Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: «Розробка обчислювальної платформи для опрацювання IoT- даних у туманному обчислювальному середовищі »

Виконав: студент IV курсу, групи СПс-41
спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Старосельський М.В. (прізвище та ініціали)

Керівник (підпис) Бачинський М.В. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю.М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М.Р. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Петрик М.Р.
(підпис) (прізвище та ініціали)
« 06 » квітня 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Старосельському Маркіяну Вікторовичу

1. Тема роботи Розробка обчислювальної платформи для опрацювання IoT- даних у туманному обчислювальному середовищі

Керівник роботи Бачинський Михайло Володимирович, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом по університету від « 06 » квітня 2026 року № 4/9-172

2. Термін подання студентом роботи 22.06.2026

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1 Аналіз предметної області. Визначення вимог до розробки .
2. Проектування та розробка платформи. 3. Тестування.
4. Безпека життєдіяльності, основи охорони праці

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема роботи. 2. Актуальність, мета, задачі дослідження
3. Взаємодія IoT пристроїв та хмарних обчислень. 4. Функціональні вимоги. .
5. Програмні засоби та технології..
6. Діаграма розгортання туманного обчислювального кластера. 7. Діаграма розгортання обчислювальної платформи. 8. Діаграма компонентів обчислювальної платформи. .
9. Діаграма прецедентів обчислювальної платформи. 10. Діаграма пакетів опису складових контейнерів мікросервісів. Схема бази даних. 11. Алгоритм розгортання мікросервісу. 12. Тестування. 13. Частина файлу налаштувань Docker.
14. Висновки по роботі.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання *06.04.2026 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Аналіз предметної галузі дослідження	06.04–17.04.26	Виконано
2.	Обґрунтування актуальності дослідження	18.04–26.04.26	Виконано
3.	Огляд існуючих механізмів та засобів	27.04–04.05.26	Виконано
4.	Проведення дослідження механізмів та засобів		
	опрацювання даних	05.05–10.05.26	Виконано
5.	Оформлення розділу «Аналіз предметної області.	11.05–14.05.26	
	Визначення вимог до розробки »		Виконано
6.	Оформлення розділу «Проектування та розробка	15.05–18.05.26	
	платформи»		Виконано
7.	Оформлення розділу «Тестування»	19.06–23.05.26	Виконано
8.	Оформлення розділу «Безпека життєдіяльності,		
	основи охорони праці»	20.05–25.05.26	Виконано
9.	Нормоконтроль	06.06–15.06.26	Виконано
10.	Перевірка на плагіат	12.06–16.06.26	Виконано
11.	Попередній захист роботи	15.06–21.06.26	Виконано
12.	Захист кваліфікаційної роботи	28.06.26	

Студент _____ Старосельський М.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Бачинський М.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка обчислювальної платформи для опрацювання IoT- даних у туманному обчислювальному середовищі // Кваліфікаційна робота освітнього рівня «бакалавр» // Старосельський Маркіян Вікторович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра програмної інженерії, група СПс-41 // Тернопіль, 2026 // С. - 51, табл. - 4, рис. - 26, слайд. – 14, додат. - 2, бібліогр. – 23.

Ключові слова: віртуалізація, мікросервіс, обчислювальна платформа, розгортання системи, IoT-пристрій

У першому розділі виконано огляд предметної область, досліджено поняття інтернету речей. Оглянуто концепції хмарних і туманних обчислень. Відображено особливості контейнерної віртуалізації і оркестрації. Визначені вимоги до платформи, як функціональні такі нефункціональні.

У другому розділі спроектовано обчислювальну платформу, описано розгортання кластера та всієї платформи, наведені компоненти та варіанти використання платформи, контейнери та їх прецеденти. Також виконано проектування бази даних. Розроблено ряд мікросервісів для надавання розподіленого програмного забезпечення для збору, зберігання і інтелектуального аналізу даних. Втілено ядро платформи, котре виконує роль публічного API-сервера взаємин юзерів та платформи у туманному обчислювальному середовищі.

Третій розділ присвячено тестування розробки. із застосуванням сервісу Postman. Було здійснено запити до ядра розробленої платформи. Усі запити успішно повернули очікуваний результат і статус. Таким чином можна стверджувати, що тестування пройшло успішно.

Четвертий розділ висвітлює важливі питання безпеки життєдіяльності та основ охорони праці.

ABSTRACT

Development of a computing platform for processing IoT data in a fog computing environment // Bachelor Thesis // Staroselskyi Markiian // Ternopil Ivan Puluj National Technical University, Computer and Information Systems and Software Engineering Faculty, Department of Software Engineering, group SPs-41 // Ternopil, 2026 // P. - 51, tab. - 4, fig. - 26, slid. -14, annexes - 2, references - 23.

Keywords: virtualization, microservice, computing platform, system deployment, IoT device

The first chapter provides an overview of the subject area, explores the concept of the Internet of Things. The concepts of cloud and fog computing are reviewed. Features of container virtualization and orchestration are reflected. Requirements for the platform, both functional and non-functional, are determined.

The second chapter designs a computing platform, describes the deployment of the cluster and the entire platform, presents components and options for using the platform, containers and their use cases. Database design is also performed. A number of microservices are developed to provide distributed software for collecting, storing and intelligently analyzing data. The platform core is implemented, which plays the role of a public API server for user and platform interactions in a fog computing environment.

The third chapter deals with the development. using the Postman service. Requests were made to the core of the developed platform. All requests successfully returned the expected result and status. Thus, it can be stated that the testing was successful.

The fourth chapter highlights important issues of life safety and the basics of occupational health and safety.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОС – операційна система.

ПЗ – програмне забезпечення.

Тестування ПЗ – це процес перевірки відповідності реальної поведінки програми очікуваній, спрямований на виявлення дефектів, помилок та недоліків перед випуском продукту.

API (англ. Application Programming Interface) – опис способів, якими одна комп'ютерна програма може взаємодіяти з іншою програмою.

IoT (агл. Internet of Things) – інтернет речей.

UML (англ. Unified Modeling Language) – універсальна графічна мова моделювання, що використовується для візуалізації, проектування та документування програмних систем, бізнес-процесів та структур даних.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Сьогодні неможливо уявити життя без повсюдної інформатизації. Інформаційні технології настільки впроваджені у життєдіяльність сучасної людини, що, здається, у світі практично не залишається осіб, які хоч раз не скористалися ними.

Однозначно, з розвитком інформаційних технологій життя стає дедалі зручнішим і безпечнішим.

Поява електронної обчислювальної техніки дозволила людині виконувати мільярди логічних та математичних операцій за короткий час, що стало еволюційним проривом для людства. Впровадження сучасних комп'ютерів створило таку еволюцію у кожному будинку. Організація єдиної мережі передачі даних між комп'ютерами дозволила людям ділитися інформацією на відстані в сотні тисяч кілометрів без тривалого очікування відправлення та доставки повідомлень, що забезпечило миттєве здобуття знань про все, що тільки існує в нашому житті. Безперервне підвищення обчислювальної ємності комп'ютерної техніки досі створює нові можливості розвитку як інформаційної сфери, так всього людства.

Актуальність розробки обчислювальної платформи з організацією збору, зберігання та аналізу IoT даних у туманному обчислювальному середовищі полягає у застосуванні новітньої обчислювальної концепції туманних обчислень, яка дозволить як скоротити час передачі даних з IoT- пристроїв до обчислювальної інфраструктури, так і розгортати власні інфраструктури. провайдерів та реалізації цифрових рішень на основі платформи з організацією взаємодії з IoT- пристроями.

Метою роботи є проєктування та програмна реалізація обчислювальної платформи з втіленням збирання, зберігання та аналізу даних IoT у туманному обчислювальному середовищі.

Для досягнення мети потрібно вирішити наступні завдання:

– провести детальний огляд предметної галузі;

- визначити функціональні та нефункціональні вимоги;
- спроектувати обчислювальну платформу;
- розробити обчислювальну платформу;
- провести тестування програмних інтерфейсів;
- виконати наліз одержаних результатів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ВИЗНАЧЕННЯ ВИМОГ ДО РОЗРОБКИ

Зараз, саме коли технології IoT і хмарних обчислень твердо знаходять своє використання, удосконалюються нові обчислювальні концепції, як-от технології туманних обчислень. Цей вид обчислень є достатньо новітньою концепцією надання обчислювальних ресурсів, котра лише починає набирати своєї популярності [1].

1.1 Поняття IoT

IoT – це концепція взаємодії сенсорів та виконавчих пристроїв як у мережі інтернет, так і за допомогою спеціальних видів мереж.

До спеціальних видів мереж IoT відносяться:

– мережі з низьким енергоспоживанням та малим діапазоном (Bluetooth, Z-Wave, NFC, ZigBee, WiFi/802.11) [2];

– мережі з низьким енергоспоживанням та широкою зоною охоплення (4G LTE, 5G, Cat-0, Cat-1, LoRaWan, LTE, CAT-M1, Sagfox).

Розвиток технологій IoT багато в чому вплинув життя сучасної людини. За допомогою «розумної» взаємодії давачів і актуаторів у повсякденній життєдіяльності виникла, раніше звична лише для промислових підприємств, автоматизація, яка дозволила звести до мінімуму звичайні та рутинні події. Варто додати, що автоматизація вплинула не тиільки на буденність і рутину, а й значно підвищила рівень життя сучасної людини з допомогою використання збору та аналізу оброблюваної інформації.

На рисунку 1.1 представлено процес взаємодії IoT- пристроїв із хмарними обчисленнями.

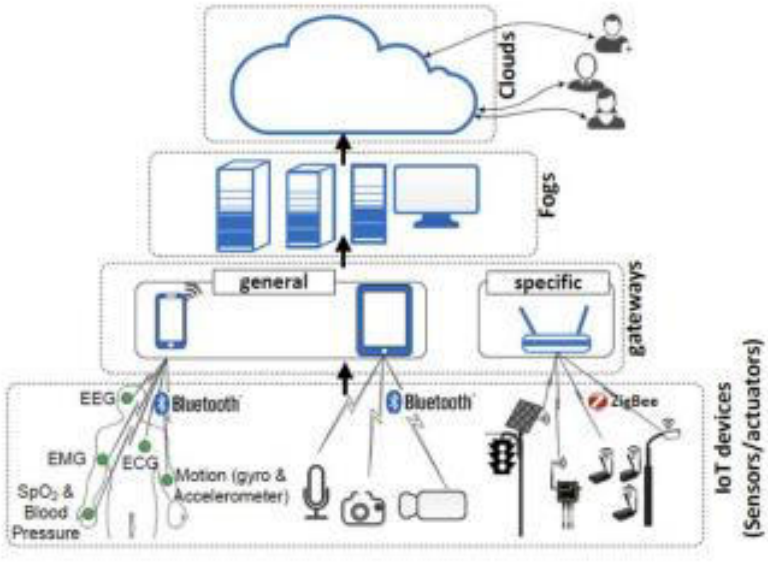

Рисунок 1.1 – Взаємодія IoT пристроїв та хмарних обчислень

1.2 Промисловий IoT

Технології IoT знайшли своє застосування не лише у підвищенні якості та безпеки життєдіяльності сучасної людини, але також дозволили внести нововведення в бізнес-процеси промислового виробництва. Технології та пристрої традиційного інтернету речей, адаптовані під потреби промисловості, було названо промисловим інтернетом речей (Industrial internet of things, IIoT).

Використання технологій IIoT багато в чому дозволило підприємствам:

- збільшити капіталізацію;
- збільшити маржинальність товарів та послуг;
- скоротити витрати;
- створити та протестувати нові бізнес-моделі;
- підвищити швидкість адаптації до зовнішніх змін;
- підвищити ефективність бізнес-процесів;
- скоротити трудовитрати виробництва.

До основних функціональних можливостей концепції IIoT відноситься прогнозування результату продукції, а також моніторинг та диспетчеризація стану обладнання та інфраструктури підприємства.

1.3 Хмарні обчислення

Це модель надання послуг обчислення та зберігання інформації у віддалених центрах обробки даних за допомогою використання Інтернету [3].

Модель надання обчислювальних послуг поділяється на 3 основні види, як наведено на рисунку 1.2:

- інфраструктура як сервіс (infrastructure as a service, IaaS);
- платформа як сервіс (platform as a service, PaaS) ;
- ПЗ як сервіс (software as a service, SaaS).

IaaS передбачає надання таких обчислювальних ресурсів, як віртуальні машини, мережі, кластери та реєстри контейнерів. Даний вид послуги спрямований на взаємодію з розробниками та системними адміністраторами.

PaaS передбачає надання зручних інструментів розробки ПЗ, які нерідко мають тісну інтеграцію з інфраструктурними хмарними компонентами. Модель надання PaaS спрямована на взаємодію з розробниками, адміністраторами, архітекторами програмних рішень та іншими, причетними до розробки ПЗ, особами.

SaaS надає функціональні можливості для прямої взаємодії з користувачем, який може не знати про розробку та адміністрування ПЗ, але бажає придбати необхідні йому функції за певну вартість.

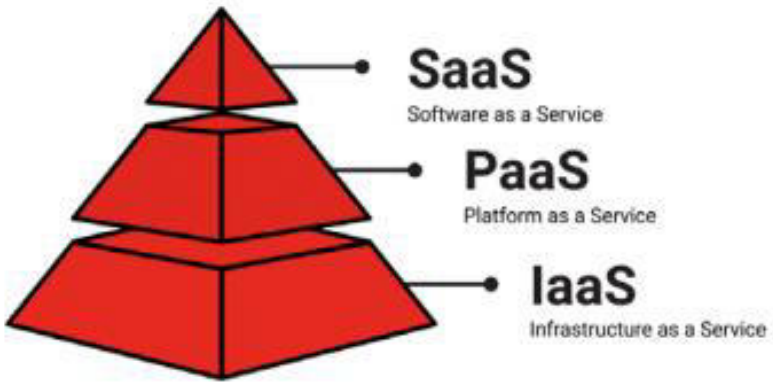

Рисунок 1.2 – Моделі надання хмарних послуг

Варто зазначити, що кожна з розглянутих раніше моделей надання хмарних обчислювальних послуг може бути представлена різною мірою доступності. Таким чином, виділяють такі види доступності хмар:

- відкриті;
- приватні;
- гібридні.

