Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

Пояснювальна записка

до магістерської кваліфікаційної роботи
магістр
(освітньо—кваліфікаційний рівень)
на тему: "Комп'ютерна система для обробки та аналізу даних з використанням інтернету речей"
08–54.МКР.017.00.000 ПЗ

Виконав: магістрант групи 1КІ—23м
Спеціальності 123 — Комп`ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
Сліденко Д.О.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф.ОТ
Городецька О.С.
(прізвище та ініціали)
Опонент: доц.каф. МБІС
Салієва О.В.
(прізвище та ініціали)

Допущено до захисту
Зав. каф. ОТ
д. т. н., проф. Азаров О. Д.
" "_____2024 р.

Вінниця 2024

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо–кваліфікаційний рівень – магістр
Спеціальність 123 – «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Зав. каф. ОТ
д. т. н., проф. Азаров О. Д.
«18» 09 2024 року

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

магістранту Сліденко Дмитру Олександровичу

1 Тема роботи: "Ком'ютерна система для обробки та аналізу даних з використанням інтернету речей" керівник роботи: к.т.н., доцент кафедри ОТ Городецька О.С. затверджена наказом Вінницького національного технічного університету від від 18.09.2024 року № 310.

2 Строк подання магістрантом роботи: 19.12.2024

3 Вихідні дані до роботи: комп'ютерна система для обробки та аналізу даних з використанням інтернету речей.

4 Зміст розрахунково—пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз стану засобів реалізації комп'ютерної системи, дослідження технологій побудови та вибір засобів комп'ютерної системи, розробка комп'ютерної системи, тестування комп'ютерної системи, розрахунок економічної доцільності створення комп'ютерної системи.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): у магістерській роботі подано структурну та функціональну схеми комп'ютерної системи, які демонструють взаємодію ключових компонентів.

6 Консультанти розділів роботи наведені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1 – 4	Городецька О. С., к.т.н., доцент каф. ОТ		
5	Небава М.І., к.е.н., професор каф. ЕПВМ		

7. Дата видачі завдання 18.09.2024р.

8. Календарний план наведено в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломної магістерської роботи	Строк виконання етапів роботи	Примітка
1	Огляд і аналіз джерел інформації	04.10.2024	Виконано
2	Теоретичні дослідження технологій побудови системи та вибір засобів реалізації	18.10.2024	Виконано
3	Розробка комп'ютерної системи для обробки та аналізу даних з використанням інтернету речей	01.11.2024	Виконано
4	Розробка і тестування комп'ютерної системи	12.11.2024	Виконано
5	Економічна частина	25.11.2024	Виконано

Магістрант _____ Сліденко Д.О.
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ Городецька О.С.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сліденко Д.О. "Комп’ютерна система для обробки та аналізу даних з використанням інтернету речей". Магістерська кваліфікаційна робота зі спеціальності 123 — Комп’ютерна Інженерія, Вінниця: ВНТУ, 2024.

На укр. мові. Бібліогр.: 20 назв; рис.:36; табл.6.

У роботі розглянуто принципи побудови комп’ютерної системи для обробки та аналізу даних з використанням технологій інтернету речей (IoT). Проаналізовано основні підходи до збору даних за допомогою мережі сенсорів та їх передачі через Wi-Fi і Ethernet. Розроблено теоретичні основи інтеграції зовнішніх погодних сервісів, зокрема OpenWeatherMap, для підвищення точності аналізу. Запропоновано використання алгоритмів штучного інтелекту для обробки даних, прогнозування та генерування рекомендацій. Розроблено архітектуру системи, яка охоплює рівні пристроїв, мережі та обробки даних. Реалізовано механізми автоматичного керування пристроями та адаптації налаштувань залежно від прогнозів і даних користувача. Система забезпечує масштабованість, ефективність і можливість використання в різних галузях.

Ключові слова: інтернет речей, обробка даних, сенсори, штучний інтелект, інтеграція сервісів, прогнозування, автоматизація.

ABSTRACT

Slidenko D.O. "Computer system for data processing and analysis using the Internet of Things". Master's thesis on specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2024.

In Ukrainian. Bibliography: 20 titles; Figures: 36; Table 6.

The paper examines the principles of designing a computer system for data processing and analysis using Internet of Things (IoT) technologies. The main approaches to data collection using sensor networks and their transmission via Wi-Fi and Ethernet are analyzed. Theoretical foundations of integrating external weather services, particularly OpenWeatherMap, are developed to enhance analysis accuracy. The use of artificial intelligence algorithms for data processing, forecasting, and generating recommendations is proposed. A structural architecture of the system is developed, encompassing device, network, and data processing levels. Mechanisms for automatic device control and adjustment of settings based on forecasts and user-provided data are implemented. The system ensures scalability, efficiency, and applicability in various fields.

Keywords: Internet of Things, data processing, sensors, artificial intelligence, service integration, forecasting, automation.

ЗМІСТ

ВСТУП

Розвиток сучасних технологій, зокрема Інтернету речей (IoT, Internet of Things), відкриває нові можливості для ефективної обробки та аналізу даних. Завдяки підключенню пристроїв до єдиної мережі, їх взаємодії та обміну інформацією в реальному часі, виникають нові шляхи вирішення складних задач управління, моніторингу та прийняття рішень. Ці системи забезпечують автоматизацію процесів у промисловості, охороні здоров’я, транспорті, сільському господарстві та інших сферах.

Системи обробки та аналізу даних на основі IoT інтегрують сенсори, обчислювальні потужності та алгоритми штучного інтелекту. Це дозволяє ефективно обробляти великі обсяги інформації, що надходить від різноманітних пристроїв, і приймати оптимальні рішення на основі отриманих даних.

Метою роботи є розширення функціональних можливостей комп'ютерної системи завдяки інтеграції з штучним інтелектом та зовнішніми погодними сервісами, що дасть можливість підвищити точність керування системами опалення та кондиціонування.

Задачі дослідження:

— провести аналітичний огляд систем для обробки та аналізу даних з використанням інтернету речей;

— розробити архітектуру комп’ютерної системи;

— вдосконалити метод обробки даних комп’ютерної системи шляхом інтеграції штучного інтелекту та зовнішніх погодних сервісів;

— виконати програмну реалізацію клієнтської та серверної частини комп’ютерної системи;

— провести тестування комп’ютерної системи.

Об’єкт дослідження — процес проєктування комп'ютерної системи для обробки та аналізу даних з використанням інтернету речей

Предмет дослідження — методи та засоби обробки даних з використанням Інтернету речей.

Наукова новизна одержаних результатів полягає у вдосконаленні методу обробки даних в комп'ютерній системі, в якому на відміну від існуючих, реалізовано інтеграцію штучного інтелекту та зовнішніх погодних сервісів, що дозволило розширити функціональні можливості системи.

Практичне значення одержаних результатів: полягає в отриманні інтегрованої системи, яка поєднує новітні технології для обробки даних з IoT, а також інтеграцію погодних сервісів і геолокації будинку.

Апробація результатів роботи — зроблено доповіді на LI науково-технічній конференції підрозділів ВНТУ та на науково-технічній конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)».

Публікації:

Сліденко Д.О. Система для обробки та аналізу даних з використанням інтернету речей / Д.О. Сліденко, О. С. Городецька, О. В. Войцеховська // Тези доповіді. LI науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії. Вінниця 2023 р. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/view/17561/15656 [1].

Комп’ютерна система обробки та аналізу даних з використанням інтернету речей / Д. О. Сліденко, О. С. Городецька, Д. В. Куклій // Матеріали Молодь в науці: дослідження, проблеми, перспективи (МН-2025), 2025 р. [Електронний ресурс] — https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/22878/18887 [2].

1 АНАЛІТИЧНИЙ ОГЛЯД СИСТЕМ ДЛЯ ОБРОБКИ ТА АНАЛІЗУ ДАНИХ З ВИКОРИСТАННЯМ ІНТЕРНЕТУ РЕЧЕЙ

1.1 Загальні відомості про інтернет речей

Інтернет речей (IoT) є концепцією, що охоплює зв'язок між фізичними об'єктами та мережею Інтернет, дозволяючи цим об'єктам збирати, обмінюватися та аналізувати дані. В основі IoT лежить ідея, що будь-який предмет, обладнаний сенсорами, програмним забезпеченням та мережевими функціями, може стати "розумним" і взаємодіяти з іншими пристроями та системами. Це створює нові можливості для автоматизації, моніторингу та контролю в різних сферах життя.

Основними компонентами IoT є сенсори, які збирають дані про навколишнє середовище, обробка цих даних, що може відбуватися на самих пристроях або в хмарі, а також мережа, яка забезпечує з'єднання між різними елементами системи (рисунок 1.1). Зокрема, сенсори можуть вимірювати такі параметри, як температура, вологість, тиск, місцезнаходження та багато інших, що робить їх незамінними у промислових, медичних та побутових застосуваннях.

