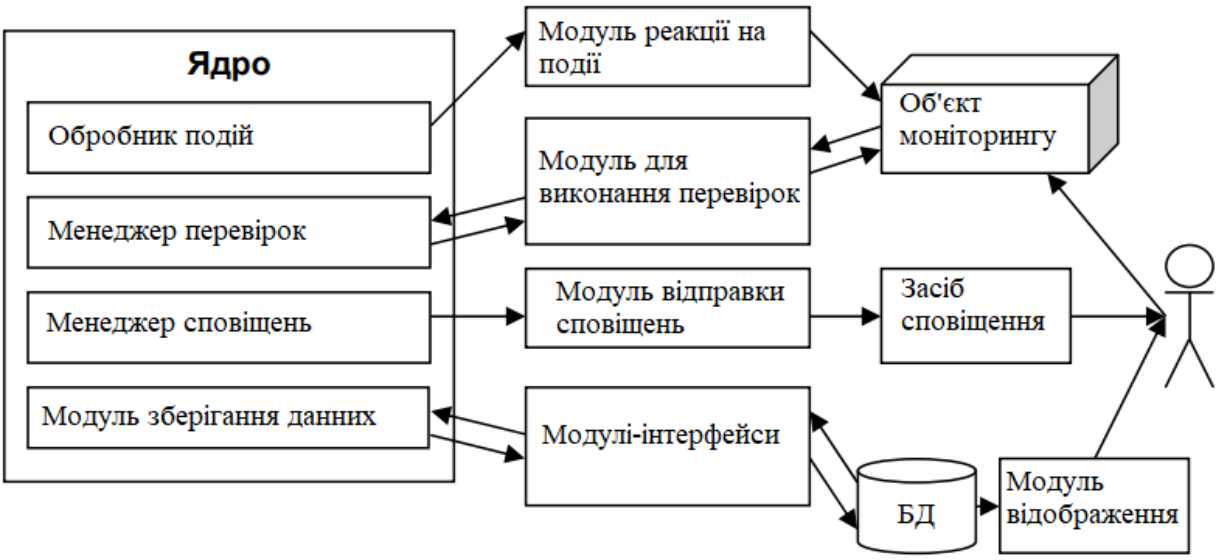

Рисунок 3.2 – Архітектура системи моніторингу

Ядро системи складають компоненти, що реалізують її основні внутрішні механізми (отримання і накопичення даних про об'єкти моніторингу довільної природи, діагностика позаштатних ситуацій і оповіщення про них, прийняття рішення про належної реакції на виявлену нештатну ситуацію). Компоненти ядра здійснюють виклик необхідних модулів, що реалізують рішення приватних завдань, відповідних області застосування системи (моніторинг конкретних типів об'єктів і служб, взаємодія з системами зберігання даних, відображення стану системи в різних форматах і т.д.). Відзначимо, що функції підсистеми виведення цілком винесені з ядра як не потребують взаємодії з самими процесами моніторингу, а взаємодіє тільки з поставляються системою зберігання накопиченими даними (як про об'єкти моніторингу, так і про роботу самої системи).

В системі моніторингу спеціалізованих комп'ютерів які збирають відомості щодо інформаційних ресурсів технологічних об'єктів може може бути кілька. Кожен клієнт також може бути пов'язаний з декількома спеціалізованими комп'ютерами моніторингу.

З технічної точки зору в систему моніторингу (рис. 3.2) об'єднані кілька територіально розподілених серверів баз даних і серверів з локальних

комп'ютерних мереж. На серверах з локальних комп'ютерних мереж контролюється і зберігається історія показників невеликої групи технологічних об'єктів, охоплених даною системою моніторингу.

Для зберігання довідкової інформації використовується центральна база даних системи на основі корпоративної інформаційної системи, в якій для цього були створені необхідні структурні елементи.

З серверів локальних комп'ютерних мереж в систему надходять дані реального часу.

Список контрольованих параметрів і частота моніторингу налаштовуються адміністратором системи моніторингу. З бази даних локальних комп'ютерних мереж в систему надходять дані, накопичені за певний проміжок часу. Результати моніторингу та дані про квотування повідомлень записуються в центральну базу даних.

Зв'язок між клієнтами і серверами реалізована на основі технології веб-сервісів. Існує кілька сервісів підтримки системи моніторингу. До основних з них відносяться такі сервіси як (рис. 3.2):

- служба "Веб-сервіс зв'язку";

- служба "Порівняння даних роботи локальних серверів з вихідними даними";

- служба "Порівняння даних роботи баз даних з вихідними даними";

- служба "Веб-сервіс клієнта".

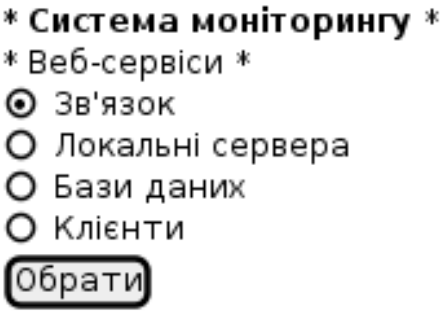

Рисунок 3.2 - Інтерфейс основної частини системи моніторингу

Служба "Веб-сервіс зв'язку" передає повідомлення кільком клієнтам і приймає повідомлення від декількох клієнтів (рис. 3.3). Служба "Веб-сервіс

зв'язку" встановлюється на спеціалізованих комп'ютерах, на яких проводиться процес моніторингу. Служба реєструє ір-адреси клієнтів, яким будуть розсилатися повідомлення, що повідомляють про аварійну і передаварійніх ситуаціях.

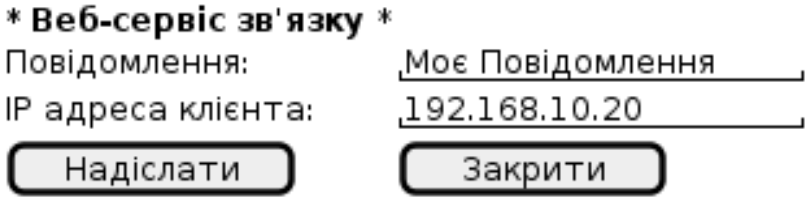

Рисунок 3.3 - Інтерфейс підсистеми веб-сервісу зв’язку системи моніторингу

Служба "Порівняння даних роботи локальних серверів з вихідними даними" встановлюється на спеціалізованих комп'ютерах моніторингу (рис. 3.3). Служба виконує функцію порівняння з допустимими значеннями вихідних даних, отриманих від локальних серверів.

Служба "Порівняння роботи баз даних з вихідними даними" встановлюється на спеціалізованих комп'ютерах моніторингу (рис. 3.4). Служба виконує функції порівняння з вихідних даних роботи баз даних з допустимими значеннями вихідних даних (рис. 3.5).

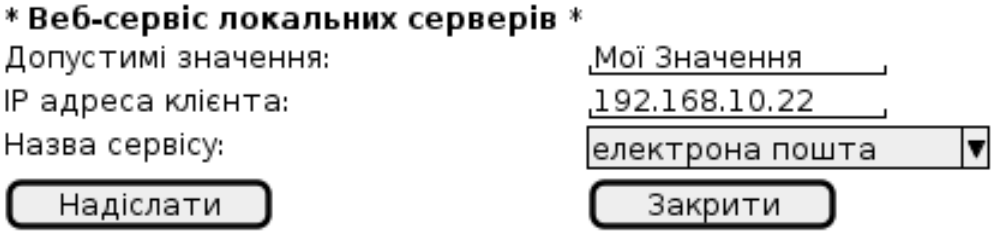

Рисунок 3.4 - Інтерфейс підсистеми веб-сервісу зв’язку системи моніторингу

Служба "Веб-сервіс клієнта" (рис. 3.6) встановлюється на комп'ютерах користувачів системи моніторингу та призначена для реєстрації користувача в службі "Веб-сервіс зв'язку" і отримання повідомляючих повідомлень.