Відкриті хмарні платформи орієнтуються на публічне надання послуг.

Приватні (приватні) хмари організуються для надання обчислювальних ресурсів конкретному та обмеженому числу споживачів.

Гібридне надання хмарних сервісів передбачає часткове розгортання обчислювальної інфраструктури в громадській області користування та часткове в приватній.

1.4 Туманні обчислення

Це концепція надання обчислювальних ресурсів у безпосередній близькості до кінцевих користувачів [4]. Слід зазначити, що кінцевим користувачем може бути IoT пристрій.

Туманна обчислювальна концепція призначена для мінімізації часу передачі від користувача до обчислювального вузла.

Варто розуміти, що туманні обчислення доповнюють хмарні, виконуючи передиктивний аналіз та агрегацію даних перед відправкою для обчислень у хмарний центр обробки даних [5].

На рисунку 1.3 наведені обчислювальні шари хмарних та туманних обчислень.

Рисунок 1.3 – Обчислювальні шари хмарних та туманних обчислень

1.5 Контейнерна віртуалізація

На відміну від апаратної віртуалізації, коли емулюється апаратне оточення і може бути запущено безліч гостьових ОС, в контейнері запускається екземпляр ОС тільки з тим же ядром, що й у вихідної (хостової) ОС. Таким чином, усі контейнери вузла використовують загальне ядро. При цьому відсутні додаткові витрати ресурсів на емуляцію віртуального обладнання і старт власне повноцінного екземпляра ОС, що якраз і є характерним для апаратної віртуалізації [6].

Таким чином, контейнери дозволяють вмістити набагато більше програм на одному фізичному сервері, ніж будь-яка віртуальна машина, яка займає набагато більше системних ресурсів. На відміну від віртуальної машини, де емулюється ОС та необхідне віртуальне обладнання, у контейнері розміщується лише застосунок та необхідний мінімум системних бібліотек. Завдяки цьому можна запустити у 2-3 рази більше програм на одному сервері. Також контейнеризація дозволяє створювати портативне та цілісне оточення для розробки, тестування та подальшого розгортання [7].

На рисунку 1.4 відображено зіставлення архітектур віртуалізації на основі гіпервізора та на основі контейнерної віртуалізації.

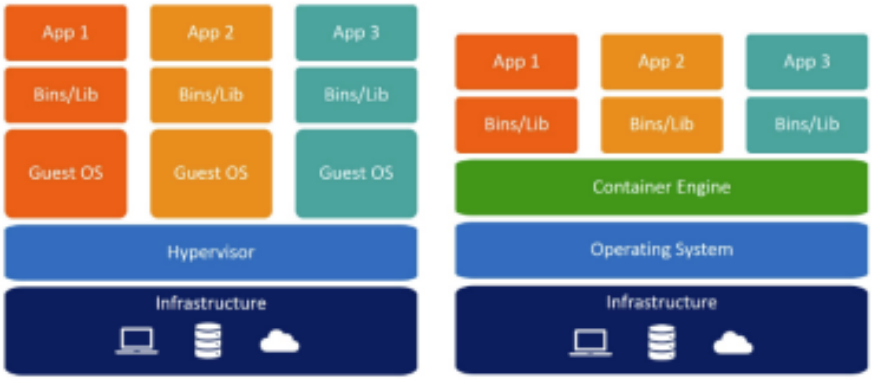

Рисунок 1.4 – Порівняння архітектури віртуалізації на базі гіпервізора та архітектури на базі контейнерної віртуалізації

1.6 Контейнерна оркестрація

Є видом організації спільної роботи множини сервісів у межах однієї програми [9]. На рисунку 1.5 наведено схему контейнерів, об'єднаних у кластер.

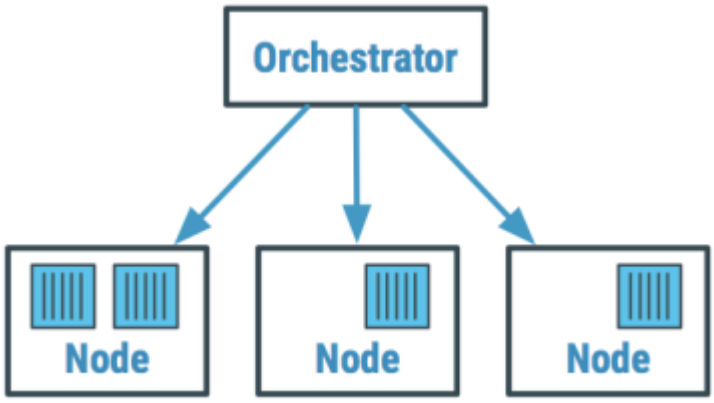

Рисунок 1.5 – Схема розгортання контейнерів у кластері

На даний момент, є безліч рішень для оркестрації контейнерів, які можуть керувати повним життєвим циклом контейнерів або групою контейнерів, масштабуючи їх вертикально або горизонтально, переміщуючи, самостійно відновлюючи, виконуючи оркестрацію зберігання та інші функції [9. 10].

Лідерами рішень контейнерної оркестрації є Kubemetes [11] та Docker Swarm [12].

1.6.1 Kubernetes

Це open-source інструмент оркестрації для забезпечення автоматизації розгортання, а також для масштабування і керування програмами на основі контейнерів. Можна запускати Kubemetes у локальній або публічній хмарній інфраструктурі. Kubemetes встановлює кластер, що складається з Kubemetes-master, Kubemetes-minions (робочі вузли) та Pod (один або група контейнерів для запуску однієї програми). Kubemetes-master спостерігає за одним або декількома міньйонами та керує застосунками [11].

Kubemetes керує та втілює контейнери на значному числі хостів, на додачу надає спільне розміщення і копіювання великого числа контейнеризованих контейнерів. Kubemetes забезпечує високу надійність контейнерів, ефективно використовує доступні ресурси, координує розгортання та реалізує взаємодію контейнерів між собою.

На рисунку 1.6 показано архітектуру платформи оркестрації Kubernetes

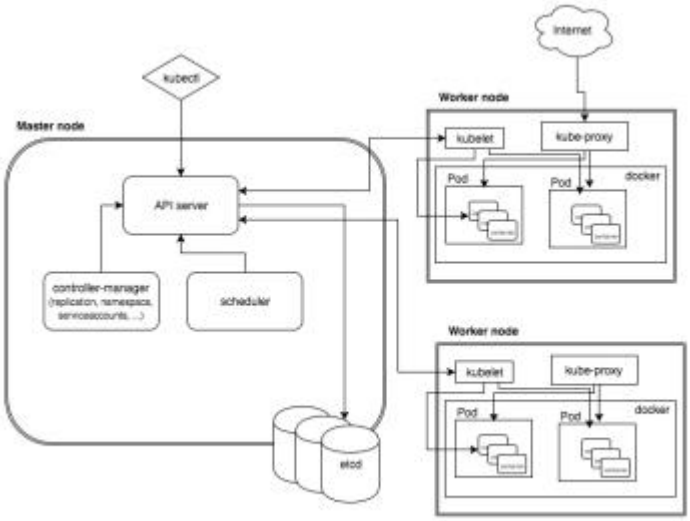

Рисунок 1.6 – Архітектура Kubernetes

1.6.2 Docker Swarm

Це кластерний режим платформи контейнерної віртуалізації Docker, що дозволяє без перешкод організовувати кластер обчислювальних вузлів для управління контейнеризованими застосунками. Є одним із найпростіших за

взаємодією оркестраторів [13].

Архітектурно складається з керуючих та робочих вузлів, кожен із виконує свою функцію [14] (рисунок 1.7).

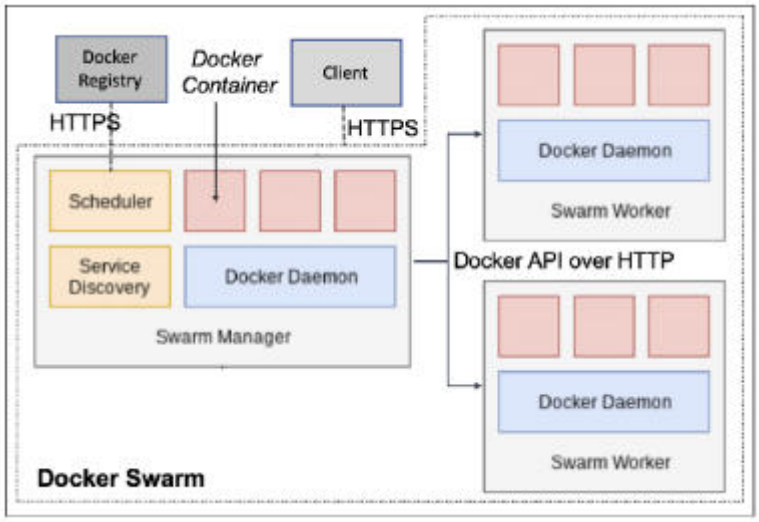

Рисунок 1.7 – Архітектура платформи оркестрації Docker Swarm

1.7 Визначення вимог

Для реалізації обчислювальної платформи з організацією збору, зберігання, аналізу IoT-даних у туманному обчислювальному середовищі необхідний наступний набір ПЗ та віртуального обладнання:

– фреймворк реалізації розподіленої обробки даних Apache Spark версії 3.2.1;

– розподілений програмний брокер повідомлень Apache Kafka версії 3.1.0;

– набір серверів (4 CPU, 8 RAM) із попередньо встановленою ОС Ubuntu Server версії 20.04.4 LTS (Focal Fossa);

– середовище розробки PyCharm Professional версії 2021.3.

1.7.1 Функціональні вимоги

До переліку функціональних вимог обчислювальної платформи належать:

– авторизація у платформі;

– додавання та ініціалізація кластера;
– створення необхідного ресурсу;
– видалення необхідного ресурсу;
– горизонтальне масштабування ресурсу;
– отримання списку запущених обчислювальних ресурсів;
– отримання інформації про стан вузлів кластера (завантаження CPU, RAM).

1.7.2 Нефункціональні вимоги

Як інструменти розробки були обрані:

– мова програмування Python 3;
– веб-фреймворк мови Python Flask;
– платформа контейнеризації Docker;
– платформа оркестрації Kubernetes;
– платформа складання та доставки програмного забезпечення GitHub Actions;
– платформа хмарного зберігання коду GitHub;

Для реалізації обчислювальної платформи необхідно надати доступ до особистого кабінету провайдера хмарних сервісів Cloud Solution з можливістю замовляти послугу надання кластера Kubernetes на необхідну кількість обчислювальних вузлів.

Для реалізації обчислювальної платформи було обрано мікросервісну архітектуру.

До основних необхідних переваг мікросервісної архітектури належать:

– незалежність інструментів та платформ розробки;
– можливість горизонтального масштабування реплікацій мікросервісів;
– високий рівень доступності розроблюваного рішення;
– спрощена можливість підтримки та доопрацювання окремих мікросервісів.

Як реляційна система управління базою даних була обрана PostgreSQL версії 14.3.

1.8 Висновки до першого розділу

В цьому розділі досліджено предметну область, зокрема описано поняття IoT та IIoT. Розглянуто концепції хмарних та туманних обчислень.

Описано особливості контейнерних віртуалізації та оркестрації, особливу увагу приділено Kubemetes та Docker Swarm, які є лідерами рішень контейнерної оркестрації.

У ході виявлення вимог було визначено функціональні та нефункціональні вимоги, у тому числі:

- визначено інструменти розробки;
- визначено інфраструктуру розробки;
- визначено архітектуру обчислювальної платформи;
- обрано систему управління базою даних.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПЛАТФОРМИ

Для опису розгортання кластера, розгортання обчислювальної платформи, компонентів платформи, варіантів використання платформи та мікросервісів, а також складових контейнерів була використана уніфікована мова моделювання UML [15], призначена для спрощеного проектування та розробки ПЗ.

2.1 Розгортання кластера

Для опису розгортання кластера використовуємо діаграму розгортання UML, наведену на рисунку 2.1.