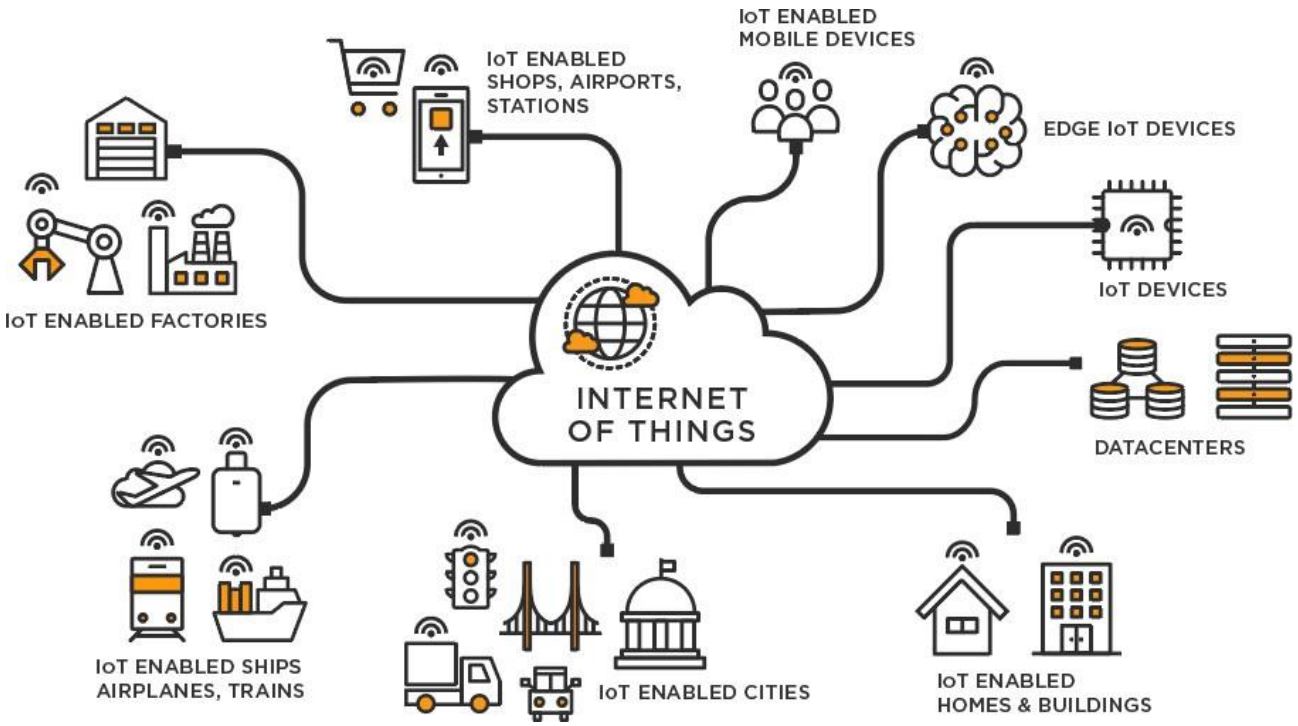

Рисунок 1.1 — Інтернет речей

IoT застосовується в багатьох галузях. У промисловості, наприклад, розумні сенсори можуть відстежувати стан машин, прогнозувати збої та оптимізувати виробничі процеси. У сільському господарстві IoT технології допомагають в автоматизації поливу, моніторингу врожайності та управлінні ресурсами. В охороні здоров'я IoT пристрої можуть моніторити стан пацієнтів у реальному часі, що покращує якість медичних послуг.

Безпека та конфіденційність даних є ще одним критично важливим питанням, пов'язаним з розвитком IoT. Оскільки пристрої постійно взаємодіють між собою, питання захисту даних від несанкціонованого доступу, а також гарантії конфіденційності інформації стають надзвичайно актуальними. Розробка протоколів безпеки, шифрування даних та систем аутентифікації є необхідними для забезпечення надійності IoT-систем. Нижче представлено таблицю 1.1, в якій наведено основні компоненти та застосування інтернету речей.

Таблиця 1.1. — Основні компоненти та застосування IoT

Компонент	Функція	Застосування
Сенсори	Збір даних про навколишнє середовище	Вимірювання температури, вологості, тиску, місця розташування
Обробка даних	Аналіз зібраних даних	Виявлення закономірностей, прийняття рішень
Мережа	З’єдання між пристроями	Передача даних між сенсорами, хмарою та іншими системами
Промисловість	Оптимізація виробництва	Моніторинг стану обладнання, прогнозування збоїв
Сільське господарство	Автоматизація процесів	Управління поливом, моніторинг врожайності
Охорона здоров’я	Моніторинг пацієнтів	Відстеження здоров'я, раннє виявлення захворювань

Загалом, Інтернет речей трансформує наші уявлення про технології та їх використання, відкриваючи нові горизонти для інновацій. Він забезпечує можливість створення інтегрованих рішень, які можуть підвищити ефективність у різних сферах діяльності. Разом із цим IoT продовжує ставити нові виклики, які потребують комплексного підходу до їх вирішення, що робить цю область дослідження надзвичайно динамічною та важливою для майбутнього.

1.2 Методи збору та обробки даних в IoT

Збір та обробка даних у контексті Інтернету речей (IoT) є ключовими процесами, які забезпечують функціонування розумних пристроїв і систем. Ці методи включають в себе різні технології та підходи, які допомагають ефективно отримувати, обробляти та аналізувати інформацію, що генерується IoT-пристроями.

Процес збору даних починається із застосування сенсорів, які встановлюються на різноманітних об'єктах, таких як автомобілі, будівлі, сільськогосподарські поля та інші середовища. Сенсори можуть мати різноманітні функції: вимірювати температуру, вологість, тиск, рух, звукові коливання та багато інших параметрів [3]. Вони забезпечують постійний моніторинг умов і здатні надсилати дані в реальному часі. Для збору даних також можуть використовуватися пристрої, які виконують специфічні задачі, наприклад, камери відеоспостереження, які фіксують зображення та відео для подальшого аналізу.

Зібрані дані зазвичай передаються через бездротові мережі, такі як Wi-Fi, Bluetooth, Zigbee або мобільні мережі. Вибір методу передачі залежить від вимог до швидкості, енергоспоживання, дальності та надійності зв'язку. Наприклад, для IoT-пристроїв, які працюють в умовах обмеженої пропускної здатності або в складних середовищах, можуть використовуватися технології, які забезпечують економію енергії, такі як LoRaWAN [4].

Після збору даних важливо їх обробити. Обробка може відбуватися на самих пристроях, що називається обробкою на краю (edge computing), або в хмарі. У випадку обробки на краю дані аналізуються безпосередньо на пристроях, що

дозволяє зменшити затримки у передачі даних та знизити навантаження на мережу. Це особливо корисно для додатків, де швидкість реагування є критично важливою, наприклад, у системах безпеки або медичних моніторах.

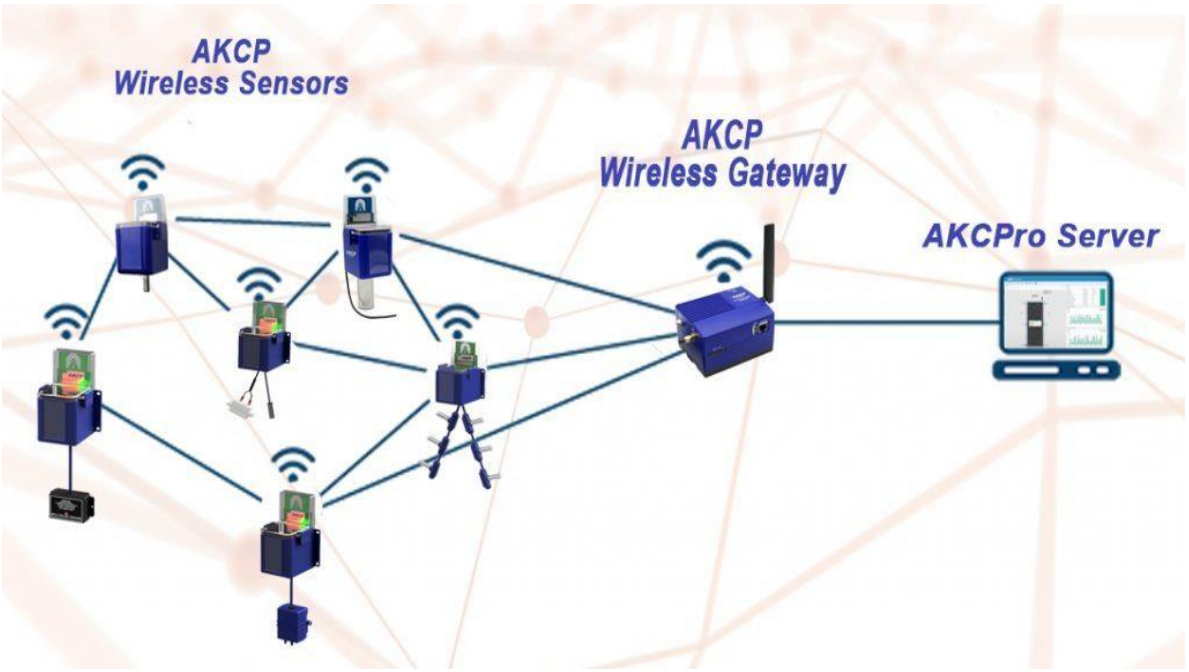

Рисунок 1.2 — Модель бездротової сенсорної мережі AKSP

Хмарна обробка, у свою чергу, забезпечує можливість обробки великих обсягів даних, що надходять з численних пристроїв. Використання потужних обчислювальних ресурсів у хмарі дозволяє реалізувати складні алгоритми аналізу, включаючи машинне навчання та штучний інтелект. Хмарні платформи також забезпечують зручний доступ до даних, що дозволяє користувачам здійснювати аналіз в реальному часі та отримувати звіти про результати.

Обробка даних включає в себе також їх очищення та нормалізацію. Це важливий етап, оскільки дані, отримані з різних джерел, можуть мати різну структуру та формат. Процес очищення допомагає усунути шуми та аномалії, які можуть спотворити результати аналізу. Нормалізація, у свою чергу, дозволяє привести дані до єдиного формату, що спрощує подальший аналіз.

Аналіз даних — ще один важливий етап, який допомагає виявити закономірності, тренди та аномалії. Методи машинного навчання, такі як класифікація, регресія та кластеризація, часто використовуються для створення

моделей, які можуть передбачати поведінку системи на основі зібраних даних. Наприклад, у промисловості такі моделі можуть допомогти передбачити збої в обладнанні та вжити заходів для їх запобігання.

Загалом, методи збору та обробки даних в IoT забезпечують ефективність та адаптивність систем, створюючи можливості для автоматизації та оптимізації процесів у різних сферах. Завдяки інтеграції новітніх технологій та алгоритмів, ці методи відкривають нові горизонти для розвитку розумних пристроїв і систем, надаючи значні переваги в управлінні ресурсами, підвищенні продуктивності та покращенні якості обслуговування.

1.3 Аналіз даних в IoT: Технології та Інструменти

Аналіз даних в контексті Інтернету речей (IoT) є критично важливим етапом, який дозволяє перетворити величезні обсяги зібраної інформації в цінні інсайти, що можуть підвищити ефективність бізнес-процесів і оптимізувати управління ресурсами. Цей процес охоплює різноманітні технології та інструменти, які забезпечують ефективну обробку, зберігання та аналіз даних, отриманих з IoT-пристроїв.

Однією з основних технологій, що використовуються для аналізу даних в IoT, є обробка великих даних (Big Data). IoT генерує величезні обсяги даних, які потребують спеціалізованих рішень для їх зберігання та обробки. Технології Big Data, такі як Apache Hadoop і Apache Spark, забезпечують можливості для паралельної обробки даних, що дозволяє швидко аналізувати та інтерпретувати інформацію. Hadoop, наприклад, використовує концепцію розподіленого зберігання та обробки даних, що дозволяє зберігати дані в дешевих системах, які можуть масштабуватися за потребою [5].