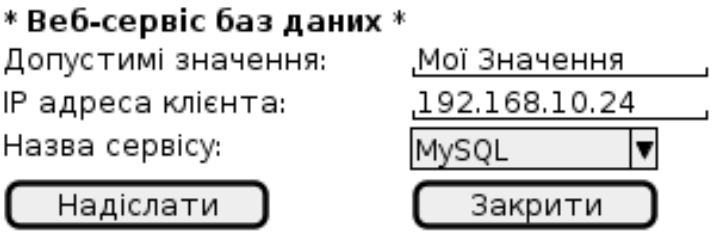

Рисунок 3.5 - Інтерфейс підсистеми веб-сервісу баз даних системи моніторингу

На екрані користувача робота цієї служби відображається в правому нижньому кутку екрану (системному треї (tray)) на панелі завдань.

Рисунок 3.6 - Інтерфейс реєстрації клієнтів у системи моніторингу

Програмне забезпечення системи моніторингу встановлюється на комп'ютерах користувачів підсистеми "Моніторинг" та реалізує наступні функції:

- контроль стану системи в режимі реального часу;
- відображення контрольованого об'єкта на інтерактивній схемі-карті;
- контроль стану технологічного процесу;
- контроль показників (відображення значень показників в табличному і графічному вигляді);
- аналіз стану парку засобів вимірювальної техніки і автоматики;
- аналіз робіт з технічного обслуговування;
- ведення архіву документації;
- адміністрування підсистеми.

Служби системи моніторингу повинні функціонувати в безперервному режимі, забезпечуючи контроль стану управління об'єктами в реальному масштабі часу. При виникненні аварійної (передаварійній) ситуації на якомусь

об'єкті служби порівняння з вихідними даними, повідомлення передають в служби "Веб-сервіс зв'язку" з урахуванням ідентифікації (об'єкт і параметр).

Далі служба "Веб-сервіс зв'язку" розсилає повідомляючі повідомлення іншим службам "Веб-сервіс клієнта", які зареєстровані на поточній платформі "Веб-сервісі зв'язку".

Реєстрація проводиться в момент запуску служби "Веб-сервіс клієнта" відповідно до областями доступу користувачів до системи моніторингу.

При отриманні повідомлення користувач може переглянути дату або час його виникнення і найменування об'єкта, на якому виявлено аварійна (передаварійна) ситуація в "Журналі повідомлень" і квітіровать його. Під час квітірованія користувач може занести в базу даних коментарі до даного повідомлення. Користувач може подивитися розташування об'єкта на інтерактивній карті-схемі, перейти до технологічними схемами об'єкта, проаналізувати ситуацію за значеннями параметрів, що контролюються, подивитися перелік засобів вимірювальної техніки, встановлених на контрольованому об'єкті, відомості про виконані роботи.

3.3 Програмна реалізація методу моніторингу кількості ігрових об'єктів

В цьому розділі опишемо тільки основні ділянки програмного коду, що використовує програмний засіб з моніторингу Програмна реалізація методу моніторингу кількості ігрових об'єктів. Повний лістинг програмного засобу розміщується у Додатку В.

Написання коду основного модуля з моніторингу інформаційних ресурсів Програмна реалізація методу моніторингу кількості ігрових об'єктів створюється після розробки алгоритму роботи основної програми що написаний за міжнародними стандартами UML нотацій (рис. 3.7).

Дані для моніторингу кількості ігрових об’єктів записуються в екземпляр List за допомогою методу add класу Builder

```
        // objects storage
        MetricClient.current().register(() -> {
        Builder<Metric> builder = ImmutableList.builder();
        List<GameObjectsStorage.Stats> stats =
GameObjectsStorage.getInnerStats();
        stats.forEach(s -> {
        builder.add(metric("object_storage." + s.name.toLowerCase())
        .field("size", s.size)
        .field("capacity", s.capacity)
        .field("init_capacity", s.initCapacity)
        .build());});
        return builder.build();});
        Після цього перевіряється їхній статус
        builder.add(metric("world")
                        .field("objects", stats.objectCount)
                        .field("npcs", stats.npcCount)
                        .field("chars", stats.charCount)
                        .field("items_char", stats.itemsChar)
                        .field("items_charwh", stats.itemsCharWh)
                        .field("items_pet", stats.itemsPet)
                        .build());
```

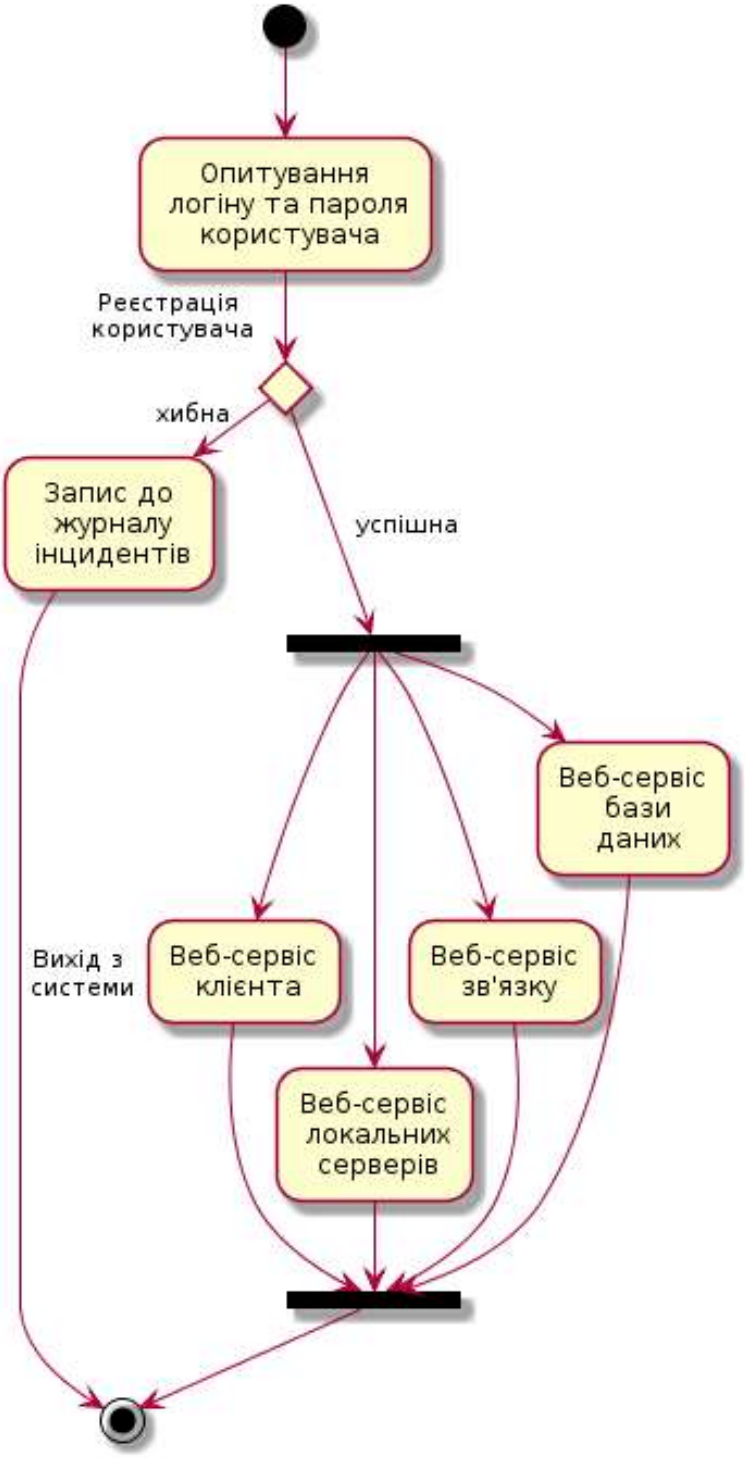

Рисунок 3.7 - Алгоритм роботи системи моніторингу що створений за UML стандартами