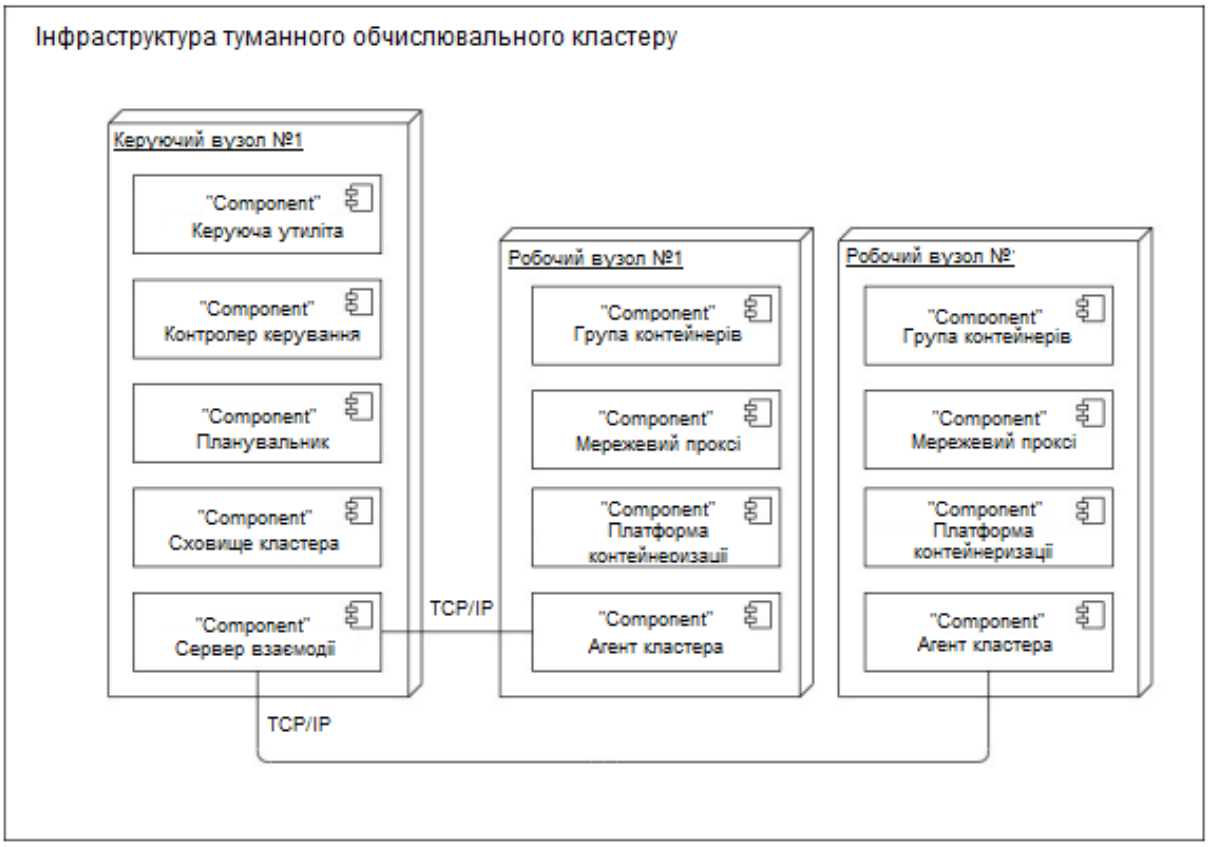

Рисунок 2.1 – Діаграма розгортання туманного обчислювального кластера

З рис. 2.1 можемо бачити, що кластер інфраструктури розгортання платформи містить керуючі та робочі вузли.

Керуючі вузли кластера містять:

– керуючу утиліту;
– контролер управління;
– планувальник ресурсів;
– системне сховище;
– сервер взаємодії через API.

Робочі вузли кластера містять:

– агент кластера;
– платформу контейнеризації;
– мережевий проксі-сервер;
– групу контейнерів (група може складатися як із множини, так і з одного контейнера).

2.2 Розгортання обчислювальної платформи

Для опису розгортання обчислювальної платформи скористаємося мовою UML, яка надає нотацію щодо опису діаграми розгортання. Діаграма розгортання обчислювальної платформи продемонстрована на рисунку 2.2.

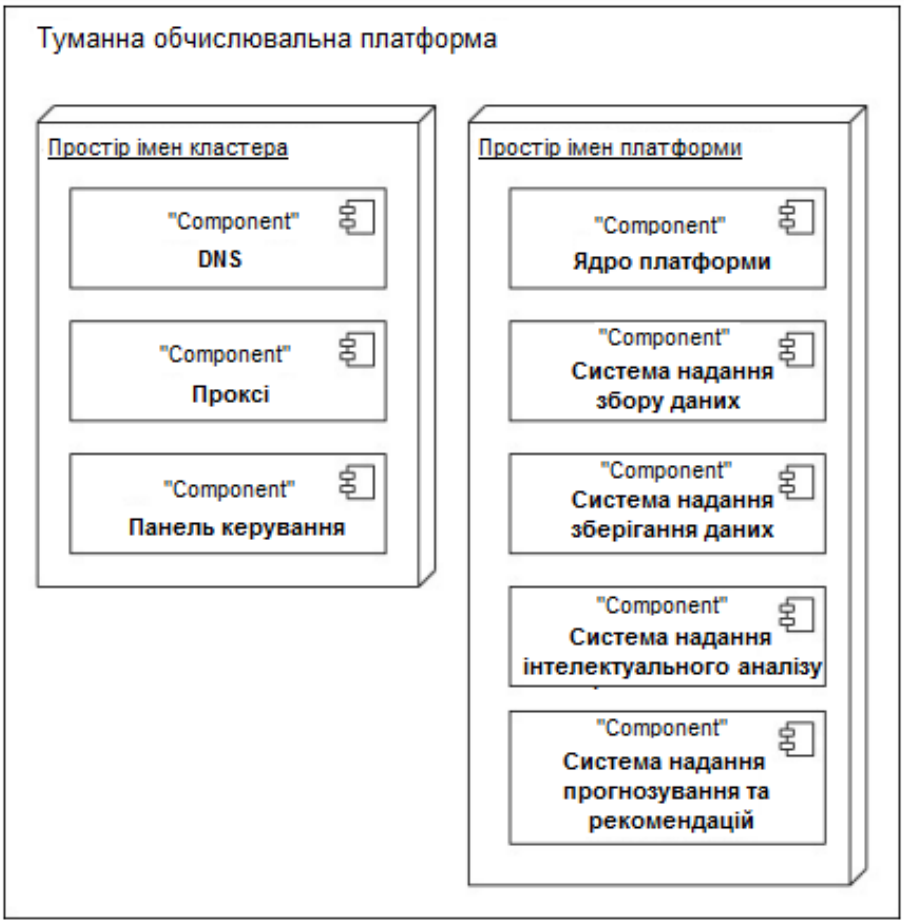

Рисунок 2.2 – Діаграма розгортання обчислювальної платформи

З рис. 2.2 можемо бачити, що в мінімальній конфігурації обчислювальна платформа складається з:

– системи (мікросервісу) надання розподіленого ПЗ для збирання даних;

– системи (мікросервісу) надання розподіленого ПЗ для зберігання даних;

– системи (мікросервісу) надання розподіленого ПЗ для інтелектуального аналізу даних.

2.3 Компоненти платформи

Для опису компонентів обчислювальної платформи використовуємо діаграму компонентів нотації мови UML. Діаграма компонентів обчислювальної платформи відображена на рисунку 2.3.

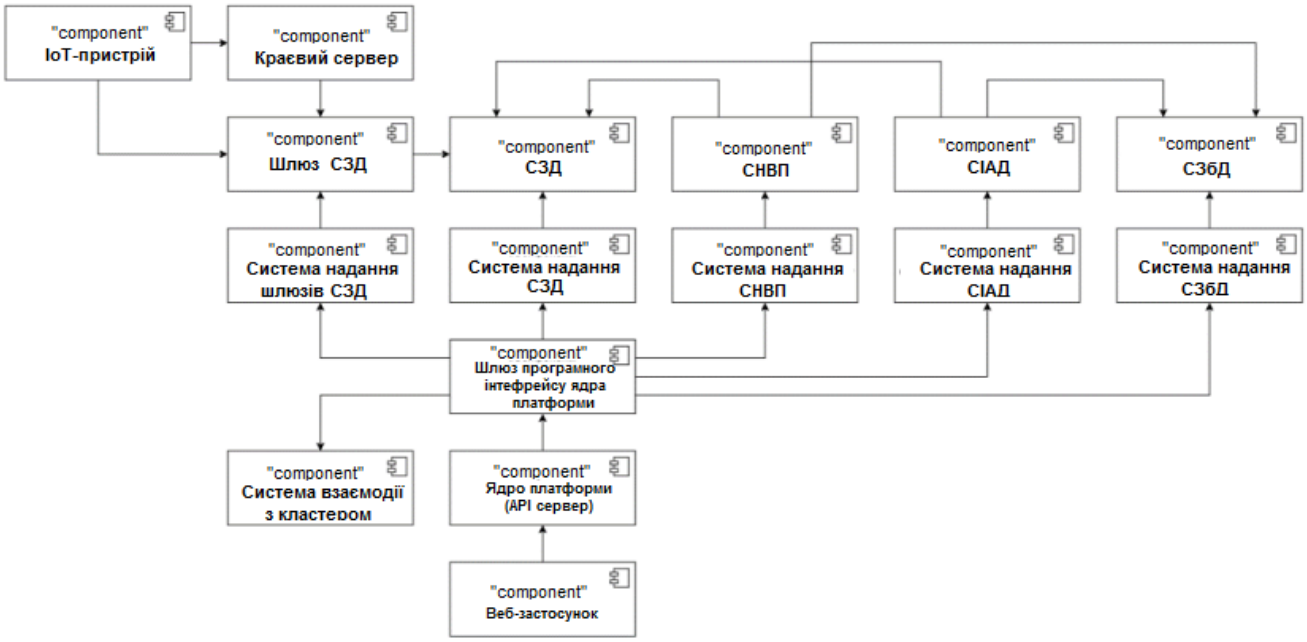

Рисунок 2.3 – Діаграма компонентів обчислювальної платформи

З діаграми компонентів бачимо, що платформа складається з:

– ядра платформи;

– шлюзу програмного інтерфейсу маршрутизації запитів;

– утиліти взаємодії із кластером;

– системи (мікросервісу) надання розподіленого ПЗ для збирання даних;

– системи (мікросервісу) надання розподіленого ПЗ для зберігання даних;

– системи (мікросервісу) надання розподіленого ПЗ для проведення інтелектуального аналізу даних;

– системи збирання даних (СЗД);

– системи інтелектуального аналізу даних (СІАД);

– системи зберігання даних (СЗбД);

– системи надання шлюзу взаємодії з IoT-пристроями та крайовими серверами (СНВП);

– шлюзу взаємодії з влаштування IoT та крайовими серверами;

– влаштування IoT;

– крайові сервери.

2.4 Варіанти використання платформи

Для опису варіантів використання обчислювальної платформою скористаємося діаграмою прецедентів від UML. Діаграма прецедентів обчислювальної платформи зображена на рисунку 2.4.

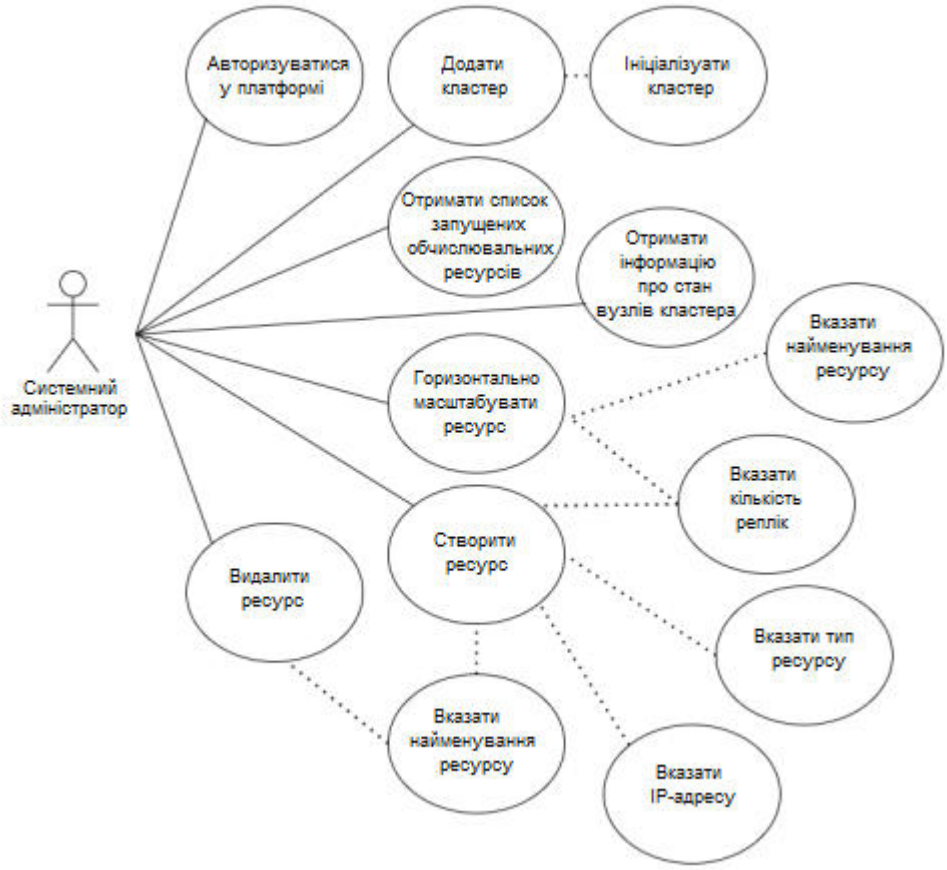

Рисунок 2.4 – Діаграма прецедентів обчислювальної платформи

З рис. 2.4 видно, що платформа надає такі можливості, як:

– авторизація у платформі;
– додавання та ініціалізація кластера;
– створення необхідного ресурсу;
– видалення необхідного ресурсу;
– горизонтальне масштабування ресурсу;
– отримання списку запущених обчислювальних ресурсів;
– отримання інформації про стан вузлів кластера (завантаження CPU,

RAM).

2.5 Опис контейнерів

Для опису складових компонентів кожного мікросервісу, що працює всередині контейнера, скористаємося можливостями спеціалізованої мови UML. На рисунку 2.5 представлена діаграма пакетів опису складових контейнерів мікросервісів.