Крім того, популярними в IoT є платформи для аналізу даних у реальному часі, такі як Apache Kafka та Apache Flink. Ці інструменти дозволяють обробляти потоки даних у реальному часі, що є надзвичайно важливим для застосувань, де швидкість реакції має критичне значення. Наприклад, у системах безпеки або в

управлінні трафіком своєчасна обробка даних дозволяє миттєво виявляти аномалії та вживати необхідних заходів [6].

Аналіз даних також передбачає використання алгоритмів машинного навчання і штучного інтелекту, які допомагають виявляти патерни та тренди в даних. Це може включати в себе різноманітні методи, такі як класифікація, регресія, кластеризація та асоціативний аналіз. Наприклад, в промислових додатках машинне навчання може використовуватися для прогнозування можливих збоїв у обладнанні на основі історичних даних, що дозволяє вживати превентивні заходи і зменшувати витрати на обслуговування.

Для інтерактивного аналізу та візуалізації даних використовуються різноманітні інструменти, такі як Tableau, Power BI і Grafana. Ці платформи дозволяють створювати візуалізації даних у вигляді графіків, діаграм і панелей моніторингу, що спрощує процес сприйняття інформації. Візуалізація є важливою складовою аналізу, оскільки вона допомагає користувачам краще розуміти дані і приймати обґрунтовані рішення.

Ще однією важливою складовою аналізу даних в IoT є побудова аналітичних моделей, які здатні працювати з даними в режимі реального часу. Це може включати в себе не лише традиційні моделі, але й адаптивні, які постійно вдосконалюються на основі нових даних. Такі моделі можуть використовуватися для різних цілей: від прогнозування поведінки споживачів до оптимізації виробничих процесів.

Безпека даних є ще одним важливим аспектом аналізу в IoT. Оскільки дані можуть містити чутливу інформацію, необхідно впроваджувати заходи для їх захисту. Шифрування даних, аутентифікація користувачів та контроль доступу є критично важливими для забезпечення конфіденційності та цілісності інформації.

Аналіз даних в IoT є складним і багатогранним процесом, що охоплює широкий спектр технологій та інструментів. Ці технології дозволяють не лише обробляти та зберігати великі обсяги даних, але й перетворювати їх на корисну інформацію, яка може суттєво вплинути на прийняття рішень. Завдяки інтеграції інструментів для обробки в реальному часі, машинного навчання та візуалізації,

IoT продовжує відкривати нові горизонти для інновацій та оптимізації в різних сферах, від промисловості до медичних технологій.

1.4 Сучасні тренди у системах для обробки даних з IoT

Системи для обробки даних з Інтернету речей (IoT) перебувають у постійному розвитку, і новітні тренди визначають їхню еволюцію, відповідаючи на зростаючі вимоги до швидкості, масштабованості та безпеки. Одним з основних трендів є зростання популярності обробки даних на краю (edge computing). Цей підхід передбачає обробку даних безпосередньо на пристроях або поблизу їхнього місця збору, що дозволяє зменшити затримки, пов'язані з передачею даних до хмарних серверів. Це особливо важливо для застосувань, де час реакції є критично важливим, наприклад, у промислових системах автоматизації або медичних пристроях, які потребують термінового аналізу інформації.

Ще одним важливим трендом є інтеграція штучного інтелекту (ШІ) та машинного навчання (МН) у процеси обробки даних. Використання алгоритмів ШІ дозволяє автоматизувати аналіз даних, виявляти патерни та робити прогнози на основі зібраної інформації. Це відкриває нові можливості для оптимізації бізнес-процесів, зокрема у сферах, таких як прогнозування попиту, управління ризиками і обслуговування клієнтів. Крім того, інтеграція ШІ дозволяє реалізувати адаптивні системи, які здатні самостійно навчатися на основі нових даних, що підвищує їхню ефективність і точність [7].

Технології блокчейн також стають важливими в контексті IoT, особливо з точки зору безпеки і прозорості. Використання дистрибуційного реєстру даних може забезпечити надійний запис усіх транзакцій і обміну інформацією між пристроями, знижуючи ризик несанкціонованого доступу або маніпуляцій з даними. Блокчейн може забезпечити аутентифікацію пристроїв і даних, що робить його перспективним рішенням для чутливих застосувань, таких як фінансові операції, медичні дані або управління ланцюгами постачання.

Необхідність обробки великих обсягів даних також веде до зростання популярності хмарних рішень. Хмарні платформи дозволяють зберігати і

обробляти величезні обсяги даних, забезпечуючи масштабованість та доступність ресурсів. Системи, що функціонують у хмарі, дозволяють підприємствам швидко адаптуватися до змінюваних умов, пропонуючи гнучкість у виборі ресурсів і потужностей, які необхідні для їхніх завдань.

Зростає також важливість аналітики в реальному часі, оскільки компанії прагнуть отримувати оперативні дані для швидкого прийняття рішень. Системи, які можуть аналізувати дані у реальному часі, надають можливість швидко реагувати на зміни в середовищі та адаптувати свої стратегії. Це особливо важливо в таких сферах, як розумні міста, де дані про трафік, енергоспоживання і безпеку можуть використовуватися для покращення якості життя мешканців.

Крім того, в IoT спостерігається зростаюча увага до екологічної стійкості та енергоефективності. Виробники пристроїв та розробники програмного забезпечення впроваджують рішення, які знижують енергоспоживання IoT-пристроїв, що, у свою чергу, зменшує вуглецевий слід. Це включає в себе оптимізацію алгоритмів обробки даних та використання нових, енергоефективних технологій зв’язку.

Сучасні тренди у системах для обробки даних з IoT вказують на швидкий розвиток технологій, які відповідають потребам бізнесу та суспільства. Інтеграція новітніх технологій, таких як обробка на краю, штучний інтелект, блокчейн і хмарні рішення, створює нові можливості для ефективного управління даними, підвищення безпеки та забезпечення швидкого реагування на зміни. Ці зміни відкривають нові горизонти для інновацій у різних сферах, від промисловості до міського управління, формуючи майбутнє IoT.

1.5. Приклади застосування IoT у різних галузях

Інтернет речей (IoT) має величезний потенціал для трансформації різних галузей, оскільки дозволяє з'єднувати фізичні об'єкти з мережею Інтернет, забезпечуючи збір і обробку даних у реальному часі. Це відкриває нові можливості для автоматизації, моніторингу та оптимізації процесів.

У сфері промисловості IoT застосовується для створення розумних фабрик, де сенсори моніторять роботу обладнання, відстежуючи його стан і виявляючи можливі збої. Це дозволяє компаніям проводити прогнозне обслуговування, що зменшує час простою та витрати на ремонти. Наприклад, компанії можуть використовувати дані з сенсорів для аналізу вібрацій, температури і споживання енергії, щоб вчасно виявити проблеми і запобігти серйозним збоям у виробничих процесах.

У сільському господарстві IoT забезпечує можливість точного землеробства. Фермери використовують сенсори для моніторингу вологості ґрунту, температури повітря та інших параметрів, що впливають на вирощування рослин. Зібрані дані допомагають оптимізувати використання води та добрив, що веде до підвищення врожайності та зменшення витрат. Наприклад, системи автоматичного поливу можуть активуватися лише тоді, коли вологість ґрунту знижується до певного рівня, що дозволяє економити ресурси.

У сфері охорони здоров'я IoT використовується для моніторингу стану пацієнтів у реальному часі. Різноманітні носимі пристрої, такі як фітнес-трекери та медичні монітори, можуть зібрати дані про серцевий ритм, артеріальний тиск і рівень цукру в крові. Ця інформація може бути передана лікарям, які зможуть вчасно реагувати на зміни стану пацієнта, що особливо важливо для хронічних захворювань. Наприклад, пацієнти з діабетом можуть отримувати сповіщення про небезпечні зміни в рівні цукру, що дозволяє їм швидше вжити необхідних заходів.

У транспортній галузі IoT активно використовується для управління логістикою та моніторингу транспортних засобів. Сенсори в автомобілях можуть відстежувати їх місцезнаходження, стан двигуна і паливну ефективність. Це дозволяє компаніям оптимізувати маршрути, зменшити витрати на паливо і покращити обслуговування клієнтів. Наприклад, системи управління флотом можуть використовувати дані для визначення найбільш ефективних маршрутів, що зменшує витрати та час доставки.

У сфері енергетики IoT використовується для моніторингу і управління енергетичними системами. Розумні лічильники можуть збирати дані про

споживання енергії в реальному часі, що дозволяє споживачам краще управляти своїм споживанням і зменшувати витрати. Крім того, ця інформація може допомогти постачальникам енергії прогнозувати навантаження на мережу і оптимізувати виробництво енергії. Наприклад, дані з розумних лічильників можуть бути використані для виявлення аномалій у споживанні, що може свідчити про втрати або несанкціоноване споживання.

У сфері розумних міст IoT дозволяє інтегрувати технології для покращення якості життя мешканців. Сенсори можуть моніторити якість повітря, рівень шуму і трафік, що дозволяє місцевим органам влади швидко реагувати на проблеми. Наприклад, системи управління трафіком можуть використовувати дані з камер і сенсорів, щоб оптимізувати світлофорні цикли і зменшити затори[8].

Таким чином, застосування IoT охоплює широкий спектр галузей, і його впровадження відкриває нові можливості для автоматизації, оптимізації і підвищення ефективності процесів. Завдяки інтеграції сенсорних технологій і аналітики даних, IoT продовжує трансформувати способи, якими ми працюємо, живемо і взаємодіємо зі світом навколо нас.

2 ТЕОРЕТИЧНІ ЗАСАДИ ВДОСКОНАЛЕНОГО МЕТОДУ ОБРОБКИ ДАНИХ КОМП'ЮТЕРНОЇ СИСТЕМИ

2.1 Проектування архітектури комп'ютерної системи обробки та аналізу даних з використанням IoT

Інтернет речей (IoT) змінює підходи до проектування комп'ютерних систем обробки даних, створюючи умови для ефективного збору, передачі та аналізу великої кількості інформації в режимі реального часу. IoT дозволяє об'єднати фізичні пристрої, сенсори та мережі у єдину екосистему, що відкриває нові можливості для автоматизації, моніторингу та прийняття рішень. Проектування архітектури такої системи вимагає врахування як технічних аспектів, так і логіки взаємодії її компонентів.

Основою будь-якої IoT-системи є пристрої, що збирають дані з реального світу. Це можуть бути сенсори температури, вологості, рівня освітлення, руху, GPS-модулі тощо. Дані, зібрані цими пристроями, мають бути передані для обробки, що вимагає розробки ефективної архітектури для роботи з мережею. Важливими аспектами на цьому етапі є забезпечення надійності передачі даних, їхньої безперервності та захищеності.