Потім, опишемо конструктор основного класу системи моніторингу:

```
public DataBaseService(String aCode, String aLimit, String aDescription)
{
  cachedHashCode = -1;
```

```
    canSetCode = true;
    code = aCode;
    limit = aLimit;
    description = aDescription;
  }
```

3.4 Програмний модуль методу збору метрик в ThreadPoolExecutor

Даний модуль дозволяє сформувати систему показників моніторингу, що здійснюють спостереження за показниками поточних активних процесів; завдань, що завершені та процесів, які прогнозуються до виконання з можливістю визначення класу реалізації та кількості часу для виконання.

```
public List<Metric> get(ThreadPoolExecutor executor) {
    ImmutableList.Builder<Metric> builder = builder();
    //@formatter:off
    builder.add(from(submitted,  "executor.submitted",  b -> b.tag("name", name)));
    builder.add(from(running,  "executor.running",  b -> b.tag("name", name)));
    builder.add(from(completed,  "executor.completed",  b -> b.tag("name", name)));
    builder.add(from(duration,  "executor.duration",  b -> b.tag("name", name)));
    builder.add(from(wait,  "executor.wait",  b -> b.tag("name", name)));
    //@formatter:on

    builder.add(
        Metric.metric("executor.queue")
            .tag("name", name)
```

```
                .field("value", executor.getQueue().size())
                .build()
        );
builder.add(
            Metric.metric("executor._completed")
                .tag("name", name)
                .field("value", executor.getCompletedTaskCount())
                .build()
        );
```

Також було визначено діаграму класів (рис. 3.8).

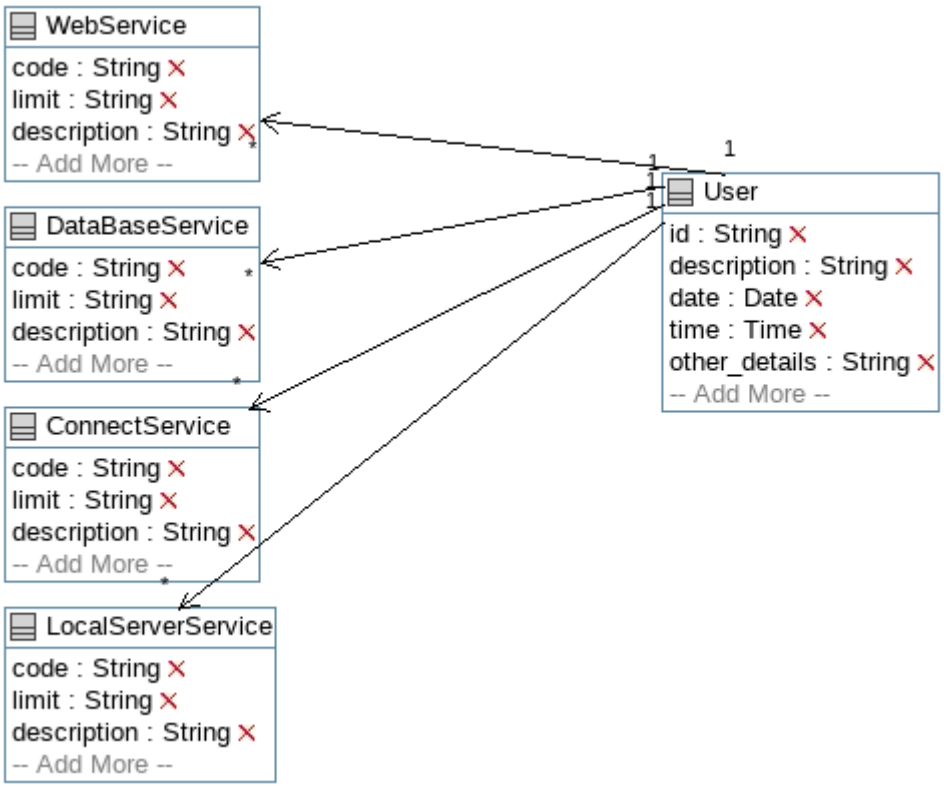

Рисунок 3.8 - Діаграма класів основного модуля системи моніторингу

Після визначення діаграми класів основного модуля системи моніторингу визначимо стандартну назву бібліотеки основного модуля який написаний на мові високого рівня Java:

package monitoringSystem;

Далі визначаємо змінних основного класу системи моніторингу:

```
public class DataBaseService{
  private String code;
  private String limit;
  private String description;
  private int cachedHashCode;
  private boolean canSetCode;
```

3.5 Приклад візуалізації процесів моніторингу кількості онлайн-користувачів

В цьому розділі визначаємо основні методи системи моніторингу кількості онлайн користувачів:

```
  public boolean setCode(String aCode)
  {
    boolean wasSet = false;
    if (!canSetCode) { return false; }
    code = aCode;
    wasSet = true;
    return wasSet;
  }
  public boolean setLimit(String aLimit)
  {
    boolean wasSet = false;
    limit = aLimit;
    wasSet = true;
    return wasSet;
  }
  public boolean setDescription(String aDescription)
```

```
{
  boolean wasSet = false;
  description = aDescription;
  wasSet = true;
  return wasSet;
}
public String getCode()
{
  return code;
}
public String getLimit()
{
  return limit;
}
public String getDescription()
{
  return description;
}
public boolean equals(Object obj)
{
  if (obj == null) { return false; }
  if (!getClass().equals(obj.getClass())) { return false; }
  DataBaseService compareTo = (DataBaseService)obj;
  if (getCode() == null && compareTo.getCode() != null)
  {
    return false;
  }
  else if (getCode() != null && !getCode().equals(compareTo.getCode()))
  {
```

```
    return false;
   }
   return true;
  }
  public int hashCode()
  {
   if (cachedHashCode != -1)
   {
    return cachedHashCode;
   }
   cachedHashCode = 17;
   if (getCode() != null)
   {
    cachedHashCode = cachedHashCode * 23 + getCode().hashCode();
   }
   else
   {
    cachedHashCode = cachedHashCode * 23;
   }
   canSetCode = false;
   return cachedHashCode;
  }
  public void delete()
  {}
```

Далі опишемо метод, що збирає дані з серверу про кількість всіх онлайн користувачів та кількість оффлайн-торговців, що теж відносяться до онлайн користувачів, при чому онлайн «живих» користувачів є різницею між кількістю всіх онлайн користувачів та кількістю оффлайн-торговців:

```
// online
        MetricClient.current().register(() -> {
            Builder<Metric> builder = ImmutableList.builder();
   int onlineCount = playerCount - offtradeCount;
            final int playerCount = GameObjectsStorage.getAllRealPlayersCount();
            final int offtradeCount = GameObjectsStorage.getAllOfflineCount();
            builder.add(metric("online")
```

```
                .field("real", playerCount)
                .field("offtrade", offtradeCount)
                .build());
        return builder.build();
    });
```

Таким чином, в даному розділі розроблено клас діаграм згідно вимог щодо побудови системи моніторингу на написано код програмного додатку.

3.6 Висновки

В даному розділі описані організація і основні функції систем моніторингу та визначено, що однією з актуальних проблем для підприємств з розвиненими автоматизованими системами є формування системи відстеження (моніторингу) в режимі реального часу.

Визначені основні функції систем моніторингу до яких належали: контроль в режимі реального часу параметрів, що надходять від об'єктів управління, контроль параметрів в режимі реального часу локальні баз даних систем управління організації, відображення значень показників, що характеризують ситуацію інформаційних ресурсів на технологічних об'єктах, формування звітів для аналізу показників, що надходять з інформаційних ресурсів технологічних об'єктах, формування звітів для аналізу стану обладнання засобів вимірювання і автоматизації.

Розроблено інтерфейс основної частини системи моніторингу та додаткових модулів: веб-сервісу зв'язку, клієнтів, локальних серверів, та баз даних.