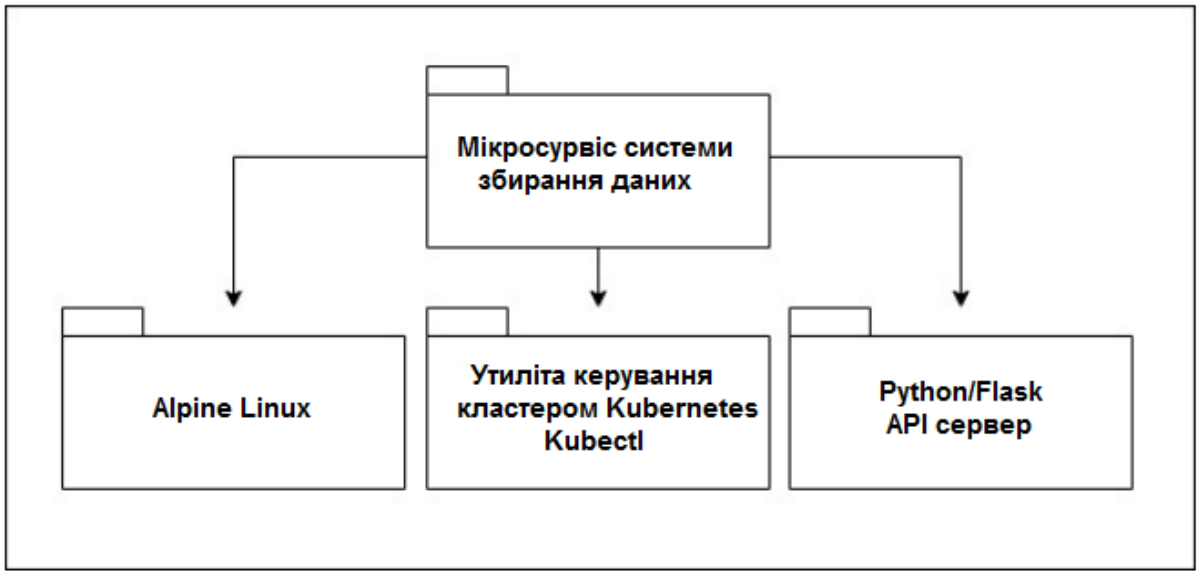

Рисунок 2.5 – Діаграма пакетів опису складових контейнерів мікросервісів

З рис. 2.5 бачимо, що діаграма пакетів описує такі компоненти, як:

– Alpine Linux - дистрибутив Linux, орієнтований на безпеку, легковаговість та невибагливість до ресурсів;

– утиліту управління кластером kubectl, за допомогою якої відбувається обробка запитів на мікросервіс;

– Python/flask api сервер, призначений для обробки запитів.

2.6 Варіанти використання мікросервісу збирання

Для опису можливостей мікросервісів, що надаються, скористаємося

нотацією UML.

На рисунку 2.6 представлена діаграма прецедентів мікросервісу збирання даних.

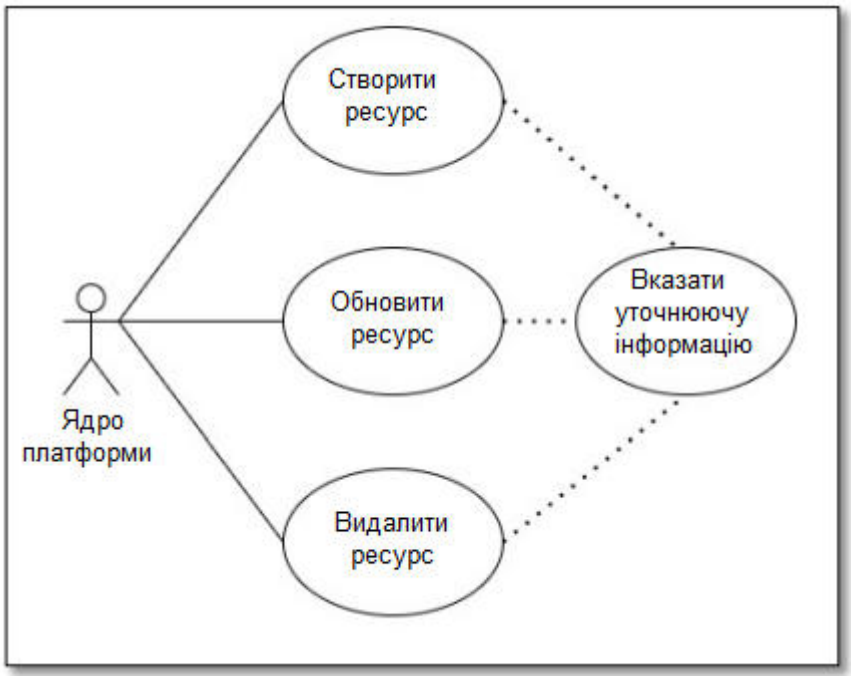

Рисунок 2.6 – Діаграма прецедентів мікросервісу збирання даних

З рис. 2.6, , бачимо, що мікросервіс забезпечує можливості:

– створення ресурсу;
– оновлення ресурсу;
– видалення ресурсу

2.7 Варіанти використання мікросервісу зберігання

Для опису функціональних можливостей мікросервісу зберігання даних використовуємо мову моделювання UML.

На рисунку 2.7 представлено діаграму прецедентів мікросервісу зберігання даних.

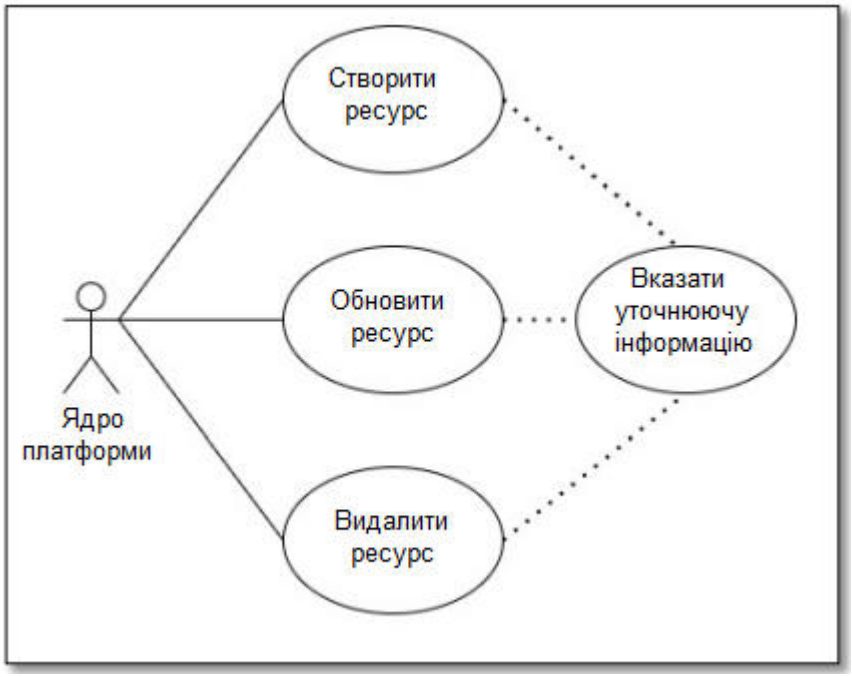

Рисунок 2.7 – Діаграма прецедентів мікросервісу зберігання даних

З діаграми прецедентів (рис. 2.7) бачимо, що мікросервіс надає такі можливості, як:

– створення ресурсу;
– оновлення ресурсу;
– видалення ресурсу

2.8 Варіанти використання мікросервісу аналізу

Для опису використання мікросервісу інтелектуального аналізу даних скористаємось мовою UML.

На рисунку 2.9 представлено діаграму прецедентів мікросервісу інтелектуального аналізу даних.

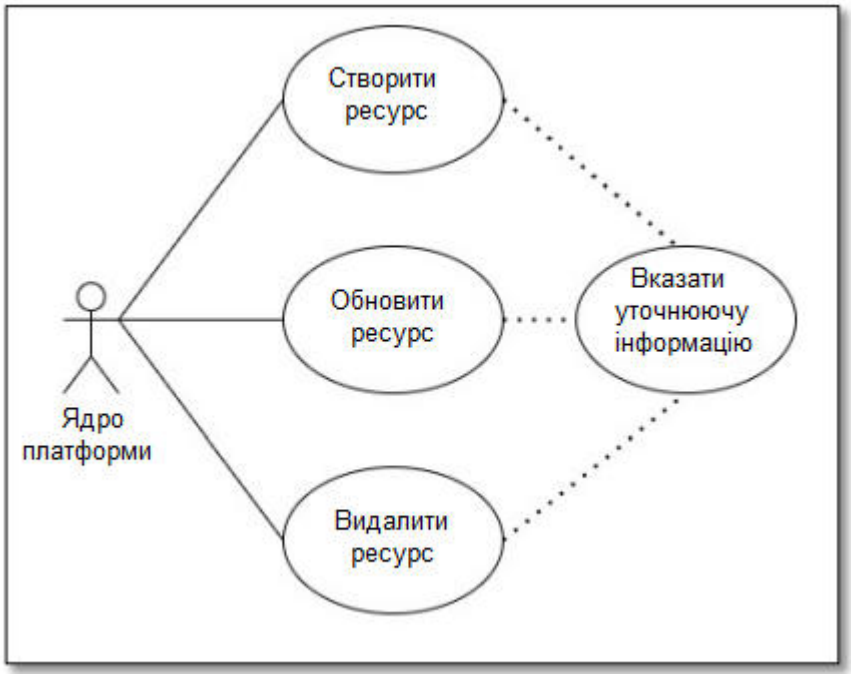

Рисунок 2.8 – Діаграма прецедентів мікросервісу мікросервісу аналізу даних

З рис. 2.8 бачимо, що діаграма прецедентів описує такі можливості мікросервісу:

– створення ресурсу;
– оновлення ресурсу;
– видалення ресурсу.

2.9 Проектування бази даних

З метою проектування структури бази даних необхідно скласти схему. На рисунку 2.9 зображено схему бази даних.

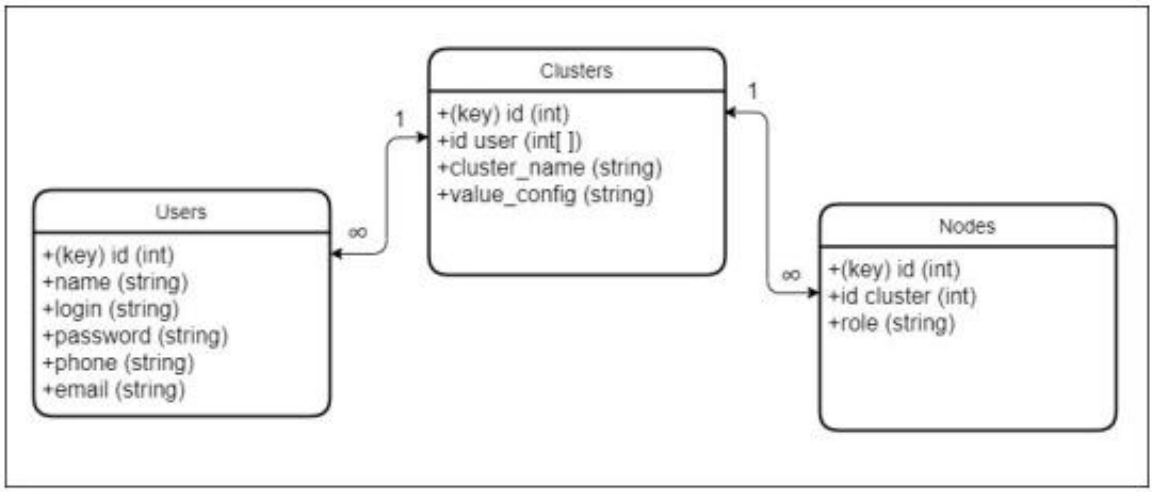

Рисунок 2.9 – Схема даних

Для опису даних надамо перелік таблиць бази даних (таблиці 2.1 – 2.4).

Таблиця 2.1 – Список таблиць бази даних

Назва таблиці	**Призначення**
Users	Дані про користувачів
Clusters	Дані про кластери
Nodes	Дані про вузли

У таблицях 2.2 – 2.4 описано атрибути полів таблиць.

Таблиця 2.2 – Атрибути таблиці "Users"

Найменування	**Тип даних**	**Призначення**
Id	int(64)	Унікальний ідентифікатор
Name	Text(необмеж.)	Ім'я (ПІБ)
Login	Text(необмеж.)	Логін
Password	Text(необмеж.)	Пароль
Phone	Text(необмеж.)	Контактний телефон
Email	Text(необмеж.)	Електронна пошта

Таблиця 2.3 – Атрибути таблиці "Clusters"

Найменування	**Тип даних**	**Призначення**
Id	Int(64)	Унікальний ідентифікатор
ID користувача	Int(64)	Ідентифікатор користувача
Cluster_name	Text(неогр.)	Ім'я кластера
Value_config	Text(неогр.)	Конфігураційні дані авторизації

Таблиця 2.4 – Атрибути таблиці "Nodes"

Найменування	**Тип даних**	**Призначення**
Id	Int(64)	Унікальний ідентифікатор
Id_cluster	Int(64)	Ідентифікатор кластеру
role	Text(необмеж.)	Роль вузла

2.10 Реалізація платформи

Реалізація обчислювальної платформи передбачає використання мікросервісної архітектури. Основними складовими компонентами платформи є ядро та мікросервіси надання розподіленого ПЗ.

2.10.1 Блок-схема алгоритму надання розподіленого програмного забезпечення

На рисунку 2.10 представлено блок-схему фрагмента алгоритму розгортання ПЗ мікросервісом.