Передача даних у IoT може здійснюватися за допомогою різних протоколів, таких як MQTT, HTTP, CoAP або AMQP. Вибір протоколу залежить від вимог до продуктивності, енергозбереження та масштабованості системи. Наприклад, MQTT (рисунок 2.1) є одним із найпопулярніших варіантів завдяки його легковажності та можливості роботи з великою кількістю пристроїв[9].

Дані, зібрані IoT-пристроями, зазвичай надходять до хмарного середовища для подальшої обробки та аналізу. Хмара забезпечує масштабованість, обчислювальну потужність та інструменти для аналізу великих обсягів інформації. Важливо також враховувати можливість використання edge-обчислень, які дозволяють виконувати попередню обробку даних безпосередньо на пристроях або поблизу джерела даних. Це зменшує затримки та знижує навантаження на центральні сервери.

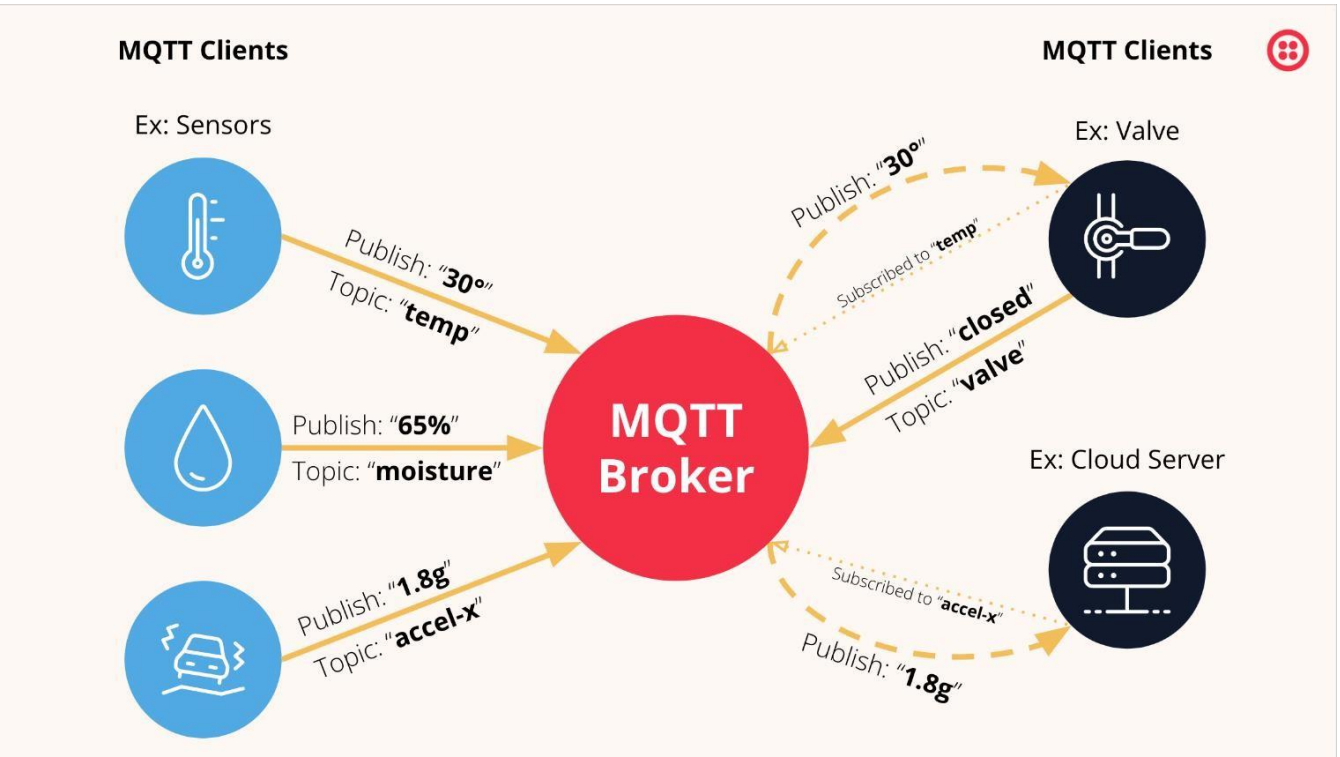

Рисунок 2.1 — Приклад роботи протоколу MQTT

Проектуючи архітектуру, потрібно враховувати обробку потоків даних у реальному часі. Це стосується як фільтрації та нормалізації даних, так і аналізу із застосуванням алгоритмів машинного навчання. Наприклад, моделі прогнозування можуть використовуватися для виявлення потенційних проблем у системах, таких як прогнозування відмов обладнання чи аналіз поведінки користувачів [10].

Одним із ключових компонентів архітектури є бази даних. У випадку IoT часто використовуються нереляційні бази даних (NoSQL), такі як MongoDB чи Cassandra, які дозволяють ефективно працювати з неструктурованими даними. Для аналізу часових рядів особливо корисними є спеціалізовані бази даних, наприклад, InfluxDB.

Система обробки даних має враховувати питання безпеки. IoT пристрої часто є вразливими до атак через обмежені ресурси для реалізації складних алгоритмів шифрування. Забезпечення захищеного з'єднання через протоколи TLS/SSL, автентифікація пристроїв та регулярне оновлення програмного забезпечення є критично важливими аспектами.

Моніторинг і управління системою IoT є важливими етапами її функціонування. Важливо забезпечити засоби для контролю стану пристроїв, оперативного виявлення збоїв і аномалій, а також можливість оновлення прошивок "по повітрю" (over-the-air, OTA). Для цього можуть використовуватися централізовані панелі управління та автоматизовані системи сповіщення.

Інтеграція IoT у бізнес-процеси вимагає створення інтерфейсів для взаємодії із зовнішніми системами. Це можуть бути API для доступу до оброблених даних або інтеграція з ERP чи CRM системами для автоматизації прийняття рішень. Наприклад, у логістиці дані з IoT-сенсорів можуть автоматично оновлювати маршрути доставки в реальному часі.

Архітектура системи IoT повинна бути модульною та масштабованою. Це дозволяє адаптувати її до змін у кількості пристроїв або обсягах даних. Мікросервісна архітектура є оптимальним рішенням, оскільки кожен сервіс може бути розроблений, розгорнутий та масштабований незалежно.

Для зменшення часу простоїв та покращення продуктивності варто використовувати технології балансування навантаження та реплікації. Це особливо важливо для систем, що працюють у реальному часі, оскільки затримки можуть призвести до критичних помилок у прийнятті рішень.

IoT-системи також відкривають можливості для аналізу великих даних (Big Data). Завдяки даним, що постійно надходять від пристроїв, можна отримати цінну інформацію для побудови прогнозів, оптимізації процесів та виявлення прихованих закономірностей. Використання платформ, таких як Apache Kafka (рисунок 2.2) чи Apache Spark (рисунок 2.3), дозволяє ефективно працювати з великими потоками інформації.

Окремо слід зупинитися на енергоспоживанні пристроїв IoT. Багато з них працюють на батарейках, тому проектування енергоефективних рішень є важливим аспектом. Використання протоколів із низьким споживанням енергії, таких як Zigbee або LoRaWAN, дозволяє значно продовжити термін роботи пристроїв. Розгортання IoT системи також вимагає аналізу потреб у зберіганні даних[11].

Дані, які не потребують оперативної обробки, можуть архівуватися у дешевших системах зберігання, таких як Amazon Glacier, що знижує витрати на інфраструктуру.

Рисунок 2.2 — Платформа Apache Kafka

Проектування архітектури IoT системи обробки та аналізу даних є складним і багатогранним завданням, яке вимагає врахування багатьох аспектів: від вибору обладнання до програмного забезпечення та протоколів взаємодії. Грамотна інтеграція цих компонентів дозволяє створити ефективну, масштабовану та безпечну систему, здатну вирішувати найрізноманітніші завдання.

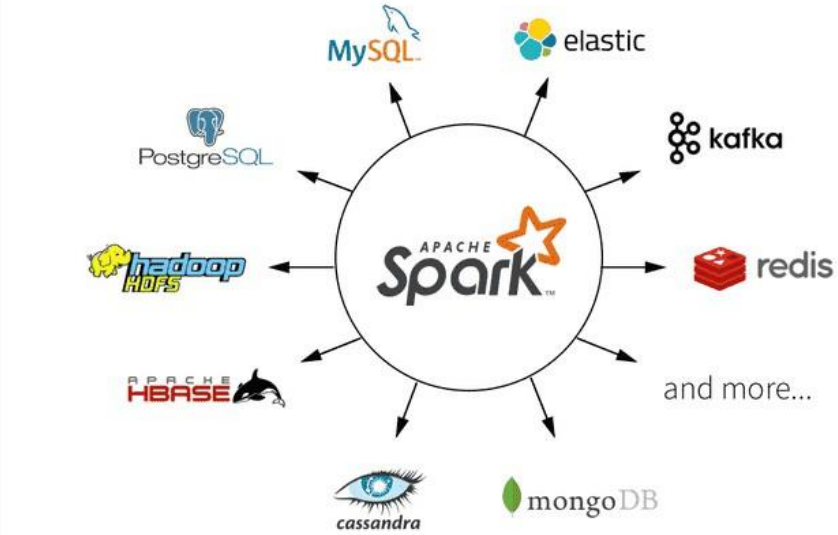

Рисунок 2.3 — Платформа Apache Kafka

Схема наведена на рисунку 2.4 ілюструє архітектуру комп’ютерної системи для обробки та аналізу даних, побудовану на основі технологій Інтернету речей (IoT). Вона складається з трьох основних рівнів: рівень пристроїв, рівень мережі та рівень обробки даних. Кожен рівень виконує певні функції, які спрямовані на забезпечення ефективного збору, передачі та обробки інформації для надання користувачеві прогнозів та рекомендацій.