Розроблено алгоритм, діаграми класів та програмного коду.

4 ТЕСТУВАННЯ РОБОТИ СИСТЕМИ

4.1 Створення програмного оточення з розробки метрик

Створення програмного оточення для роботи програмного засобу передбачає якісну роботу системи моніторингу та складається з таких основних функцій (рис. 4.1):

- надання сховища метрикам;

- візуалізація та оповіщення різних станів роботи дослідних об'єктів для фізичних серверів, віртуальних машин, контейнерів або сервісів;

- забезпечення роботи різних систем взаємодії з графічними об'єктами: Graphite, Prometheus, Elastic Beats, тощо.

Рисунок 4.1 - Основні функції програмного оточення

Головними чинниками розробки систем моніторингу є створення та налаштування метрик. Метрики являють собою кількісні та якісні параметри роботи інформаційних та технологічних процесів.

Проблемами під час роботи з системою моніторингу є підтримка багатовимірних метрик і мова запитів до них. Можливість використання одніє і тієї ж мови для різних графічних зображень і повідомлення значно спрощує процес моніторингу дослідних об'єктів. Рішення деякий проблем бере на себе система Prometheus яка здійснює тестування за методом як чорного, так і білого ящика. Це означає, що існує можливість тестування цілої інфраструктуру організації та контролювати внутрішній стан роботи власних додатків.

Використання системи Prometheus надає можливість:

- розгорнути цілий стек програмних засобів системи моніторингу з використанням контейнерів;

- легко створювати додаткові програмні структури для розподілених систем і інфраструктур організації;

- виконувати масштабований збір даних, що не залежать від розподіленого сховища даних;

- використовувати гнучку системи пошуку послуг (вбудовану підтримку для контейнерів Kubernetes, Consul, EC2, Azure);

- виконувати цільовий експорт даних для таких сервісів як HAProxy, MySQL, PostgreSQL, Memcached, Redis і інші.

Набір метрик системи Prometheus величезний. Це означає, що можна знайти метричні експортери для цілого ряду систем, починаючи від баз даних, систем управління повідомленнями MQ, прикладних сервісів HTTP до систем, які пов'язані з апаратними засобами, таких як IoT або IPMI.

Тестування за методом білого ящика системи Prometheus також має відмінне покриття. Це означає, що існують цілі клієнтські бібліотеки Prometheus для Go, Java, Python, Ruby, .NET, PHP та інших мов програмування.

Роботу с системи Prometheus необхідно починати з вивчення структури системи у репозиторії dockprom на GitHub. Для цього можна використовувати dockprom в якості початкової точки моніторингового рішення. Це дозволить

за допомогою однієї команди керувати цілим стеком: Prometheus, Grafana, cAdvisor, NodeExporter і AlertManager.

Для встановлення системи необхідно скопіювати репозиторій dockprom на докер-хост, а потім перейти в директорію dockprom і запустити compose up:

$ git clone https://github.com/stefanprodan/dockprom

$ cd dockprom

$ docker-compose up -d

Це забезпечить роботу з контейнерами:

- Prometheus - метрична база даних, що знаходиться за адресою http://<host-ip>:9090;

- AlertManager - управління оповіщенням, що знаходиться за адресою http://<host-ip>:9093;

- Grafana - візуалізація метрик, що знаходиться за адресою http://<host-ip> 3000;

- NodeExporter - відповідальна система за складання хостових метрик до основного вузла;

- cAdvisor - відповідальна система за складання метрик до контейнеру вузла.

В цьому випадку, якщо Gafana підтримує аутентифікацію користувачів, то сервіси Prometheus і AlertManager не мають такої функції. При наявності базової аутентифікації для Prometheus і AlertManager можливо вилучити відображення портів з файлу контейнера docker-compose і використовувати веб сервер Nginx як зворотний проксі-сервер.

4.2 Тестування системи за допомогою Grafana

Grafana являє собою платформу для аналітики необхідних показників системи моніторингу яка дозволяє запитувати, візуалізувати, сповіщати та розуміти зібрані показники незалежно від того, де вони зберігаються.

Для використання Grafana необхідно перейти на адресу http://<host-ip>:3000 і авторизуватись, використовуючи логін admin і пароль changeme (рис. 4.2).

Рисунок 4.2 - Зовнішній вигляд системи Grafana перед початком роботи

Також можливо змінити пароль за допомогою меню Grafana UI або змінивши файл user.config на потрібний зі зміненим паролем у будь-якому текстовому редакторі.

Для тестування системи необхідно з меню Grafana обрати пункт «Джерела даних» (Data Sources) і клікніть «Додати джерело даних» (Add Data Source).

Для того, щоб додати контейнери Prometheus як джерело даних, необхідно використовувати наступні значення:

Ім'я: Prometheus

Тип: Prometheus

Url: http://<host-ip>/prometheus:9090

Доступ: proxy

Після цього можливо виконувати імпорт шаблонів панелі керування з директорії Grafana. Для цього, з меню Grafana виберіть пункт «Панель управління» та натисніть «Імпорт».

Потім, під час тестування заданого вузла за допомогою системи Grafana будуть відображатись основні метрики роботи що були налаштовані за замовчуванням (рис. 4.3):

- час працездатності сервера, відсоток простою процесору, кількість ядер процесору, swap-пам'ять, дискові сховища;

- графік середнього навантаження на системи, графік виконаних і заблокованих процесів вводу-виведення, графік переривань;