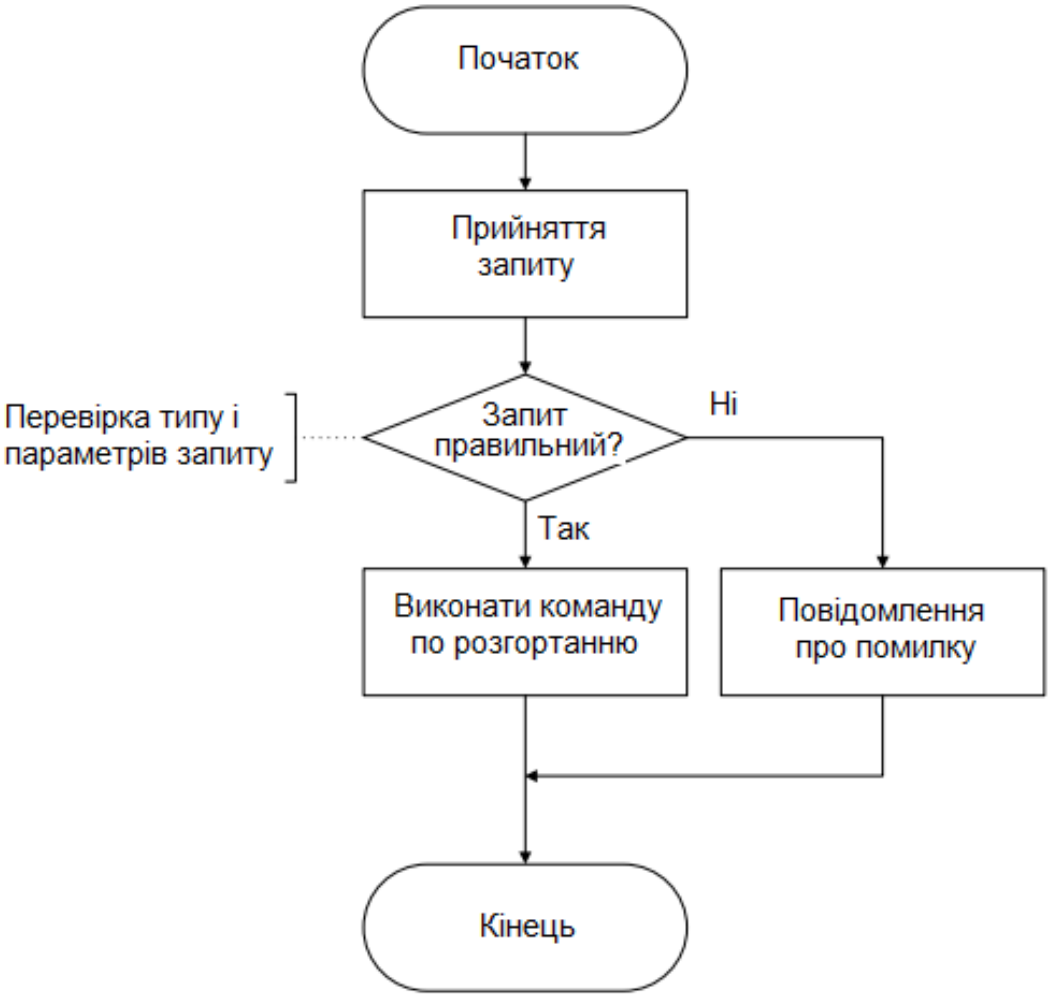

Рисунок 2.10 – Блок-схема фрагмента алгоритму розгортання мікросервісу

2.10.2 Блок-схема ядра платформи

На рисунку 2.11 представлено блок-схему фрагмента алгоритму функції виконання запиту до мікросервісів від ядра платформи.

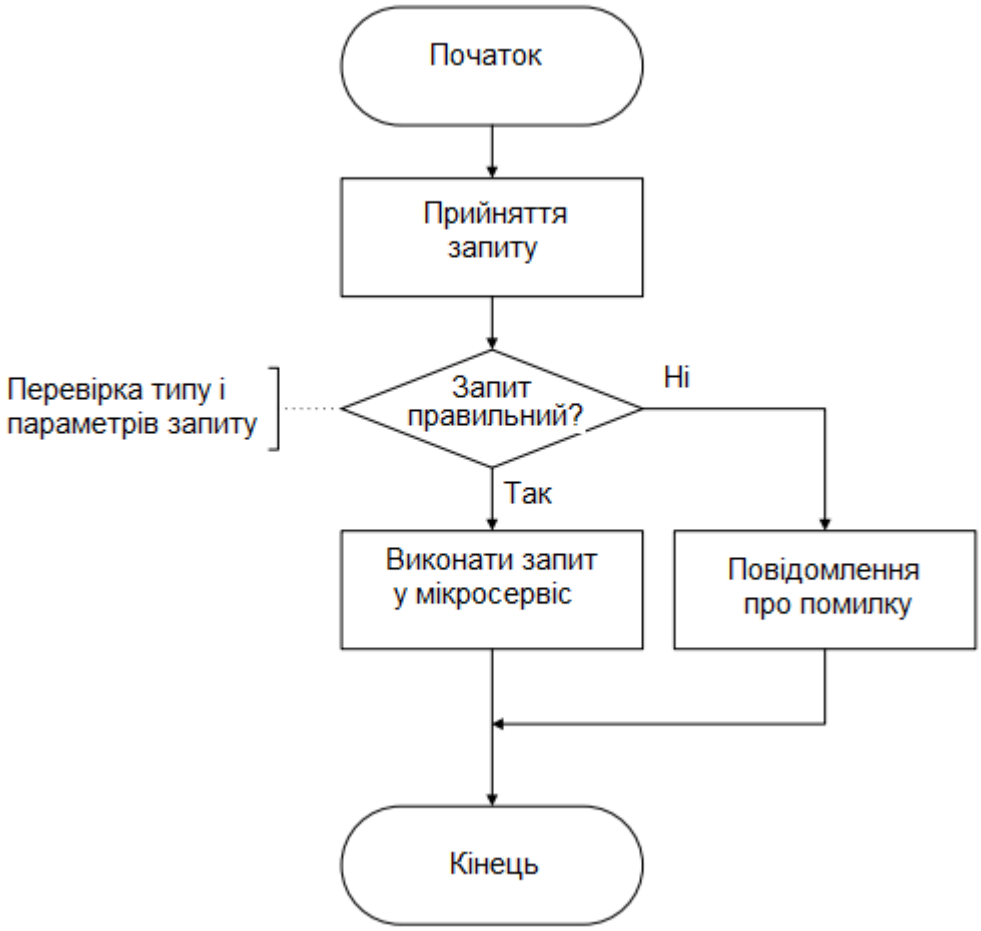

Рисунок 2.11 – Фрагмент блок-схеми ядра платформи

2.11 Висновки до другого розділу

У ході проектування були описані такі деталі обчислювальної платформи, як розгортання кластера та обчислювальної платформи, компоненти та варіанти використання платформи, контейнери та їх прецеденти. Також було спроектовано схему бази даних з описом даних.

У ході реалізації обчислювальної платформи було розроблено набір мікросервісів надання розподіленого ПЗ для збирання, зберігання та аналізу даних.

Також було розроблено ядро платформи, що виконує роль публічного API-сервера взаємодії користувачів та платформи у туманному обчислювальному середовищі.

3 ТЕСТУВАННЯ

Тестування є значною стадією життєвого циклу розробки ПЗ. Правильно спроектована процедура тестування програмних систем дозволить побачити помилки розробки та вчасно зробити рішення для їх виправлення.

3.1 Поняття тестування, його види

Основна ідея проведення тестування - це не тільки віднайти і детектувати помилки і вади, але й гарантувати високу якість та надійність ПЗ, котре створюється, перед його власне впровадженням та використанням кінцевим споживачем.

Тестування - це ітеративний процес, так як виправлення всього лиш однієї помилки здатне призвести до детектування інших, вже знчано складніших проблем чи навіть спричиняти появу нових. Фахівці-тестувальники вибирають певні стратегії, котрі якнайкраще відповідають функціям і вимогам програмного продукту. Такі стратегії можуть містити ручне чи автоматизоване тестування, тестування так званих «білої і чорної скриньки».

До видів тестування ПЗ належать зокрема такі (рис. 3.1):

- функціональне;
- інтеграційне;
- системне;
- тестування продуктивності;
- регресійне;
- модульне;
- тестування безпеки.

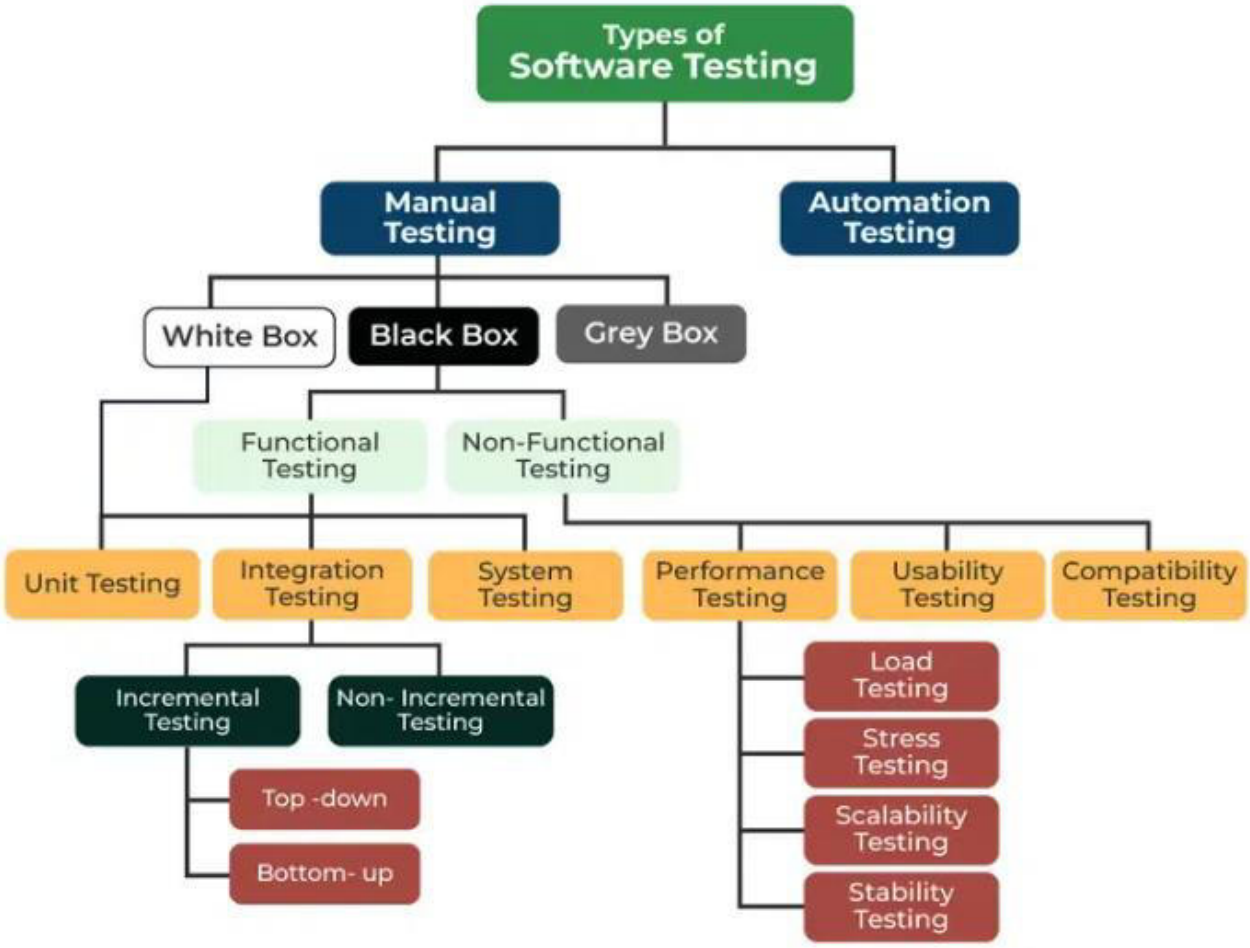

Рисунок 3.1 – Види тестування ПЗ

Найважливішим для реалізації програмних систем є функціональне тестування. Зокрема, при використанні в програмній системі, що розробляється, функцій мережевої взаємодії, функціональне тестування розширюється до тестування функцій і програмних інтерфейсів.

Рівень програмних інтерфейсів API є одним із найбільш важливих елементів будь-якої системи, яка має мережеву взаємодію.. Програмний інтерфейс виконує роль каналу з'єднання між клієнтом та сервером або одним мікросервісом з іншим. Тестування API дозволить виявити неправильні чи некоректні відповіді, відстежити працездатний стан як програмного клієнта, і програмного сервера.

Врешті решт якісно спланований та втілений процес тестування значно покращує якість ПЗ і суттєво зменшує імовірність збоїв в його функціонуванні.

3.2 Методологія тестування

Для виконання тестування скористаємося платформою Postman (рисунок 3.2), яка дозволяє розробникам ПЗ проектувати, створювати та тестувати виконання запитів до програмних інтерфейсів системи, що розробляється. Платформи тестування API Postman дозволяє вводити необхідну адресу, вказувати вид запиту, відстежувати відповідь, що отримується, і його статус.

Рисунок 3.2 – Логотип платформи Postman

Postman допомагає перевіряти, як «спілкування» через API відбувається, дає змогу надсилати запити і переглядати відповіді. Програмний засіб доступний у двох виконаннх - як настільна програма та як веб-застосунок. Для зручності проведення тестування Postman дозволяє зберігати і групувати запити, що часто повторюються.

Postman здатен перетворює складні запити до сервера на цілком зрозуміле візуальне виконання. Іншими словами, фактично це ніби браузер для API-запитів, проте значно розумніший.

3.3 Проведення процедури тестування

Для проведення тестування програмних інтерфейсів виконаємо почергово ряд запитів.