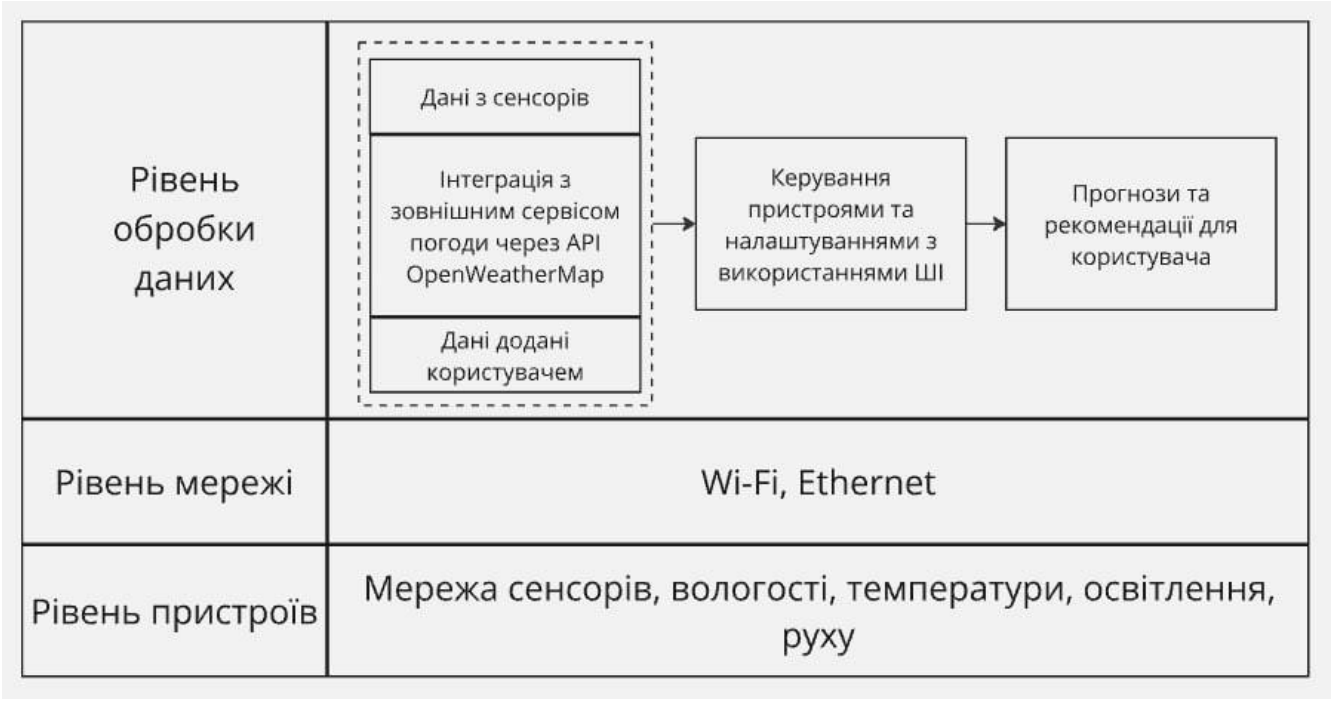

Рисунок 2.4 — Ілюстрація рівнів структури реалізованої системи

Перший рівень схеми — рівень пристроїв. На цьому рівні розташована мережа сенсорів, які збирають первинні дані з навколишнього середовища. Сенсори вимірюють такі параметри, як вологість, температура, освітлення, рух та інші показники, залежно від цілей системи. Наприклад, сенсори руху можуть використовуватися для охорони приміщень, а сенсори вологості — для моніторингу стану ґрунту в сільському господарстві.

Другий рівень — рівень мережі. Він забезпечує передачу даних від сенсорів до наступного рівня системи за допомогою мережевих технологій, таких як Wi-Fi або Ethernet. Цей рівень є критичним, оскільки якість і швидкість передачі даних значною мірою визначають загальну продуктивність системи. Завдяки

використанню сучасних стандартів передачі даних система може забезпечувати збирання інформації в реальному часі.

Третій рівень — рівень обробки даних. Тут зібрані дані проходять кілька етапів обробки. Спершу вони надходять із сенсорів, які безпосередньо взаємодіють із середовищем. Окрім цього, на цьому рівні відбувається інтеграція із зовнішнім погодним сервісом через API, зокрема OpenWeatherMap (рисунок 3.2). Цей елемент дозволяє доповнити дані з сенсорів інформацією про погодні умови.

2.2 Вдосконалення методу обробки даних шляхом інтеграції штучного інтелекту, регресійних моделей та зовнішніх погодних сервісів

Інтернет речей (IoT) продовжує трансформувати різні галузі, надаючи можливість автоматизувати процеси та створювати ефективніші системи за рахунок збору й аналізу великої кількості даних у режимі реального часу. Вдосконалення IoT шляхом інтеграції технологій штучного інтелекту (ШІ), регресійних моделей та зовнішніх погодних сервісів дозволяє значно підвищити продуктивність, точність прогнозів і адаптивність систем до змін зовнішнього середовища.

Інтеграція штучного інтелекту в IoT дає змогу проводити аналіз великих обсягів даних і знаходити приховані закономірності, які не можуть бути виявлені за допомогою традиційних методів. ШІ дозволяє не лише збирати дані, але й приймати оптимальні рішення на основі їхнього аналізу. Наприклад, в аграрному секторі IoT-системи, оснащені ШІ, можуть прогнозувати стан врожаю залежно від погодних умов, рівня вологості ґрунту та інших факторів.

Регресійні моделі (рисунок 2.4) є основою для багатьох аналітичних інструментів, інтегрованих у IoT. Вони забезпечують можливість прогнозування майбутніх значень на основі історичних даних. Наприклад, у системах моніторингу енергоспоживання регресійні моделі допомагають передбачати пікові навантаження, що дозволяє заздалегідь оптимізувати роботу енергетичних систем. Включення таких моделей до IoT-архітектури покращує точність і адаптивність аналітики [12].

Зовнішні погодні сервіси, інтегровані з IoT, надають додатковий шар інформації для систем, які мають адаптуватися до змін зовнішнього середовища. У сільському господарстві ці сервіси дозволяють коригувати полив, планувати внесення добрив чи прогнозувати ризик заморозків. З'єднання IoT з такими сервісами значно підвищує точність і ефективність прийнятих рішень.

Одним із ключових аспектів інтеграції ШІ, регресійних моделей та погодних даних є їхня спільна робота в реальному часі. IoT-пристрої, такі як сенсори, постійно збирають дані, які аналізуються ШІ з урахуванням зовнішніх змін, що надходять із погодних сервісів. Ця інформація дозволяє створювати динамічні моделі, які адаптуються до змін навколишнього середовища та прогнозують майбутні сценарії.

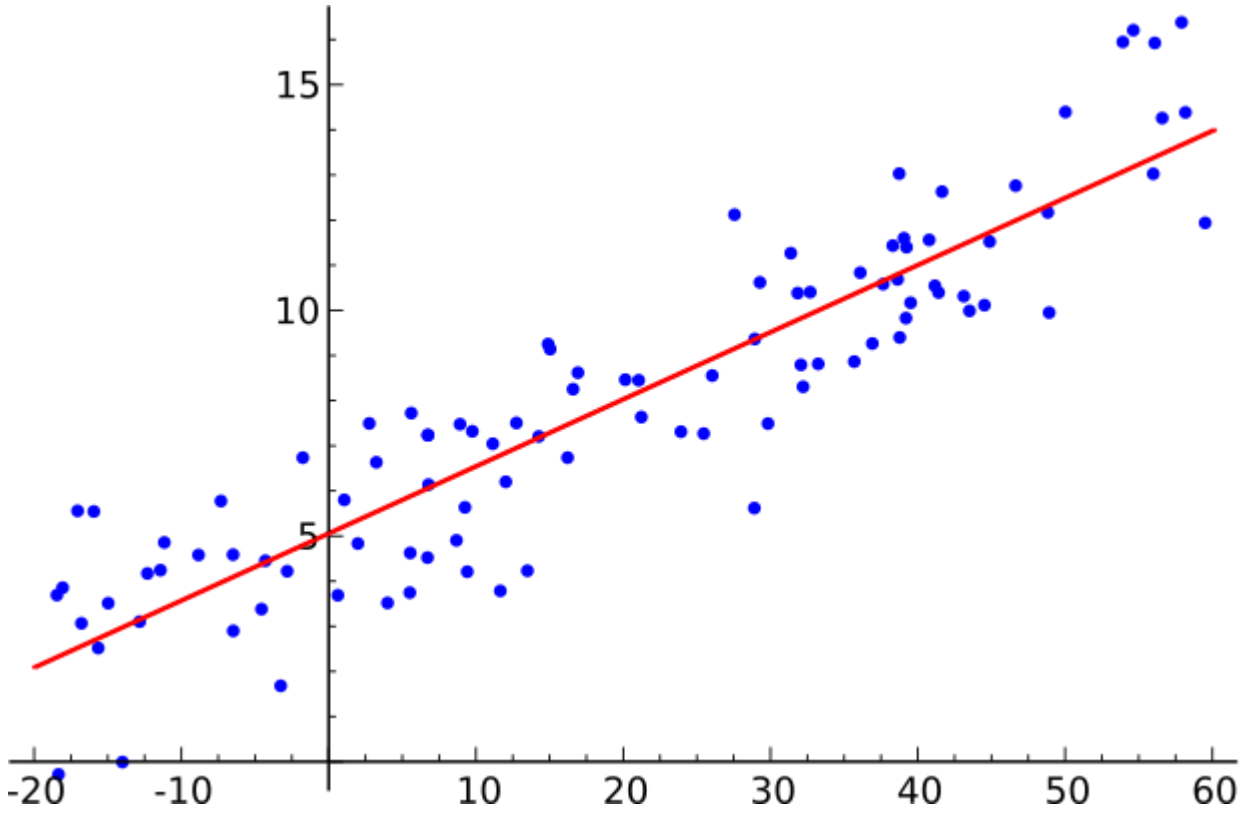

Рисунок 2.5 — Приклад лінійної регресійної моделі

Наприклад, у логістиці інтеграція IoT із ШІ та погодними сервісами може оптимізувати маршрути доставки в умовах складної погоди. За допомогою регресійних моделей система аналізує минулі дані про вплив погоди на швидкість

доставки, а ШІ генерує альтернативні маршрути. У результаті зменшується час доставки та підвищується задоволеність клієнтів [13].

Важливим завданням у рамках вдосконалення IoT-систем є інтеграція передових технологій із метою підвищення точності аналізу. Регресійні моделі на основі машинного навчання дозволяють створювати адаптивні алгоритми, які враховують специфіку різних галузей. Наприклад, у системах прогнозування споживання ресурсів такі моделі враховують як історичні дані, так і поточні умови.

Штучний інтелект у поєднанні з погодними сервісами особливо ефективний у сфері "розумного" управління містами [14]. Наприклад, IoT-системи можуть регулювати роботу систем освітлення або опалення на основі прогнозів температури чи вологості, що забезпечує економію ресурсів. ШІ аналізує минулі дії системи та рекомендує оптимальні сценарії управління.