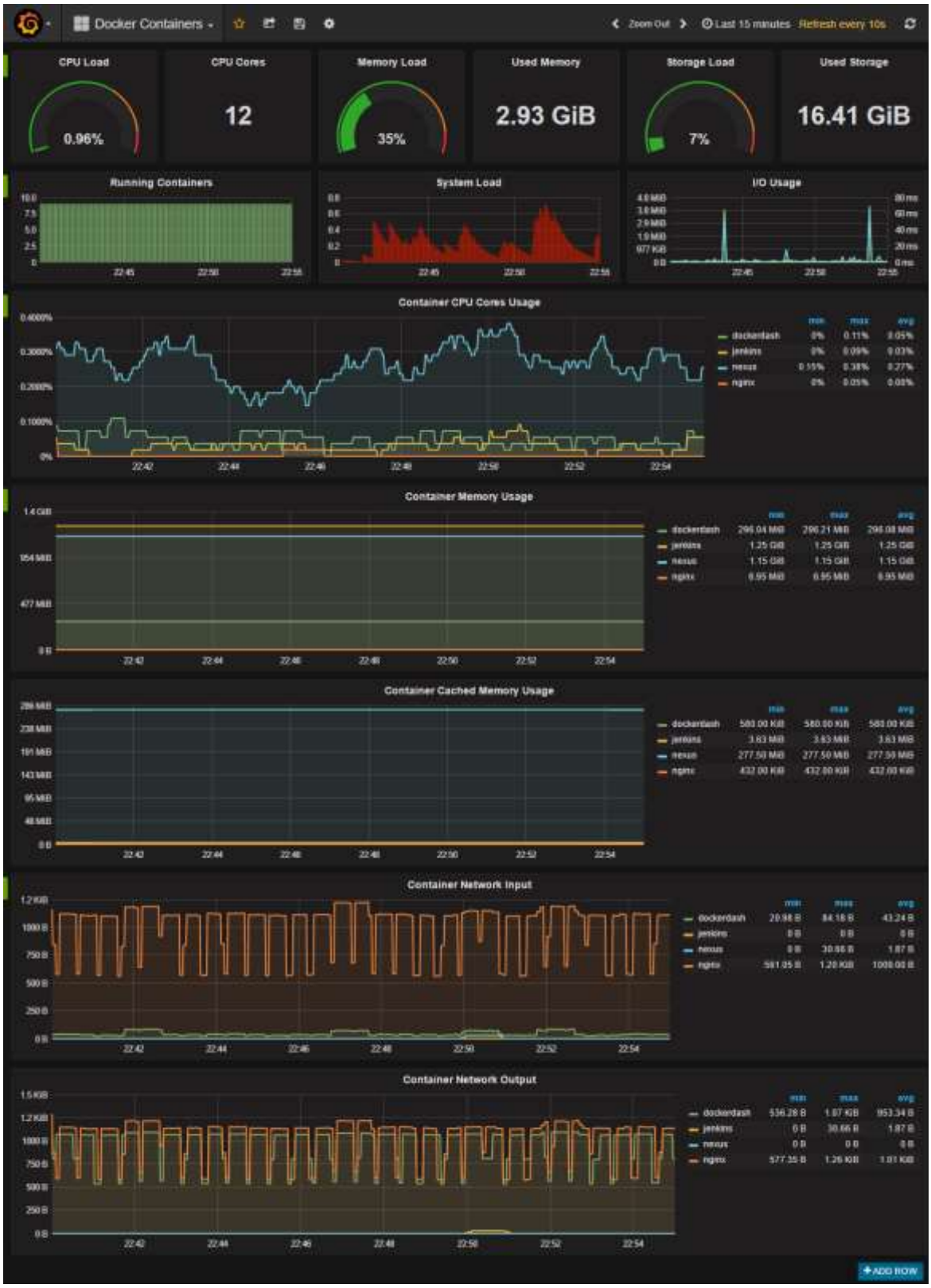

Рисунок 4.3 - Тестування заданого вузла корпоративної мережі за допомого метрик системи Grafana що налаштовані за замовчуванням

- графік використання процесору в режимах guest, idle, iowait, irq, nice, softirq, steal, system, user;

- графік використання пам'яті за розподілом (використано, вільно, буфери, кешуватися);

- графік використання системи вводу-виведення (read Bps, read Bps and IO time);

- графік використання мережевих пристроїв (входить Bps, що виходить Bps).

Панель управління контейнерів докера (рис. 4.3) відображає ключові метрики для моніторингу використовуваних контейнерів під час тестування системи. До таких основних метрик належать:

- загальне навантаження контейнерів процесору, використання пам'яті і сховища;

- графік навантаження системи, графік використання системи вводу-виведення;

- графік використання контейнера процесору;

- графік використання пам'яті контейнера;

- графік використання кешованої пам'яті;

- графік вхідного та вихідного трафіків використання мережі контейнерів.

Під час тестування панель управління моніторинговими сервісами відображає ключові метрики для моніторингу контейнерів, які є основними складовими моніторинговий стек.

Тестування роботи контейнеру Prometheus, показувало загальне використання пам'яті моніторингового стека, фрагменти і серії пам'яті локального сховища Prometheus які складались з наступних графіків:

- використання контейнера процесору;

- використання пам'яті контейнера;

- операції вводу-виведення Prometheus і тривалості встановлення контрольних точок;

- відсотка використаних шаблонів Prometheus, цільових зчитувань та операцій тривалості зчитування;

- запитів Prometheus до сервісу HTTP;

- різних повідомлень Prometheus.

Контролювати використання пам'яті Prometheus відбувалось приєднанням фрагментів пам'яті локального дискового сховища. При цьому, під час тестування системи моніторингу змінювалось максимальне значення фрагментів у файлі docker-compose.yml. Налаштовані значення були записані до файлу storage.local.memory-chunks та досягали значень до 100000.

В результаті тестування системи моніторингу за допомогою Prometheus виявлено, що якщо визанчати значення основних обє'ктів інформаційних ресурсів серверів (процесору, оперативної пам'яті, доскового простору та мережевої картки) у кількості 10 контейнерів, то дослідна система буде використовувати близько 2 Гб оперативної пам'яті.

4.3 Тестування файлів конфігурацій

Під час тестування важливо визначати повідомлення які встановлюються у трьох конфігураційних файлах:

- повідомлення сервісів моніторингу targets.rules;

- повідомлення хоста докера hosts.rules;

- повідомлення контейнерів докера containers.rules.

В цьому випадку можливо змінювати правила повідомлення та перезавантажувати їх за допомогою запиту HTTP POST:

curl -X POST http://<host-ip>:9090//reload

Повідомлення сервісів моніторингу відбуваються у випадку, якщо один з цільових об'єктів (node-exporter і cAdvisor) не відповідає більше 30 секунд. Тоді включається повідомлення за допомогою скрипта який знаходиться у конфігураційному файлі:

ALERT monitor_service_down

IF up == 0

FOR 30s

LABELS { severity = "critical" }

ANNOTATIONS {

summary = "Monitor service non-operational",

description = "{{ $labels.instance }} service is down.",

}

Повідомлення хоста докера відбувається у випадку, якщо процесор хоста докера знаходиться під високим навантаженням понад 30 секунд. Тоді включається повідомлення за допомогою такого скрипта який знаходиться у конфігураційному файлі:

ALERT high_cpu_load

IF node_load1 > 1.5

FOR 30s

LABELS { severity = "warning" }

ANNOTATIONS {

summary = "Server under high load",

description = "Docker host is under high load, the avg load 1m is at {{ $value}}. Reported by instance {{ $labels.instance }} of job {{ $labels.job }}.",

}

При цьому, знімимо граничне значення навантаження відповідно до кількості ядер процесору. Якщо пам'ять хоста докера заповнена, то включимо повідомлення за допомогою такого скрипта який знаходиться у конфігураційному файлі:

ALERT high_memory_load

*IF (sum(node_memory_MemTotal) - sum(node_memory_MemFree + node_memory_Buffers + node_memory_Cached)) / sum(node_memory_MemTotal) * 100 > 85*

FOR 30s

LABELS { severity = "warning" }

ANNOTATIONS {

summary = "Server memory is almost full",

description = "Docker host memory usage is {{ humanize $value}}%. Reported by instance {{ $labels.instance }} of job {{ $labels.job }}.",

}

Якщо дискове сховище вузла докера заповнене, то включимо повідомлення наступним чином:

ALERT hight_storage_load

*IF (node_filesystem_size{fstype="aufs"} - node_filesystem_free{fstype="aufs"}) / node_filesystem_size{fstype="aufs"} * 100 > 85*

FOR 30s

LABELS { severity = "warning" }

ANNOTATIONS {

summary = "Server storage is almost full",

description = "Docker host storage usage is {{ humanize $value}}%. Reported by instance {{ $labels.instance }} of job {{ $labels.job }}.",

}

Якщо контейнер не відповідає протягом 30 секунд, включимо наступне повідомлення за допомогою такого коду скрипта:

ALERT jenkins_down

IF absent(container_memory_usage_bytes{name="jenkins"})

FOR 30s

LABELS { severity = "critical" }

ANNOTATIONS {

summary= "Jenkins down",

description= "Jenkins container is down for more than 30 seconds."

}

Якщо контейнер використовує більше ніж 10% ядер процесору більше 30 секунд, то включимо повідомлення за допомогою такого коду скрипта:

ALERT jenkins_high_cpu

*IF sum(rate(container_cpu_usage_seconds_total{name="jenkins"}[1m])) / count(node_cpu{mode="system"}) * 100 > 10*

FOR 30s

LABELS { severity = "warning" }

ANNOTATIONS {

summary= "Jenkins high CPU usage",

description= "Jenkins CPU usage is {{ humanize $value}}%."

}

У випадку, якщо контейнер використовує понад 1,2 Гб RAM протягом 30 секунд, то включимо повідомлення таким чином:

ALERT jenkins_high_memory

IF sum(container_memory_usage_bytes{name="jenkins"}) > 1200000000

FOR 30s

LABELS { severity = "warning" }

ANNOTATIONS {

summary = "Jenkins high memory usage",

description = "Jenkins memory consumption is at {{ humanize $value}}.",

}

В системі моніторингу існує важливий сервис AlertManager який відповідає за передачу повідомлень сервера Prometheus. В ході тестування AlertManager може посилати повідомлення за допомогою електронної пошти, Pushover, Slack, HipChat та інших систем, що використовують інтерфейс webhook.