POST запит до /api/v0.1/dcs/ (рисунок 3.1):

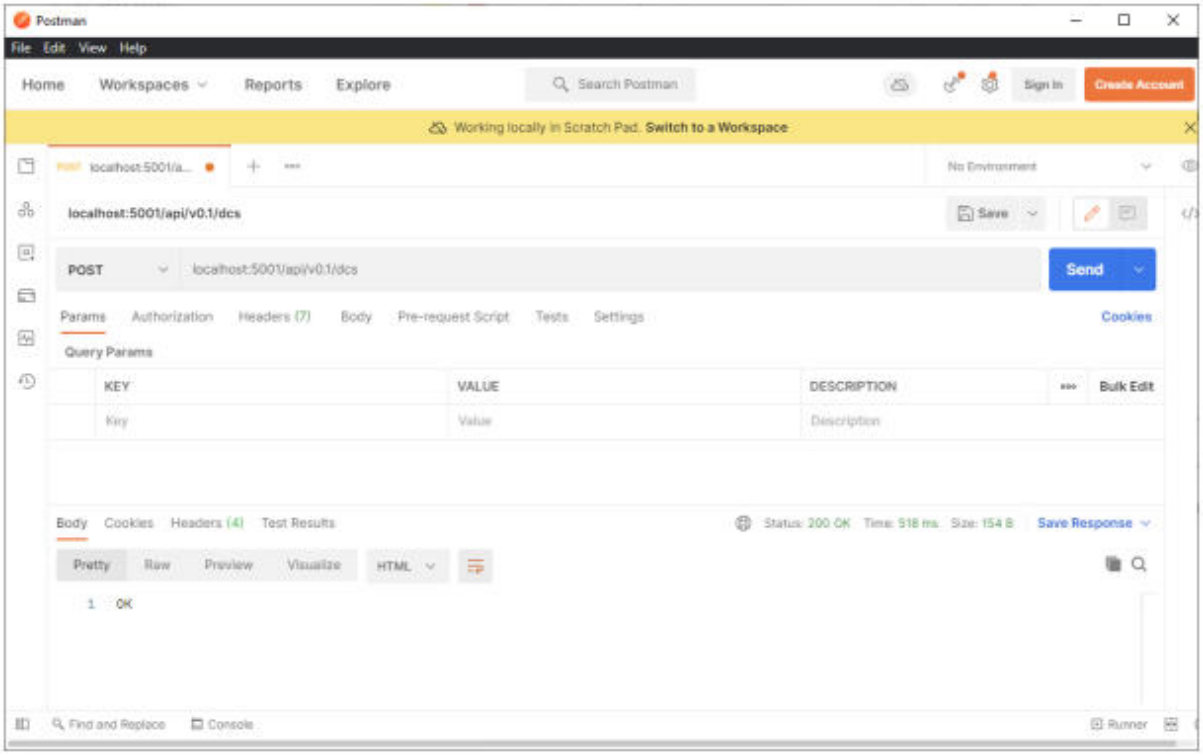

Рисунок 3.1 – POST запит до /api/v0.1/dcs/

DELETE запит до /api/v0.1/dcs (рисунок 3.2):

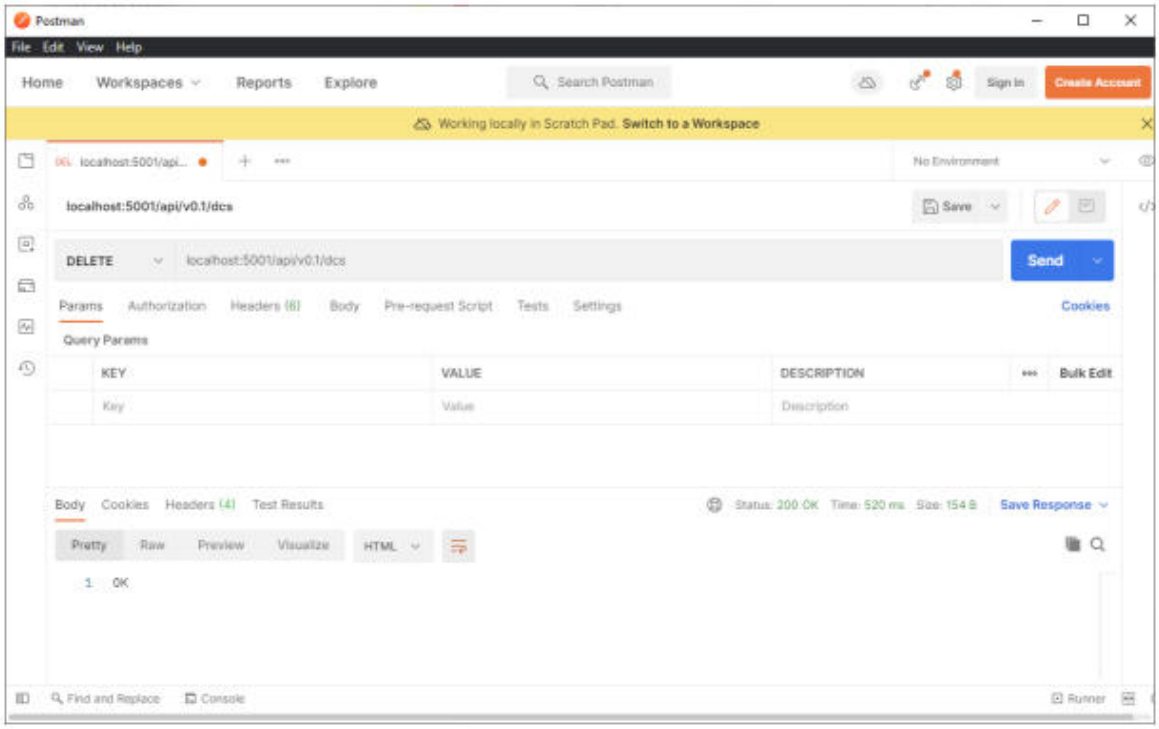

Рисунок 3.2 – DELETE запит до /api/v0.1/dcs/

POST запит до /api/v0.1/dss/ (рисунок 3.3):

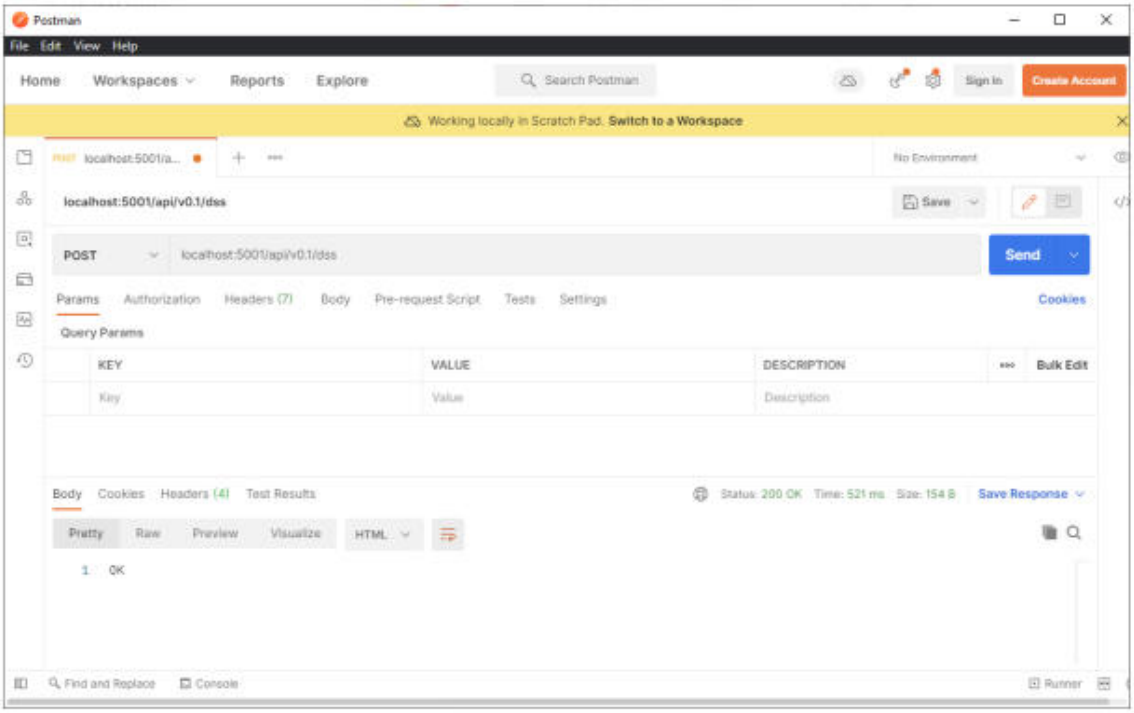

Рисунок 3.3 – POST запит до /api/v0.1/dss/

DELETE запит до /api/v0.1/dss/ (рисунок 3.4):

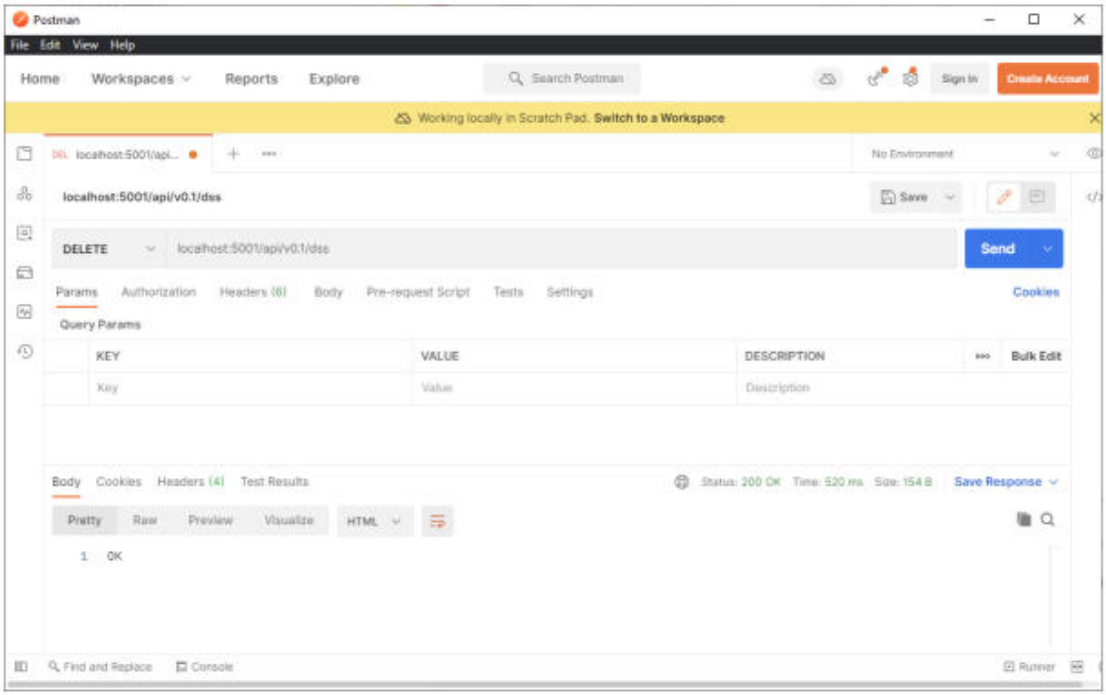

Рисунок 3.4 – DELETE запит до /api/v0.1/dss/

PUT запит до /api/v0.1/dss/ (рисунок 3.5):

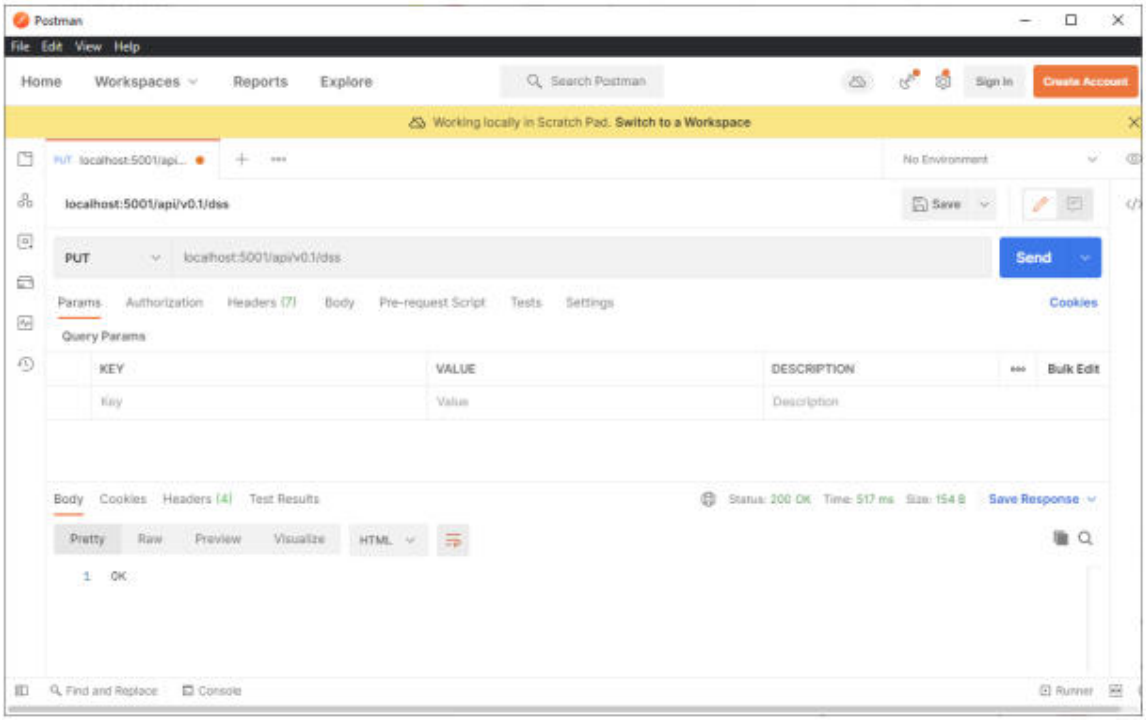

Рисунок 3.5 – PUT запит до /api/v0.1/dss/

POST запит до /api/v0.1/dms/ (рисунок 3.6):