Одним із викликів інтеграції є забезпечення точності даних, що надходять із зовнішніх сервісів. IoT-системи повинні бути здатними перевіряти отриману інформацію на достовірність, щоб уникнути помилок у прогнозах. Це вимагає створення систем контролю якості даних і застосування ШІ для їх перевірки.

Ефективна інтеграція IoT із ШІ та погодними сервісами також потребує врахування питань безпеки. Захист даних, що надходять із зовнішніх джерел, є критичним завданням. Використання протоколів шифрування, автентифікації пристроїв та систем моніторингу мережі дозволяє забезпечити належний рівень захищеності.

Розвиток технологій дозволяє створювати ще точніші регресійні моделі, які працюють з великими масивами даних. Наприклад, у транспортній сфері ці моделі можуть прогнозувати затори, враховуючи як погодні умови, так і динаміку руху транспорту (рисунок 2.6). Це дозволяє IoT-системам пропонувати оптимальні рішення в режимі реального часу.

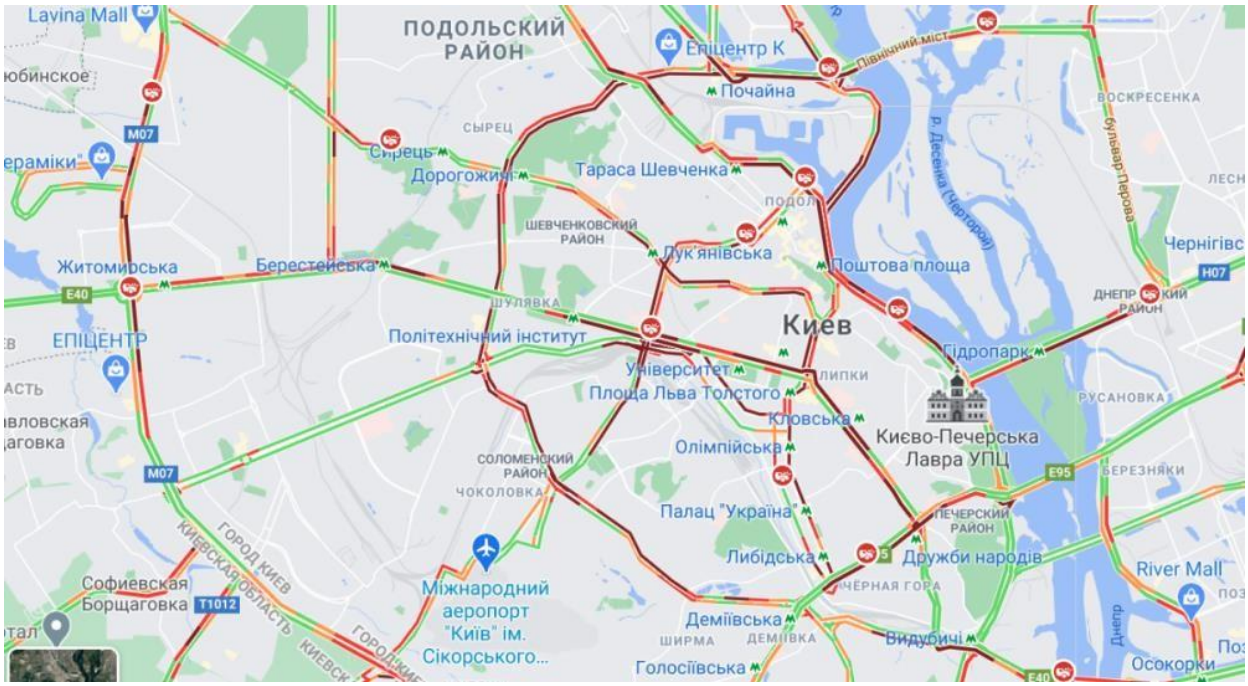

Рисунок 2.6 — Приклад аналізу заторів в місті Київ

Впровадження таких рішень є особливо важливим у промисловості, де точність прогнозів і адаптивність мають критичне значення. Системи, які об’єднують IoT, ШІ, регресійні моделі та погодні сервіси, дозволяють значно зменшити втрати, підвищити ефективність і покращити безпеку на виробництві.

Переваги інтеграції IoT із сучасними технологіями виходять за межі окремих галузей. У поєднанні вони створюють умови для впровадження стійких екосистем, які враховують екологічні, економічні та соціальні аспекти. Наприклад, у системах "розумного" сільського господарства можна одночасно зменшити витрати ресурсів та підвищити врожайність (рисунок 2.7).

Ще однією перспективною сферою є управління інфраструктурою, де ШІ та регресійні моделі допомагають виявляти потенційні проблеми ще до їх виникнення. Наприклад, у водопостачанні IoT-системи можуть прогнозувати витоки, аналізуючи зміни тиску в трубопроводах залежно від погодних умов і часу доби.

Варто зазначити, що інтеграція ШІ, регресійних моделей та погодних сервісів із IoT створює не лише нові можливості, а й виклики. Високі вимоги до

обчислювальних ресурсів і зберігання даних вимагають оптимізації системи на всіх рівнях – від пристроїв до хмарних обчислень.

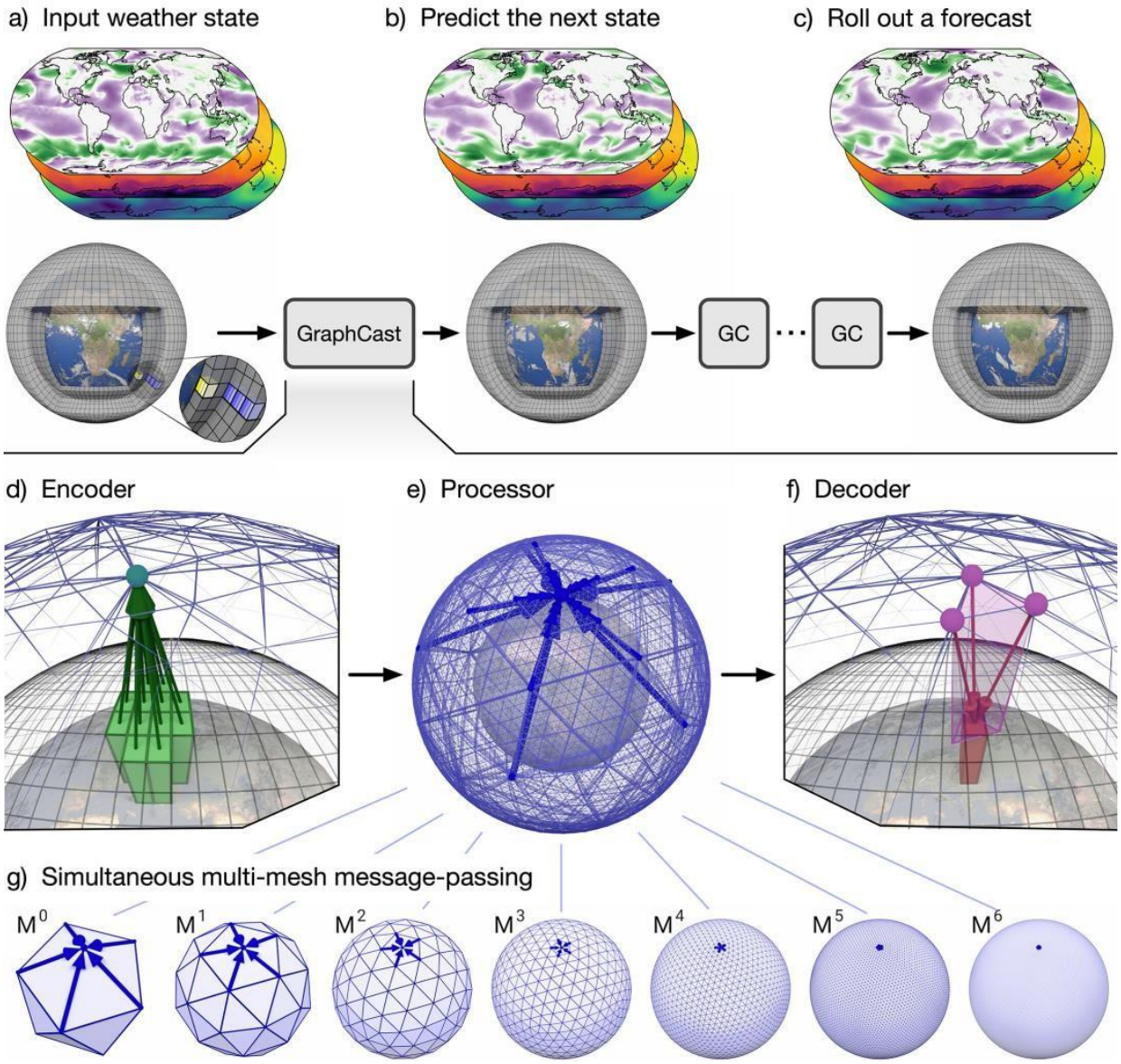

Рисунок 2.7 — Приклад роботи аналізу моделі погодних умов

Значною перевагою є можливість IoT-систем адаптуватися до умов у реальному часі. Наприклад, під час природних катастроф такі системи можуть автоматично змінювати стратегії управління ресурсами чи інформувати про загрози [15].

Таким чином, інтеграція штучного інтелекту, регресійних моделей та зовнішніх погодних сервісів є потужним інструментом для вдосконалення IoT-

систем. Ця комбінація дозволяє досягти високого рівня адаптивності, ефективності та точності в обробці даних. У перспективі такі технології стануть основою для багатьох інновацій, спрямованих на створення стійких і ефективних систем управління у різних сферах.

2.3 Вибір архітектури програмного забезпечення

2.3.1 Аналіз архітектурних стилів програмування

Архітектурні стилі програмування є ключовим етапом у процесі розробки програмного забезпечення, оскільки визначають фундаментальні принципи його структури, поведінки та взаємодії. Вибір архітектурного стилю впливає на масштабованість, продуктивність, безпеку, зручність у використанні й навіть на економічну ефективність створеного продукту. У цьому контексті розглянемо найбільш відомі та поширені архітектурні стилі, їхні характеристики та сфери застосування.

Монолітна архітектура є одним із найстаріших підходів до побудови програмних систем. Вона передбачає створення цілісного додатку, де всі функціональні модулі інтегровані в одне виконуване середовище. Така архітектура забезпечує простоту розробки на початкових етапах, оскільки всі компоненти взаємодіють без необхідності складних механізмів комунікації. Однак, зі збільшенням масштабів додатку монолітна архітектура стає важкою для підтримки. Зміна навіть невеликої частини програми може вимагати повторного тестування всього додатку, що підвищує ризик помилок і сповільнює процес розробки[16].