Підсистему повідомлень можливо переглянути або вимкнути за адресою:

http://<host-ip>:9093 .

Режими отримання повідомлень можна змінювати в конфігураційному файлі alertmanager/config.yml .

В ході тестування, щоб отримувати повідомлення через підсистему Slack, необхідно налаштувати інтеграцію, вибравши пункт «Вихідні мережеві прив'язки» на сторінці додатка. Після цього, скопіюйте значення Slack Webhook URL до поля api_url і визначте канал Slack наступним чином:

```
route:
  receiver: 'slack'
receivers:
  - name: 'slack'
    slack_configs:
      - send_resolved: true
        text: "{{ .CommonAnnotations.description }}"
        username: 'Prometheus'
        channel: '#<channel>'
api_url: 'https://hooks.slack.com/services/<webhook-id>'
```

Таким чином, тестування системи моніторингу за допомогою Prometheus ефективно виявляє та визначає значення основних обє'ктів інформаційних ресурсів серверів: процесору, оперативної пам'яті, доскового простору та мережевої картки у різній кількості контейнерів. Використання підсистеми повідомлень дозволяє оперативно відслітковувати різні типи критичних ситуацій.

4.4 Висновки

В цьому розділі створено програмне оточення з розробки основних метрик системи моніторингу яка передбачала якісну роботу та складалась з таких основних функцій: надання сховища метрикам, візуалізація та оповіщення різних станів роботи дослідних об'єктів для фізичних серверів,

віртуальних машин, контейнерів або сервісів та забезпечення роботи різних систем взаємодії з графічними об'єктами - Graphite, Prometheus, Elastic Beats, тощо.

Визначені можливості використання системи Prometheus, описані кроки встановлення цієї системи з метою проведення тестування.

Описано тестування системи моніторингу за допомогою Grafana. Виконано тестування файлів конфігурацій системи моніторингу, наведені основні кроки та протоколи роботи програмного засобу.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть _____ та _____.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.1.

Таблиця 5.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Експерт 1	2. Експерт 2
	Бали, виставлені експертами:	
1	4	4
2	3	3
3	3	4
4	4	4
5	3	4
6	3	4
7	3	3
8	3	4
9	4	4
10	4	3
11	3	4
12	3	4
Сума балів	$СБ_1 = 41$	$СБ_2 = 45$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 43$	

Отже, з отриманих даних таблиці 5.1 видно, що нова розробка має високий рівень комерційного потенціалу.

5.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (5.1):

$$З_о = \frac{М}{Т_р} \cdot t, \quad (5.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р$ = 21 день;

t - число днів роботи розробника, t = 40 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 5.2.

Таблиця 5.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	7000	333,33	5	1666,65
Інженер-програміст	4500	214,28	40	8571,2
Всього:				10237,85

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0,1 \cdot 10237,85 = 1023,78 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп} = (З_о + З_р) \cdot \frac{\beta}{100}, \quad (5.2)$$

$$Н_{зп}= (10237,85+1023,78)\cdot \frac{36,3}{100} = 4087,97 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц\cdot Н_а}{100}\cdot\frac{Т}{12}, \quad (5.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 5.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп’ютер	9000	25	3	562,5
Всього:				562,5

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (5.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 5.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	150	1	150
Пачка паперу	уп.	120	1	120
Ручка	шт.	10	1	10
Всього з урахуванням транспортних витрат				308

Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п \; ; \tag{5.5}$$

де В – вартість 1кВт-години електроенергії (В=1,7 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=180 год.);

$К_п$ – коефіцієнт використання потужності ($К_п < 1$, $К_п = 0{,}8$).

$$В_е = 1{,}7 \cdot 0{,}6\ 180 \cdot 0{,}8 = 146{,}88 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \tag{5.6}$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (10237{,}85 + 1023{,}78) = 11261{,}63 \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 10237,85+1023,78+4087,97+562,5+146,88+308+11261,63= 27682,61 (грн.)

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (5.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{27682,61}{1} = 27682,61$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \quad (5.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{27682,61}{0,9} = 30698,45 \text{ (грн.)}$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості

грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \quad (5.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 25 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 25 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 300 користувачів, протягом другого року – на 200 користувачів, протягом третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 1500 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 200 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 25 \cdot 1500 + (200 + 25) \cdot 300 = 105000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 25 \cdot 1500 + (200 + 25) \cdot (300 + 200) = 150000 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 25 \cdot 1500 + (200 + 25) \cdot (300 + 200 + 100) = 172500 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - PV), \qquad (5.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 5.1.

Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{30698,45}{(1+0,1)^0} + \frac{105000}{(1+0,1)^2} + \frac{150000}{(1+0,1)^3} + \frac{172500}{(1+0,1)^4} = 347992,34 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 347992,34 - 30698,45 = 317293,89 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[Т]{1 + \frac{Е_{абс}}{PV}} - 1 \qquad (5.12)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$Е_в = \sqrt[3]{1 + \frac{317293,89}{30698,45}} - 1 = 1,24 \text{ або } 124\ \%$$

Далі, розраховану величина $Е_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в = 124\% > \tau_{мін} = 0{,}3 = 30\%$, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}$$

$$T_{ок} = \frac{1}{1{,}24} = 0{,}80 \text{ року}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

5.5 Висновки

У п’ятому розділі було проведено оцінювання комерційного потенціалу розробки, виконано технологічний аудит для оцінювання комерційного потенціалу розробки. Залучено 2-х незалежних експертів. Спрогнозовано витрати на виконання науково-дослідної роботи та конструкторсько-технологічної роботи.

На вісі часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР показано життєвий цикл наукової розробки, початок отримання прибутків та його період.

Вартість чистого прибутку складає 347992,34 (грн.). Абсолютна ефективність вкладених інвестицій: 317293,89 грн. Відносна ефективність вкладених інвестицій: 1,24 або 124 %. Термін окупності: 0,80 року. Обрахувавши вартість чистого прибутку, абсолютну і відносну ефективність вкладених інвестицій та термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

В першому розділі виконано аналіз стану проблеми сучасних систем моніторингу серверів та комп'ютерних мереж. Розглянуто пасивний та активний моніторинг та призначення кожного з них. Наведені приклади роботи з SNMP протоколом який поширено використовується у системах моніторингу.

Розглянуті загальні характеристики поширених систем моніторингу: CACTI, MRTG, Nagios і Zabbix, HP OpenView і IBM Tivoli. Наголошено, на що слід звертати увагу під час вибору системи моніторингу.

Виконано порівняльний аналіз аналогів: Nagios, Zabbix, Hyperic, SolarWinds, ManageEngine OpManager, HP Operations Manager. Розглянуті основні вимоги щодо побудови системи: мультиплатформеність, швидкість обробки інформації, розподіленість в отриманні даних, безпека інформації, використання шаблонів та візуалізація даних.

Визначено, що рішення задач візуалізації полученених даних може бути забезпечено за рахунок: побудови звітів, створення графіків і діаграм, побудови мережевих карт.

Виконано постановка задачі до якої відносилось аналіз аналогів, виконання робіт що пов'язані з проектуванням, розробкою алгоритма, написання програмного коду та тестування програмного засобу.

В другому розділі визначені основні задачі проектування системи моніторингу. Визначено, що важливою частиною моніторингу серверів є спостерігання за їх працездатностю, тобто це робота системи, яка виконує постійний нагляд за обчислювальною мережею в пошуках повільних або пошкоджених систем. Під час виявлення будь-яких недоліків система повинна повідомляти про них адміністратору мережі за допомогою спеціальних засобів оповіщення.