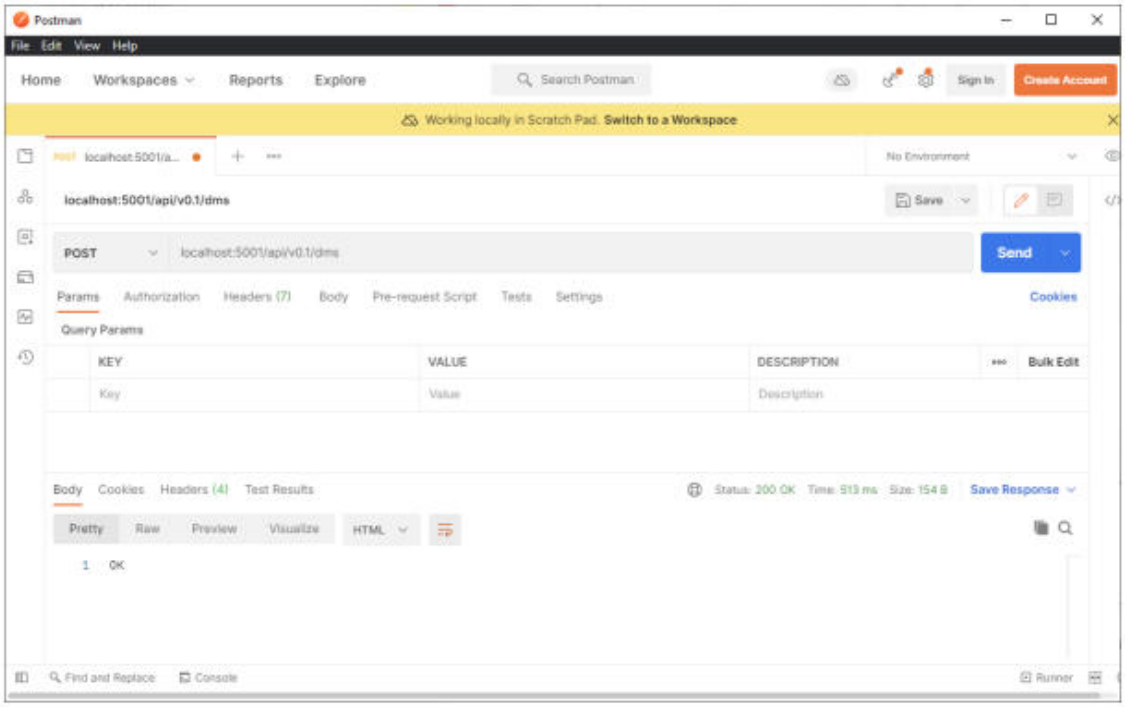

Рисунок 3.6 – POST запит до /api/v0.1/dms/

PUT запит до /api/v0.1/dms/ (рисунок 3.7):

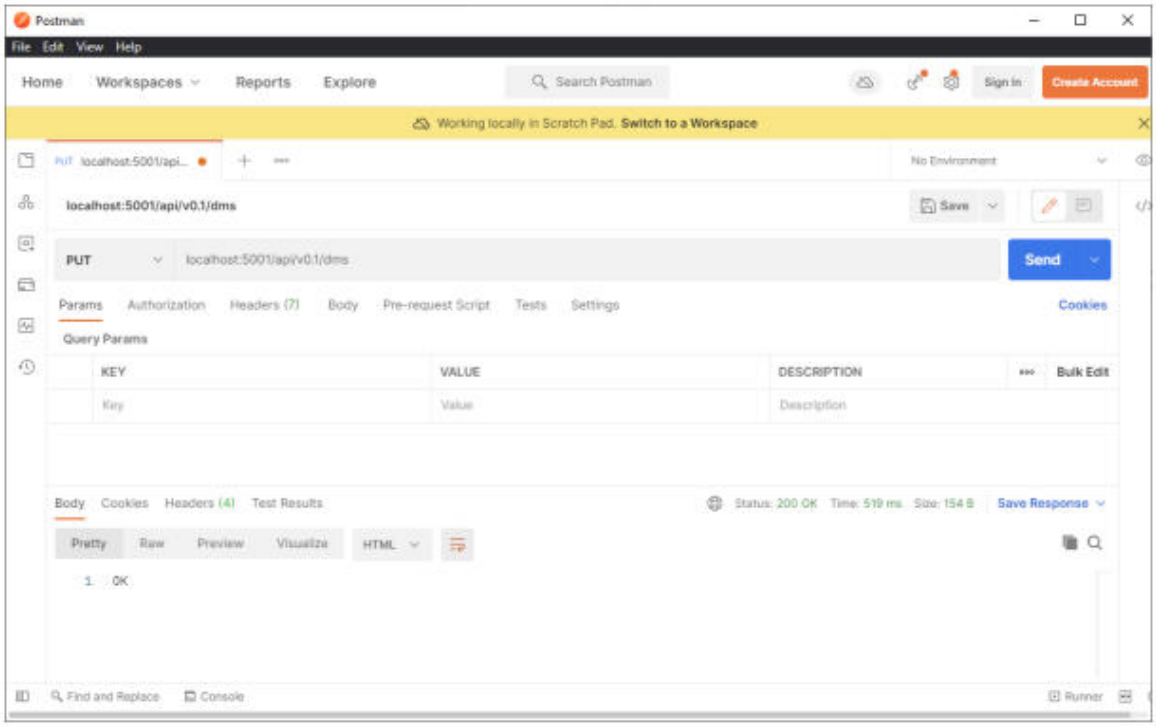

Рисунок 3.7 – PUT запит до /api/v0.1/dms/

DELETE запит до /api/v0.1/dms/ (рисунок 3.8):

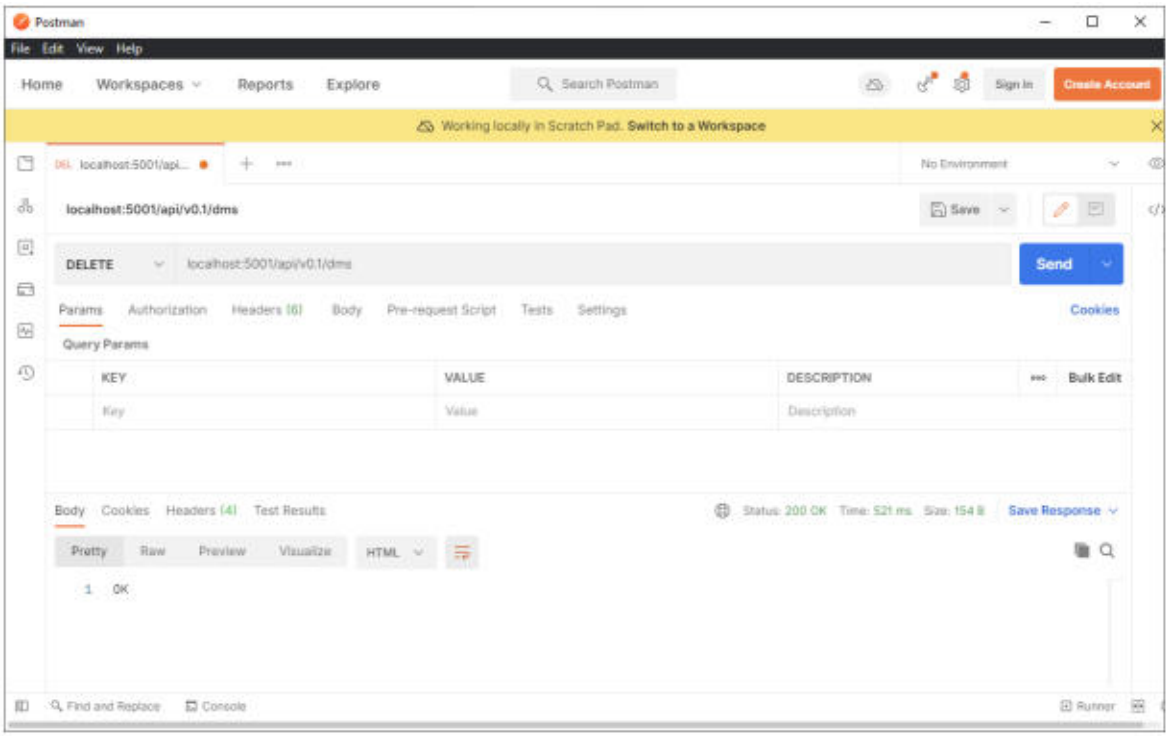

Рисунок 3.8 – DELETE запит до /api/v0.1/dms/

Як видно з рис. 3.1 – 3.8, виконані запити повернули очікувану відповідь та статус від API сервера ядра платформи.

3.4 Висновки до третього розділу

Описано поняття та види тестування. Запропоновано методику проведення тестування, яка базується на використанні спеціалізованого ПЗ – Postman.

У ході проведення тестування було виконано запити до ядра платформи. Усі запити повернули очікуваний результат та статус.

Таким чином можна стверджувати, що тестування пройшло успішно.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

В цьому розділі необхідно проаналізувати важливі питання, котрі стосуються безпеки життєдіяльності та основ охорони праці.

4.1 Класифікація шкідливих та небезпечних виробничих факторів

Шкідливий виробничий фактор – небажане явище, яке супроводжує виробничий процес і вплив якого на працюючого може призвести до погіршення самопочуття, зниження працездатності, захворювання, виробничо зумовленого чи професійного, і навіть смерті, як результату захворювання. Небезпечний виробничий фактор – небажане явище, яке супроводжує виробничий процес і дія якого за певних умов може призвести до травми або іншого раптового погіршення здоров’я працівника (гострого отруєння, гострого захворювання) і навіть до раптової смерті [24].

Поділ несприятливих чинників виробничого середовища на шкідливі та небезпечні зумовлене різним характером їх дії на людський організм, тим, що вони потребують різних заходів та засобів для боротьби з ними та профілактики викликаних ними ушкоджень, а також рядом причин організаційного характеру. В той же час між шкідливими та небезпечними виробничими факторами інколи важко провести чітку межу. Один і той же чинник може викликати травму і профзахворювання (наприклад, високий рівень іонізуючого або теплового випромінювання може викликати опік або навіть призвести до миттєвої смерті, а довготривала дія порівняно невисокого рівня цих же факторів – до хвороби; пилинка, що потрапила в око, спричиняє травму, а пил, що осідає в легенях, – захворювання, що зветься пневмоконіоз). Через це всі несприятливі виробничі чинники часто розглядаються як єдине поняття – небезпечний та шкідливий виробничий фактор (НШВФ) [24]. За своїм походженням та природою дії всі НШВФ можна поділити на 5 груп: фізичні, хімічні, біологічні, психофізіологічні та соціальні До фізичних НШВФ відносяться машини та механізми або їх елементи, а

також вироби, матеріали, заготовки тощо, які рухаються або обертаються; конструкції, які руйнуються; системи, устаткування або елементи обладнання, які знаходяться під підвищеним тиском; підвищена запиленість та загазованість повітря; підвищена або понижена температура повітря, поверхонь приміщення, обладнання, матеріалів; підвищені рівні шуму, вібрації, ультразвуку, інфразвуку; підвищений або понижений барометричний тиск та його різкі коливання; підвищена та понижена вологість; підвищена швидкість руху та підвищена іонізація повітря; підвищений рівень іонізуючих випромінювань; підвищене значення напруги в електричній мережі; підвищені рівні статичної електрики, електромагнітних випромінювань; підвищена напруженість електричного, магнітного полів; відсутність або нестача світла; недостатня освітленість робочої зони; підвищена яскравість світла; понижена контрастність; прямий та віддзеркалений блиск; підвищена пульсація світлового потоку; підвищені рівні ультрафіолетової та інфрачервоної радіації; гострі крайки, зачипки, шершавість на поверхні заготовок, інструментів та обладнання; розташування робочого місця на значній висоті відносно землі (підлоги); слизька підлога; невагомість.

Хімічні НШВФ:

- за характером дії на організм людини поділяються на токсичні, задушливі, наркотичні, подразнюючі, сенсибілізуючі, канцерогенні, мутагенні та такі, що впливають на репродуктивну функцію;

- за шляхами проникнення в організм людини поділяються на такі, що потрапляють через: 1) органи дихання; 2) шлунково-кишковий тракт; 3) шкіряні покриви та слизова оболонка;

- які перебувають у різному агрегатному стані: 1)твердому 2)газоподібному 3)рідкому.

Біологічні НШВФ – це: - патогенні мікроорганізми (бактерії, віруси, рикетсії, спірохети, грибки, найпростіші) та продукти їхньої життєдіяльності; - макроорганізми (тварини та рослини) та продукти їхньої життєдіяльності. До психофізіологічних НШВФ відносяться фізичні (статичні та динамічні) перевантаження і нервово-психічні перевантаження (розумове перенапруження,

перенапруження аналізаторів, монотонність праці, емоційні перевантаження). 6 Соціальні НШВФ – це неякісна організація роботи, понаднормова робота, змушеність праці в колективі з поганими відносинами між його членами, соціальна ізольованість з відривом від сім_ї, зміна біоритмів, незадоволеність роботою, фізична та/або словесна образа та її ризик, насильство та його ризик. Один і той же НШВФ за природою своєї дії може належати водночас до різних груп.

4.2 Вплив вібрації на людину

Вібрація - це механічні коливання пружних тіл або коливальні рухи механічних систем. Для людини вібрація є видом механічного впливу, який має негативні наслідки для організму [25].

Причиною появи вібрації є неврівноважені сили та ударні процеси в діючих механізмах. Створення високопродуктивних потужних машин і швидкісних транспортних засобів при одночасному зниженні їх матеріалоємності неминуче призводить до збільшення інтенсивності і розширення спектру вібраційних та віброакустичних полів. Цьому сприяє також широке використання в промисловості і будівництві високоефективних механізмів вібраційної та віброударної дії.

Дія вібрації може приводити до трансформування внутрішньої структури і поверхневих шарів матеріалів, зміни умов тертя і зносу на контактних поверхнях деталей машин, нагрівання конструкцій. Через вібрацію збільшуються динамічні навантаження в елементах конструкцій, стиках і сполученнях, знижується несуча здатність деталей, ініціюються тріщини, виникає руйнування обладнання. Усе це приводить до зниження строку служби устаткування, зростання імовірності аварійних ситуацій і зростання економічних витрат. Вважають, що 80% аварій в машинах і механізмах здійснюється внаслідок вібрації. Крім того, коливання конструкцій часто є джерелом небажаного шуму. Захист від вібрації є складною і багатоплановою в науково-технічному та важливою у соціальноекономічному відношеннях проблемою нашого суспільства [25].