Сервіс-орієнтована архітектура (SOA) з'явилася як відповідь на обмеження моноліту. Вона передбачає розбиття програмного забезпечення на сервіси, які взаємодіють між собою через визначені інтерфейси. Ці сервіси можуть бути реалізовані різними мовами програмування та працювати на різних платформах, що забезпечує гнучкість і масштабованість системи. SOA активно використовується у корпоративних середовищах, де важлива інтеграція різнорідних систем і забезпечення їхньої сумісності.

Мікросервісна архітектура (рисунок 2.8) є еволюцією SOA і набрала популярності завдяки розвитку хмарних технологій та контейнеризації. У цьому підході кожен мікросервіс відповідає за вузькоспеціалізовану функцію, наприклад, управління користувачами або обробку платежів. Цей підхід забезпечує високу стійкість до помилок, адже збій одного мікросервісу не призводить до повної зупинки системи.

Крім того, мікросервіси можуть масштабуватися незалежно один від одного, що робить їх ідеальним вибором для динамічних середовищ із високим навантаженням. Водночас така архітектура ускладнює тестування, управління конфігурацією та оркестрацію сервісів, що вимагає використання додаткових інструментів, таких як Kubernetes [17].

Архітектура на основі подій (Event-Driven Architecture) є підходом, що орієнтується на обробку асинхронних подій. Цей стиль популярний у додатках, які вимагають низької затримки обробки даних, наприклад, у системах фінансового трейдингу або IoT. Події можуть передаватися через черги повідомлень, такі як Apache Kafka чи RabbitMQ, що забезпечує високу швидкість і масштабованість. Така архітектура дозволяє побудувати високопродуктивні системи, але вимагає ретельного проектування, щоб уникнути складнощів із трасуванням подій та підтримкою консистентності даних.

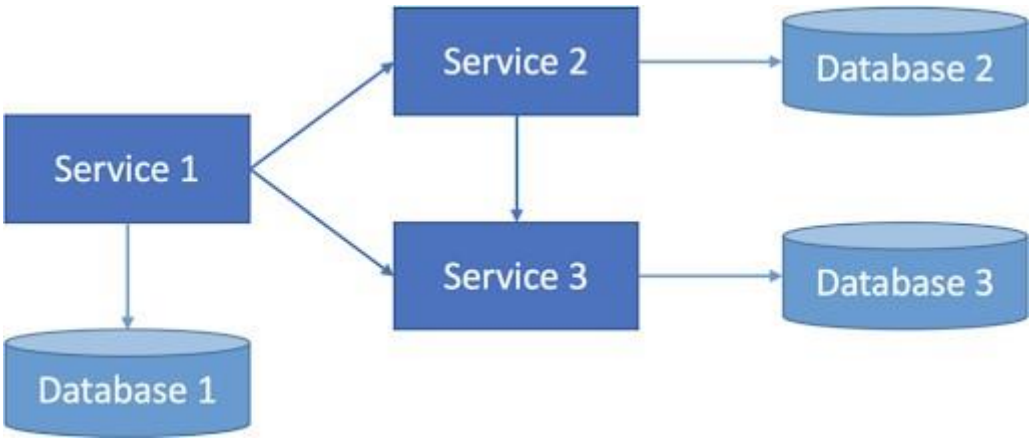

Рисунок 2.8 — Приклад роботи мікросервісної архітектури

Архітектура клієнт-сервер є одним із найпоширеніших стилів, що лежить в основі більшості сучасних веб-додатків. Вона розділяє функціонал між сервером,

який відповідає за зберігання, обробку даних та управління бізнес-логікою, і клієнтом, який забезпечує інтерфейс користувача. Цей підхід дозволяє забезпечити чіткий поділ відповідальності, що полегшує підтримку і розвиток додатків. Сучасні реалізації цього стилю включають REST, GraphQL та gRPC, які оптимізують комунікацію між клієнтом і сервером, роблячи її ефективною й легко масштабованою [18].

Прогресивна архітектура без сервера (Serverless Architecture) (рисунок 2.9) є новим і швидко зростаючим підходом. Вона дозволяє розробникам фокусуватися на бізнес-логіці, делегуючи управління серверами хмарним провайдерам, таким як AWS Lambda або Google Cloud Functions. Serverless Architecture ідеально підходить для розробки подійно-орієнтованих додатків або систем із непостійним навантаженням, адже оплата здійснюється лише за використані ресурси. Водночас цей стиль обмежує контроль над інфраструктурою і може створювати складнощі у відмовостійкості через залежність від хмарного провайдера [19].

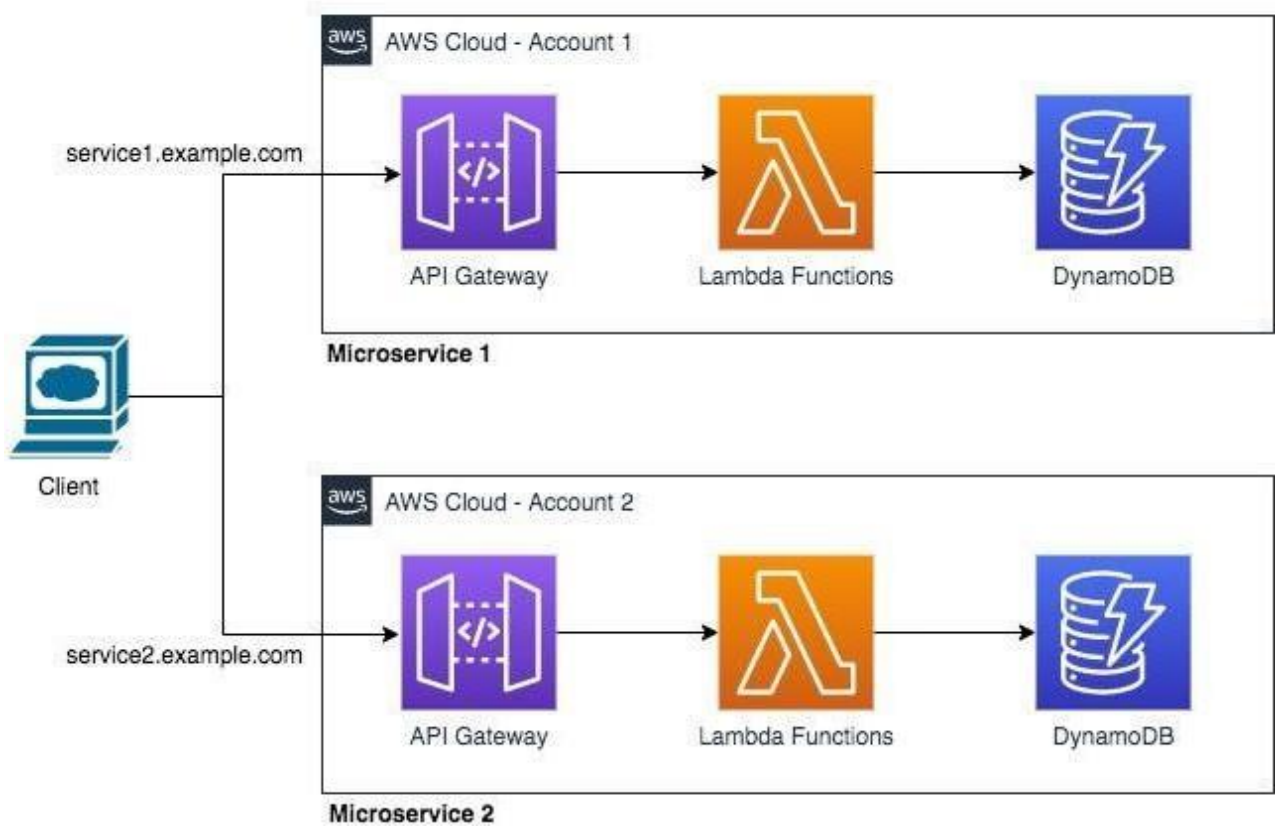

Рисунок 2.9 — Приклад архітектури без сервера

Таким чином, кожен архітектурний стиль має свої переваги та обмеження. Вибір стилю залежить від конкретних вимог до проекту, таких як масштабованість, продуктивність, безпека та доступність ресурсів. Ретельний аналіз і врахування особливостей кожного підходу дозволяють забезпечити розробку ефективного програмного забезпечення, що відповідає сучасним вимогам.

2.3.2 Проектування клієнт-серверної архітектури

Клієнт-серверна архітектура є фундаментальною моделлю організації програмного забезпечення, що використовується для створення багатьох сучасних додатків і систем. Ця архітектура базується на чіткому розподілі ролей між двома головними компонентами: клієнтом, який ініціює запити та взаємодіє з користувачем, і сервером, який обробляє ці запити, надає дані та виконує бізнес-логіку. Такий підхід забезпечує високу структурованість системи, гнучкість у розробці та легкість масштабування. Проектування клієнт-серверної архітектури потребує врахування багатьох аспектів, серед яких розподіл функціональності, методи комунікації, вибір протоколів і забезпечення продуктивності.

На етапі проектування важливо визначити функціональні обов'язки клієнта та сервера. Клієнтська частина здебільшого відповідає за взаємодію з кінцевим користувачем, забезпечуючи зручний і функціональний інтерфейс. Вона надсилає запити до сервера через визначені API (інтерфейси прикладного програмування), а також обробляє отримані дані для їхнього відображення. У деяких випадках клієнтська частина може виконувати певні обчислення або зберігати тимчасові дані, але основна бізнес-логіка залишається на стороні сервера. Сервер виконує роль центрального вузла, що забезпечує доступ до бази даних, обробляє складні операції та надає клієнту результати у вигляді відповідей на його запити.

Комунікація між клієнтом і сервером є критично важливим аспектом проектування. Сучасні системи здебільшого використовують протокол HTTP або його більш сучасний варіант HTTPS для передачі даних через мережу. Ці протоколи забезпечують універсальність та сумісність із веб-браузерами, що є важливим для багатьох веб-додатків. Інші проекти можуть використовувати

спеціалізовані протоколи, такі як WebSocket для реалізації постійного зв’язку, або gRPC для ефективної двосторонньої передачі даних. Важливо враховувати, що кожен протокол має свої переваги та обмеження. Наприклад, REST на базі HTTP забезпечує простоту реалізації та гнучкість у створенні API, але може бути менш ефективним у високонавантажених середовищах, порівняно з такими рішеннями, як gRPC.