Також визначені поширені класи програмних продуктів які використовуються у системах моніторингу серверів та є такі: засоби

управління системою, системи управління мережею, вбудовані системи діагностики і управління, аналізатори протоколів, експертні системи та багатофункціональні пристрої аналізу та діагностики.

Розроблені основні кроки проектування системи моніторингу до яких відносилось визначення предмету дослідження, формування мети, визначення категорій інформації та вимірювання, визначення відповідального за збір даних, виконання аналізу та прийняття рішення, формування.

Під час проектування інструментів моніторингу систем визначено, що сучасні вимоги щодо проектування серверів та комп'ютерних мереж вимагають більш точного і гнучкого підходу до систем моніторингу. Розглянуті робота веб та поштових серверів, розглянуті важливі етапи проектування систем моніторингу до яких відносились: особливості розгортання та супроводу систем моніторингу, щоб підібрати інструмент, проведення аналізу інформаційних та технологічних ресурсів, визначення рівня компетенції команди IT-спеціалістів, що будуть обслуговувати систему моніторингу, обчислення загальної вартості системи моніторингу що буде впроваджуватись.

Проектування важливих інструментів систем моніторингу передбачала дослідження таких категорій дослідження як: функціональність системи, інтерфейсу користувача, системи повідомлень, інформаційної структури, інсталяції та супроводу системи.

В третьому розділі описані організація і основні функції систем моніторингу та визначено, що однією з актуальних проблем для підприємств з розвиненими автоматизованими системами є формування системи відстеження (моніторингу) в режимі реального часу.

Визначені основні функції систем моніторингу до яких належали: контроль в режимі реального часу параметрів, що надходять від об'єктів управління, контроль параметрів в режимі реального часу локальні баз даних систем управління організації, відображення значень, формування звітів для

аналізу показників, формування звітів для аналізу стану обладнання засобів вимірювання і автоматизації. Розроблено інтерфейс основної частини системи моніторингу та додаткових модулів: веб-сервісу зв'язку, клієнтів, локальних серверів, та баз даних. Розроблено алгоритм, діаграми класів та програмного коду.

В четвертому розділі створено програмне оточення з розробки основних метрик системи моніторингу яка передбачала якісну роботу та складалась з таких основних функцій: надання сховища метрикам, візуалізація та оповіщення різних станів роботи дослідних об'єктів для фізичних серверів, віртуальних машин, контейнерів або сервісів та забезпечення роботи різних систем взаємодії з графічними об'єктами - Graphite, Prometheus, Elastic Beats, тощо.

Визначені можливості використання системи Prometheus, описані кроки встановлення цієї системи з метою проведення тестування.

Описано тестування системи моніторингу за допомогою Grafana. Виконано тестування файлів конфігурацій системи моніторингу, наведені основні кроки та протоколи роботи програмного засобу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. J. Kowall, W. Cappelli. Magic Quadrant for Application Performance Monitoring. Gartner, Oct. 2014. / [Електронний ресурс] – Режим доступу: URL http://www.gartner.com/doc/2889421/magic-quadrant-application- performance-monitoring - Назва з екрану.

2. Vendor Landscape: Systems Management. Info-Tech Research Group, 2011. / [Електронний ресурс] – Режим доступу: URL http://www.infotech.com/ research/ss/it-vendor-landscape-systems-management - Назва з екрану.

3. K. Fatema et al. A Survey of Cloud Monitoring Tools: Taxonomy, Capabilities and Objectives // J. Parallel and Distributed Computing. — 2014. Vol. 74, N. 10. — P. 2918–2933.

4. A Simple Network Management Protocol (SNMP). / [Електронний ресурс] – Режим доступу: URL http://www.ieft.org/rfc1157.txt - Назва з екрану.

5. IBM Tivoli — Integrated Management software [электронный ресурс] / [Електронний ресурс] – Режим доступу: URL http://www.ibn.com/ software/tivoly - Назва з екрану.

6. IT Performance Suite | HP Enterprise Software [Электронный ресурс] / [Електронний ресурс] – Режим доступу: URL http://www.managementsoftware.hp.com - Назва з екрану.

7. Stallings William. SNMP, SNMPv2, SNMPv3, and RMON 1 and 2 (3rd Edition). Addisison-Wesley Professional, 2009. P. 42.

8. A Simple Network Management Protocol (SNMP). / [Електронний ресурс] – Режим доступу: URL http://www.ieft.org/rfc1157.txt - Назва з екрану.

9. IT Performance Suite | HP Enterprise Software / [Електронний ресурс] – Режим доступу: URL http://www.managementsoftware.hp.com - Назва з екрану.

10. Josephsen David. Building a Monitoring Infrastructure with Nagios. Prentice Hall, 2007. P. 92.

11. Lavlu Ibrahim, Kundu Dinangkur. Cacti 0.8 Network Monitoring. Pact Publishing, 2009. P. 87.

12. Stallings William. SNMP, SNMPv2, SNMPv3, and RMON 1 and 2 (3rd Edition). Addisison-Wesley Professional, 2012. P. 38.

13. Shipway Steve. Using MRTG with RRDtool and Routers2. Cheshire Cat Computing, 2016. P. 52.

14. Josephsen David. Building a Monitoring Infrastructure with Nagios. Prentice Hall, 2016. P. 63.

15. Lavlu Ibrahim, Kundu Dinangkur. Cacti 0.8 Network Monitoring. Pact Publishing, 2014. P. 38.

16. Olups Rihards. Zabbix 1.8 Network Monitoring. Packt Publishing, 2015. P. 32.

17. Shipway Steve. Using MRTG with RRDtool and Routers2. Cheshire Cat Computing, 2013. P. 38.

18. IBM Tivoli — Integrated Management software / [Електронний ресурс] – Режим доступу: URL http://www.ibn.com /software/tivoly - Назва з екрану.

19. Сафонов А.Ю., Костенецкий П.С. Система сбора и отображения статистики о загрузке суперкомпьютеров ЛСМ ЮУрГУ / Параллельные вычислительные технологии (ПаВТ'2015): труды международной научной конференции (30 марта – 3 апреля 2015 г., г. Екатеринбург). Челябинск: Издательский центр ЮУрГУ, 2015. С. 516.

20. Воеводин В. В. Ситуационный экран суперкомпьютера // Открытые системы. 2014. № 3. С.37-41.

21. Козырев В.И., Костенецкий П.С. Опыт использования VDI-системы «Персональный виртуальный компьютер» в ЮУрГУ // Научный сервис в сети Интернет: поиск новых решений: Труды Международной суперкомпьютерной конференции (17-22 сентября 2012 г., г. Новороссийск). М.: Изд-во МГУ, 2012. С. 285-286.

22. Костенецкий П.С., Семенов А.И., Соколинский Л.Б. Создание образовательной платформы ”Персональный виртуальный компьютер” на базе облачных вычислений // Научный сервис в сети Интернет: экзафлопсное будущее: Труды Международной суперкомпьютерной конференции. М.: Издательство МГУ, 2011. С. 374–377.

23. Костенецкий П.С., Семенов А.И. Организация виртуальных персональных компьютеров студентов на базе суперкомпьютера // Параллельные вычислительные технологии (ПаВТ’2011): Труды международной научной конференции (28 марта – 1 апреля 2013 г., г. Москва). Челябинск: Издательский центр ЮУрГУ, 2013. С. 699.

24. Кизина И.Д. Математическое моделирование и прикладные информационные технологии для MES-уровня управления //Автоматизация, телемеханизация и связь в нефтяной промышленности. – М.: ОАО "ВНИИОЭНГ", 2012. – № 4. – С. 37–45.

ДОДАТКИ

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
"____ "_____2019 р.

Технічне завдання
на магістерську кваліфікаційну роботу
«Розробка системи моніторингу корпоративних серверів»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
к.т.н., доцент кафедри ПЗ, О.О. Коваленко
р.
Виконав:
_____
студент гр. 1ПІ-18м, Д.В. Колос

Вінниця ВНТУ 2019

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка системи моніторингу корпоративних серверів».