Вплив вібрації на людину залежить від її спектрального складу, напрямку дії, прикладення, тривалості впливу, а також від індивідуальних особливостей людини. При оцінці вібраційного впливу потрібно враховувати, що коливальні процеси притаманні живому організму. В основі серцевої діяльності і кровообігу та біострумів мозку лежать ритмічні коливання. Внутрішні органи людини можна розглядати як коливальні системи з пружними зв'язками. Частоти їх власних коливань лежать у діапазоні 3..6 Гц. Частоти власних коливань плечового пояса, стегон і голови щодо опорної поверхні (положення стоячи) складають 4...6 Гц, голови щодо плечей (положення сидячи) 25...30 Гц.

При впливі на людину зовнішніх коливань (хитавиці, струсів, вібрації) відбувається їхня взаємодія з внутрішніми хвильовими процесами, виникнення резонансних явищ. Так, зовнішні коливання частотою менш 0,7 Гц утворюють хитавицю і порушують у людини нормальну діяльність вестибулярного апарата. Інфразвукові коливання (менш 16 Гц), впливаючи на людину, пригнічують центральну нервову систему, викликаючи почуття тривоги, страху. При певній інтенсивності на частоті 6..7 Гц інфразвукові коливання, втягуючи у резонанс внутрішні органи і систему кровообігу, здатні викликати травми, розриви артерій, тощо [25].

Вібрація, що діє на людину, має широкий діапазон – від десятих часток одного до декількох тисяч Гц. Характерними ознаками шкідливого впливу вібрації на людину є можливі зміни у функціональному стані: підвищена втома, збільшення часу моторної реакції, порушення вестибулярної реакції. Медичними дослідженнями встановлено, що вібрація є подразником периферичних нервових закінчень, розташованих на ділянках тіла людини, що сприймають зовнішні коливання. Адекватним фізичним критерієм оцінки її впливу на організм людини є коливальна енергія, що виникає на поверхні контакту, а також енергія, поглинена тканинами і передана опорно-руховому апарату та іншим органам. У результаті впливу вібрації виникають нервовосудинні розлади, ураження кістково-суглобної та інших систем організму. Відзначаються, наприклад, зміни функції щитовидної залози, сечостатевої системи, шлунково-кишкового тракту. Так, медичні

дослідження показали , що у працюючих в умовах вібрації відбуваються значні зміни кістковосуглобної системи, які виражаються у функціональній перебудові кісткової тканини, регіональному остеопорозі, кістовидних утвореннях у кістках, асептичному некрозі кісток, хронічних переломах. Відзначається, що терміни виникнення змін у кістках у працівників вібраційних професій коливається в межах від 6-8 місяців до 2-5 років.

Шкідливість вібрації збільшується при одночасному впливі на людину таких факторів, як знижена температура, підвищений шум, запиленість повітря, тривала статична напруга тощо. Сучасна медицина розглядає виробничу вібрацію як могутній стрес-фактор, що має негативний вплив на психомоторну працездатність, емоційну сферу і розумову діяльність, підвищує ймовірність виникнення різних захворювань і нещасних випадків. Особливо небезпечний тривалий вплив вібрації для жіночого організму. Широкий комплекс патологічних відхилень, викликаний впливом вібрації на організм людини, кваліфікується як віброзахворювання [25].

Вібрація як фізичний чинник виробничого середовища спостерігається в металообробній, гірничодобувній, металобудівній, машинобудівній, авіаційній та інших галузях народного господарства. Джерелом вібрації можуть бути різні механізми, вібраційне устаткування, віброінструменти, акустичні системи, транспортні та сільськогосподарські машини.

Загальна вібрація поділяється на транспортну вібрацію, яка діє на людину на робочих місцях в транспортних засобах (трактори сільськогосподарські та промислові, самохідні сільськогосподарські машини (комбайни), тягачі, грейдери ті інші); транспортно-технологічну вібрацію, яка діє на людину на робочих місцях машин з обмеженою рухливістю (екскаватори, крани промислові та будівельні, гірничі комбайни, транспорт виробничих приміщень та інші) та технологічну вібрацію, яка діє на людину на робочих місцях стаціонарних машин чи передається на робочі, де немає джерел вібрації (верстати та метало-деревообробне, пресувально-ковальське обладнання, ливарні машини, електричні машини, насосні агрегати та вентилятори, обладнання для буріння свердловин, бурові верстати, машини для тваринництва, очищення та сортування зерна (у тому числі сушарні),

обладнання промисловості будматеріалів (крім бетоноукладачів), установки хімічної та нафтохімічної промисловості та інші.

Оператори машин, які зазнають у процесі трудової діяльності впливу вібрації, підлягають попереднім та періодичним медичним оглядам відповідно до Порядку проведення медичних оглядів працівників певних категорій, затвердженого Наказом МОЗ України від 21.05.2007 р. №246. Обов'язкові попередні (під час прийняття на роботу) та періодичні (протягом трудової діяльності) медичні огляди дозволять визначити стан здоров'я працівника та можливість виконання без погіршення стану здоров'я професійних обов'язків, своєчасно виявити ранні ознаки хронічного професійного захворювання, забезпечує динамічне спостереження за станом здоров`я в умовах дії шкідливих та небезпечних факторів і трудового процесу, вирішує питання щодо можливості продовжувати роботу в умовах дії шкідливих та небезпечних факторів і трудового процесу [25].

За результатами періодичних медичних оглядів роботодавець забезпечує проведення відповідних оздоровчих заходів Заключного акта у повному обсязі та усуває причини, що призводять до професійних захворювань. Організовує проведення лабораторних досліджень умов праці на робочих місцях та вживає заходів до усунення небезпечних і шкідливих для здоров'я виробничих факторів.

4.3. Висновки до четвертого розділу

В цьому розділі проаналізовано важливі питання охорони праці та безпеки в надзвичайних ситуаціях, висвітлено питання класифікації шкідливих та небезпечних виробничих факторів та впливу вібрації на людину.

ВИСНОВКИ

У рамках виконання кваліфікаційної роботи було спроектовано та реалізовано обчислювальну платформу для збору, зберігання, аналізу даних інтернету речей у туманному обчислювальному середовищі.

Обчислювальна платформа архітектурно складає набір взаємопов'язаних мікросервісів, реалізованих мовою програмування Python та веб-фреймворку Flask із застосуванням технологій контейнерної віртуалізації Docker та контейнерної оркестрації Kubemetes.

Для досягнення мети було виконано такі завдання:

- виконано огляд літератури;
- визначено вимоги;
- спроектовано та реалізовано обчислювальну платформу;
- проведено тестування API;
- проведено аналіз результатів.

Основним результатом виконання роботи є розроблена програмна система, яка, організуючи збір, зберігання та аналіз даних з пристроїв IoT, дозволить організовувати цифрові двійники промислових підприємств.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hu P. Survey on fog computing: architecture, key technologies, applications and open issues // Journal of Network and Computer Applications. 2017. Vol. 98. P. 27–42.

2. Laghari, A. A., Wu, K., Laghari, R. A., Ali, M., & Khan, A. A. (2021). A review and state of art of Internet of Things (IoT). Archives of Computational Methods in Engineering, 1-19

3. Integration of Cloud Computing and IoT. URL: https://drive.google.com/file/d/1WNoRWlMKyvLjpricYmjZHI3L7QM0H8zF/view / (дата звернення: 01.05.2026)

4. An overview of IoT architectures, technologies, and existing open-source projects. URL: https://doi.org/10.1016/j.iot.2022.100626 (дата звернення: 01.05.2026)

5. Бойко Н.І. Хмарні обчислення та їх застосування в інформаційних системах: монографія. Київ: НТУу "КПІ", 2023. 186 с.

6. Hoque, S. Towards Container Orchestration in Fog Computing Infrastructures / S. Hoque, M.S. de Brito, T. Magedanz, A. Willner, O. Keil // IEEE 41st Annual Computer Software and Applications Conference. 2017. P. 294–299.

7. Virtualization Technology & Virtual Machine Software: What is Virtualization? URL: https://www.vmware.com/ru/solutions/virtualization.html (дата звернення: 09.05.2026) .

8. Що таке контейнеризація та в чому її переваги. URL: https://gigacloud.ua/articles/shho-take-kontejneryzacziya-ta-v-chomu-yiyi-perevagy/ (дата звернення: 10.05.2026) .

9. Віртуалізація vs контейнеризація – порівняння відмінностей. URL: https://alexhost.com/uk/faq/virtualization-vs-containerization-comparing-differences/ (дата звернення: 10.05.2026) .

10. Навіщо нам контейнеризація: переваги для DevOps. URL: https://itedu.center/ua/blog/articles/navishho-nam-kontejnerizaciya-perevagi-dlya-

devops/?srsltid=AfmBOooOLZR9oH5VNY7cbf32wlN3TwRCw0tyqYDdSOo0NReUR6uJufBs (дата звернення: 10.05.2026)

11. Docker: програмна платформа для розробників і девопсів. URL: https://brander.ua/technologies/docker / (дата звернення: 01.05.2022)

12. Turnbull, J. "Docker in Action, Second Edition." Manning Publications, Shelter Island, NY, (2020).

13. de Brito, M. S. A Service Orchestration Architecture for Fog-enabled Infrastructures / M. S. de Brito, S. Hoque, T. Magedanz R. Steinke, A. Willner, D. Nehls, O. Keils, F. Schreiner // 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC). 2017. P. 127–132.

14. Khan A. Key Characteristics of a Container Orchestration Platform to Enable a Modern Application / IEEE Cloud computing. 2017. P. 42–48.

15. UNIFIED MODELING LANGUAGE® (UML®). URL: https://www.omg.org/uml// (дата звернення: 19.05.2022)

16. Електронна документація Docker Swarm. URL: https://docs.docker.com/engine/swarm/ (дата звернення: 19.05.2022)

17. Електронна документація Apache Kafka. URL: https://kafka.apache.org/documentation/ (дата звернення: 21.05.2022)

18. Електронна документація Kubernetes. [URL: https://kubernetes.io/docs/home/ (дата звернення: 21.05.2022).

19. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів спеціальності 121 – Інженерія програмного забезпечення, всіх форм навчання / укладачі Михалик Д.М., Цуприк Г.Б., Бревус В.М. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2024. 45 с.

20. Stefanyshyn, I. , Pastukh, O., Stefanyshyn, V. , Baran, I. , Boyko, I. Robustness of AI algorithms for neurocomputer interfaces based on software and hardware technologies CEUR Workshop Proceedings, 2024, 3742, pp. 137–149.

21. Boyko, I. , Petryk, M. , Mudryk, I. , Stoianov, Y., Tsupryk, H. Mathematical Model of the Capacitor Based on Zeolite Material. Proceedings - International Conference on Advanced Computer Information Technologies, ACIT, 2021, pp. 45–48.

22. Tsupryk H. LLM-based Extraction from Resumes / D. Olianin, H. Tsupryk // Advanced Technologies in Scientific Research: collection of scientific papers with proceedings of the 1st International Scientific and Practical Conference, Rotterdam, Netherlands, 20–22 August 2025. – International Scientific Unity, 2025. – 72-76.

23. . R. Petryk, A. Khimich, M. M. Petryk, and J. Fraissard, “Experimental and computer simulation studies of dehydration on microporous adsorbent of natural gas used as motor fuel,” Fuel, Vol. 239, 1324–1330 (2019). https://doi.org/https://doi.org/10.1016/j.fuel.2018.10.134

24. Заікіна Д., Глива В. Основи охорони праці та безпека життєдіяльності. 2019. URL: https://doi.org/10.31435/rsglobal/001 (дата звертання: 10.12.2025).

25. Зеркалов Д.В. Безпека життєдіяльності та основи охорони праці. Навчальний посібник. К.: «Основа». 2016. – 267 с.

ДОДАТКИ

Додаток А

Частина файлу налаштувань Docker Registry

```
version: 0.1

storage:
  cache:
    blobdescriptor: inmemory
  filesystem:
    rootdirectory: /var/lib/registry
  delete:
    enabled: true

http:
  addr: :5000
  headers:
    X-Content-Type-Options: [nosniff]

notifications:
  events:
    includereferences: false
  endpoints:
    - name: ImagesService
      disabled: false
      url: http://<images_service_ip>:82/event
      timeout: 500ms
      threshold: 5
      backoff: 1s
      ignoredmediatypes:
        - application/octet-stream
      ignore:
        mediatypes:
          - application/octet-stream
        actions:
          - pull

auth:
  token:
    realm: http://<images_service_ip>:82/auth
    service: "registry.docker.com"
    issuer: "auth.docker.com"
    rootcertbundle: /mnt/local/certs/token.crt
```

Додаток Б

Створення кластеру IoT-пристроїв

```
try:
  cluster = self.docker_client.services.create(
    endpoint_spec=endpoint_spec,
    mode=docker.types.ServiceMode('replicated', replicas=0),
    image=self.registry_url + '/' + request.image)

  self.clusters_collection.update_one({'_id': cluster.attrs.get("ID")},
{'$set': {'cloudlets': nearest_cloudlets_id}}, upsert=True)

  for cloudlet_id in nearest_cloudlets_id:
    cloudlet_labels = self.docker_client.nodes.get(cloudlet_id).at-
trs.get('Spec').get('Labels')
    cloudlet_labels[cluster.attrs.get('ID')] = 'True'
    self.docker_client.nodes.get(cloudlet_id).update({'Availability':
'active', 'Labels': cloudlet_labels, 'Role': 'worker'})

  cluster.update(constraints=[f'node.labels.{cluster.at-
trs.get("ID")}==True'], mode=docker.types.ServiceMode('replicated', rep-
licas=1))

except Exception as error:
  cluster.remove()
  status = scheduling_service_pb2.Response(code=500, message='An internal
server error occurred.' + str(error))
  return scheduling_service_pb2.ResponseWithCluster(status=status)

cluster_task = cluster.tasks()[0]
state = cluster_task.get('Status').get('State')

status = scheduling_service_pb2.Response(code=201, message='Cluster of
IoT devices has been created successfully.')
return scheduling_service_pb2.ResponseWithCluster(status=status, clus-
ter=scheduling_service_pb2.Cluster(id=cluster.attrs.get("ID"),
state=state))
```