Галузь застосування – моніторинг серверів.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Мета дослідження – розширення функціоналу системи моніторингу об’єктів ІТ-інфраструктури..

Призначення роботи – розробка методу та системи моніторингу корпоративних серверів.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Lavlu Ibrahim, Kundu Dinangkur. Cacti 0.8 Network Monitoring. Pact Publishing, 2009. P. 87.
2. Сафонов А.Ю., Костенецкий П.С. Система сбора и отображения статистики о загрузке суперкомпьютеров ЛСМ ЮУрГУ / Параллельные вычислительные технологии (ПаВТ'2015): труды международной научной конференции (30 марта – 3 апреля 2015 г., г. Екатеринбург). Челябинск: Издательский центр ЮУрГУ, 2015. С. 516.
3. Блинов, Романчик. Java. Методы программирования.– СПб. : Минск, 2013. – 897 с.

5. Технічні вимоги

Мова програмування: Java

Технологія розробки: IntelliJ IDEA

Браузери (або ОС): Windows

6. Конструктивні вимоги.

Програмний продукт повинен відповідати всім вимогам, повинен бути зручним та зрозумілим у використанні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Техніко-економічне обґрунтування доцільності розробки системи моніторингу корпоративних серверів	07.10.2019 – 27.10-2019
2	Варіативне моделювання системи моніторингу об'єктів іт-інфраструктури	28.10.2019 – 8.11.2019
3	Розробка програмного засобу для моніторингу корпоративних серверів	9.11.2019 – 20.11.2019
4	Тестування роботи системи	21.11.2019 – 3.12.2019
5	Економічне обґрунтування розробки програмного продукту	4.12.2019 – 7.12.2019

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

Додаток Б. Програмний код додатку

```
package monitoringSystem;
public class DataBaseService{
  //------------------------
  // Визначення змінних основного класу системи моніторингу
  //------------------------
  //DataBaseService Attributes
  private String code;
  private String limit;
  private String description;
  //Helper Variables
  private int cachedHashCode;
  private boolean canSetCode;
  //------------------------
  // Визначення конструктора основного класу системи моніторингу
  //------------------------
  public DataBaseService(String aCode, String aLimit, String aDescription)  {
    cachedHashCode = -1;
    canSetCode = true;
    code = aCode;
    limit = aLimit;
    description = aDescription;
  }
  //------------------------
  // Визначення основних методів системи моніторингу
  //------------------------
  public boolean setCode(String aCode)  {
    boolean wasSet = false;
    if (!canSetCode) { return false; }
    code = aCode;
    wasSet = true;
    return wasSet;
  }
  public boolean setLimit(String aLimit)  {
    boolean wasSet = false;
    limit = aLimit;
    wasSet = true;
    return wasSet;
  }
  public boolean setDescription(String aDescription)  {
    boolean wasSet = false;
    description = aDescription;
    wasSet = true;
    return wasSet;
  }
  public String getCode()  {
    return code;
  }
  public String getLimit()  {
    return limit;
  }
  public String getDescription()  {
```

```
    return description;
  }
  public boolean equals(Object obj)  {
    if (obj == null) { return false; }
    if (!getClass().equals(obj.getClass())) { return false; }
    DataBaseService compareTo = (DataBaseService)obj;
    if (getCode() == null && compareTo.getCode() != null)    {
      return false;
    }
    else if (getCode() != null && !getCode().equals(compareTo.getCode()))    {
      return false;
    }
    return true;
  }
  public int hashCode()  {
    if (cachedHashCode != -1)    {
      return cachedHashCode;
    }
    cachedHashCode = 17;
    if (getCode() != null)    {
      cachedHashCode = cachedHashCode * 23 + getCode().hashCode();
    }
    else    {
      cachedHashCode = cachedHashCode * 23;
    }
    canSetCode = false;
    return cachedHashCode;  }
  public void delete()  {}

//  Метод для виведення отриманих та визначених даних до системи
  public String toString()  {
    return super.toString() + "["+
          "code" + ":" + getCode()+ "," +
          "limit" + ":" + getLimit()+ "," +
          "description" + ":" + getDescription()+ "]";
  }
}
public void init() {
        // online
        MetricClient.current().register(() -> {
            Builder<Metric> builder = ImmutableList.builder();

            final int playerCount = GameObjectsStorage.getAllRealPlayersCount();
            final int offtradeCount = GameObjectsStorage.getAllOfflineCount();
            builder.add(metric("online")
                    .field("real", playerCount)
                    .field("offtrade", offtradeCount)
                    .build());

            return builder.build();
        });

        // objects storage
```

```
        MetricClient.current().register(() -> {
            Builder<Metric> builder = ImmutableList.builder();

            List<GameObjectsStorage.Stats> stats = GameObjectsStorage.getInnerStats();
            stats.forEach(s -> {
                builder.add(metric("object_storage." + s.name.toLowerCase())
                        .field("size", s.size)
                        .field("real_size", s.realSize)
                        .field("capacity", s.capacity)
                        .field("init_capacity", s.initCapacity)
                        .build());
            });

            return builder.build();
        });

        // object stats
        MetricClient.current().register(() -> {
            Builder<Metric> builder = ImmutableList.builder();
            World.Stats stats = World.getInnerStats();
            builder.add(metric("world")
                    .field("reflections", stats.reflections)
                    .field("objects", stats.objectCount)
                    .field("nullObjects", stats.nullObjectCount)
                    .field("monsters", stats.monsterCount)
                    .field("minions", stats.minionCount)
                    .field("npcs", stats.npcCount)
                    .field("chars", stats.charCount)
                    .field("doors", stats.doorCount)
                    .field("summons", stats.summonCount)
                    .field("ai", stats.AICount)
                    .field("extended_ai", stats.extendedAICount)
                    .field("summon_ai", stats.summonAICount)
                    .field("active_ai", stats.activeAICount)
                    .field("global_ai", stats.globalAICount)
                    .field("items_char", stats.itemsChar)
                    .field("items_charwh", stats.itemsCharWh)
                    .field("items_clan", stats.itemsClan)
                    .field("items_freight", stats.itemsFreight)
                    .field("items_mail", stats.itemsMail)
                    .field("items_pet", stats.itemsPet)
                    .field("items_void", stats.itemsVoid)
                    .build());
            return builder.build();
        });
    public List<Metric> get(ThreadPoolExecutor executor) {
        ImmutableList.Builder<Metric> builder = builder();
        //@formatter:off
        builder.add(from(submitted,     "executor.submitted",   b -> b.tag("name", name)));
        builder.add(from(running,       "executor.running",     b -> b.tag("name", name)));
        builder.add(from(completed,     "executor.completed",   b -> b.tag("name", name)));
        builder.add(from(duration,      "executor.duration",    b -> b.tag("name", name)));
        builder.add(from(wait,          "executor.wait",        b -> b.tag("name", name)));
```

```
    //@formatter:on

    builder.add(
            Metric.metric("executor.queue")
                    .tag("name", name)
                    .field("value", executor.getQueue().size())
                    .build()
    );

    builder.add(
            Metric.metric("executor._completed")
                    .tag("name", name)
                    .field("value", executor.getCompletedTaskCount())
                    .build()
    );

    RunnableStats stats = this.stats;
    if (stats != null) {
        builder.addAll(stats.get());
    }

    return builder.build();
}
```

Додаток В. Ілюстративний матеріал

ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДО ЗАХИСТУ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Завідувач кафедри ПЗ, д. т. н., професор _____ О. Н. Романюк

Науковий керівник, к. т. н., доцент кафедри ПЗ_____ О. О. Коваленко

Рецензент, к.т.н, доцент кафедри КН_____ І.В. Богач

Нормоконтроль, д. т. н., проф. кафедри ПЗ _____ О. О. Коваленко

Виконавець, студент групи 1ПІ-18м _____ Д. В. Колос