Проектуючи клієнт-серверну архітектуру, також необхідно подбати про зберігання даних. У більшості випадків сервер є єдиним джерелом істини, тобто всі дані зберігаються централізовано в базі даних, до якої мають доступ лише серверні компоненти. Це дозволяє підтримувати консистентність даних та забезпечувати їхній захист. Для роботи з базою даних сервер зазвичай використовує ORM (Object-Relational Mapping) бібліотеки або прямі запити SQL. У випадках, коли додаток вимагає обробки великого обсягу даних у реальному часі, може бути доцільно використовувати NoSQL-бази, такі як MongoDB або Cassandra.

Масштабованість є одним із найважливіших аспектів проектування клієнт-серверної архітектури. Сервер повинен бути готовим обробляти зростаючу кількість запитів, забезпечуючи стабільну продуктивність навіть при значному збільшенні навантаження. Це може бути досягнуто за допомогою горизонтального масштабування, коли додаються нові сервери, або вертикального, коли збільшуються ресурси існуючих серверів. У сучасних системах часто використовується балансувальник навантаження, який рівномірно розподіляє запити між кількома серверами, що дозволяє забезпечити стабільну роботу системи під час пікових навантажень.

Безпека також є критичним аспектом у клієнт-серверній архітектурі. Оскільки клієнт і сервер взаємодіють через мережу, важливо забезпечити захист даних від перехоплення чи несанкціонованого доступу. Для цього використовуються шифрування (наприклад, протокол HTTPS), аутентифікація користувачів (JWT, OAuth 2.0) і авторизація. Сервер також має бути захищеним від таких атак, як SQL-ін'єкції, міжсайтовий скриптинг (XSS) та атаки типу «відмова в обслуговуванні» (DDoS) (рисунок 2.10) [20].

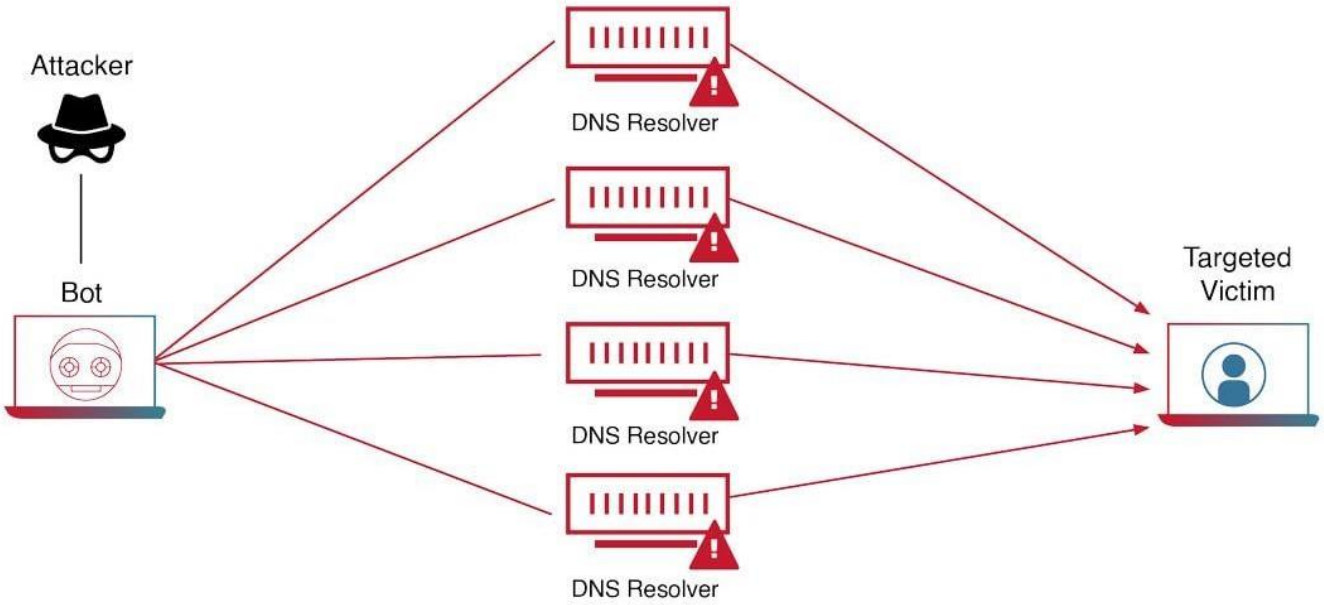

Рисунок 2.10 — Схема DDoS атаки

Особливу увагу варто приділити тестуванню та налагодженню клієнт-серверної архітектури. Оскільки клієнт і сервер є незалежними компонентами, їх можна тестувати окремо, але важливо також проводити інтеграційні тести, які перевіряють взаємодію між ними. Наприклад, сервер може генерувати несподівані помилки або повільно обробляти запити, що впливатиме на клієнтський досвід. Використання інструментів для моніторингу серверів, таких як Prometheus або ELK Stack (рисунок 2.11), дозволяє виявляти і вирішувати проблеми продуктивності в реальному часі.

Проектування клієнт-серверної архітектури є багатогранним процесом, який потребує врахування багатьох факторів. Чітке розподілення функцій між клієнтом і сервером, вибір протоколів комунікації, забезпечення безпеки та масштабованості, а також увага до продуктивності та тестування створюють основу для успішного програмного продукту. Цей підхід залишається одним із найбільш ефективних і поширених у сучасній розробці програмного забезпечення, адаптуючись до різноманітних вимог і умов використання.

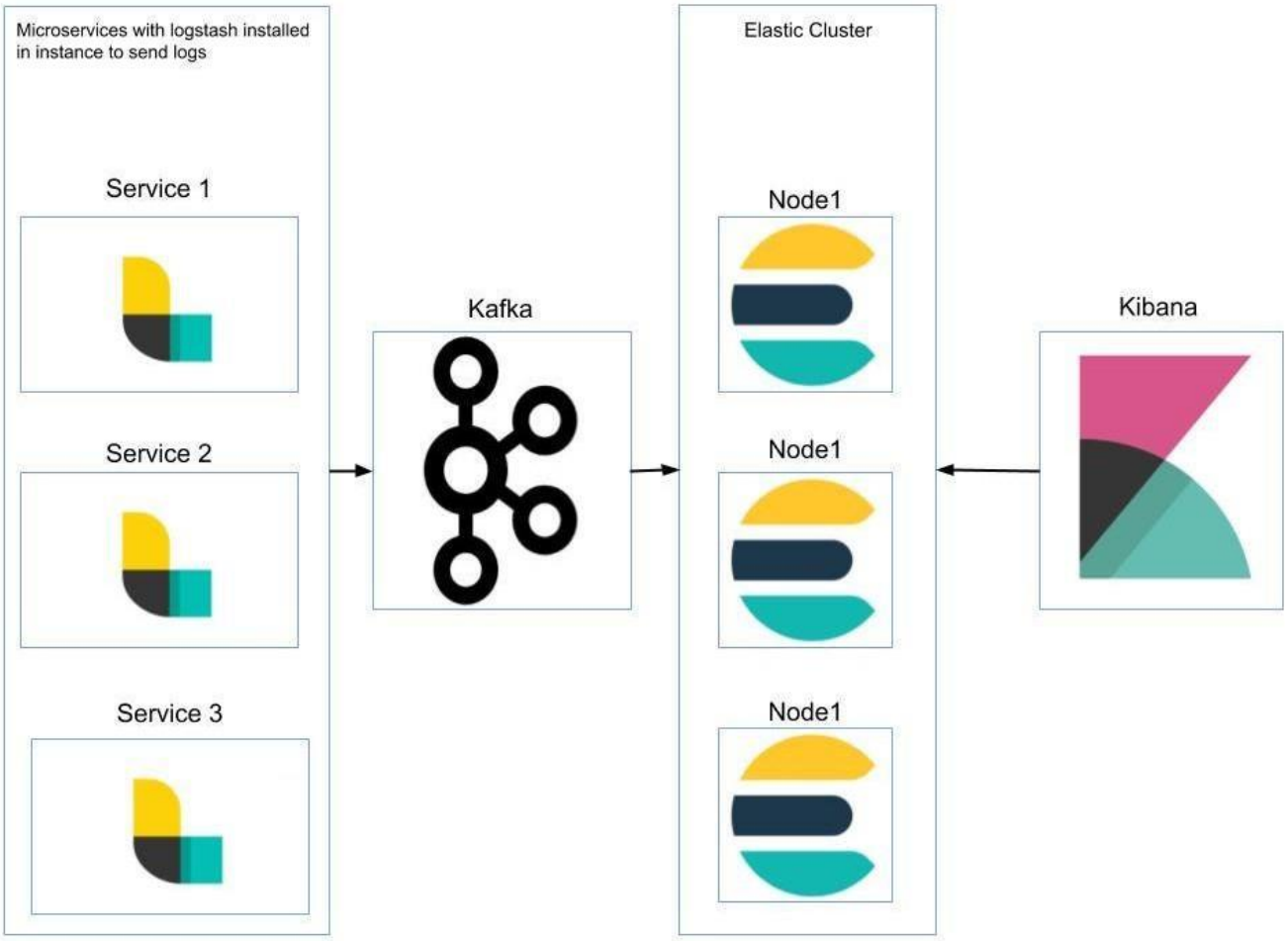

Рисунок 2.11 — Схема ELK Stack

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Розробка загальної структури програмної реалізації

Початок роботи системи включає в себе аналіз розміру екрану пристрою, з якого користувач зайшов у систему. Для цього автоматично визначається роздільна здатність екрану, і за допомогою цього аналізу, застосунок може адаптувати свій інтерфейс та вигляд до поточних параметрів пристрою, забезпечуючи оптимальне відображення і зручну навігацію для користувача. Нижче наведений детальний опис головних сторінок системи (рисунок 3.1).

Перша — сторінка авторизації в системі, є початковим кроком для входу користувача в обліковий запис та отримання доступу до функціональності системи. На цій сторінці користувачу необхідно ввести свої облікові дані, такі як ім'я користувача або електронну пошту та пароль.

Друга — сторінка реєстрації, яка дозволяє користувачам створювати особисті облікові записи і отримувати доступ до додаткового функціоналу. Зазвичай, на сторінці реєстрації користувачам пропонується заповнити форму з обов'язковими полями, такими як ім'я, електронна пошта та пароль.

Також можуть бути додаткові поля для введення інформації, яка вимагається в конкретному додатку. Після введення необхідних даних, користувач натискає кнопку "Зареєструватися", і його обліковий запис створюється в системі. Часто на сторінці реєстрації також можуть бути посилання на умови використання, політику конфіденційності або інші важливі документи.

Третя — вхід на сторінку користувача, є важливим етапом взаємодії з веб-додатком, який дозволяє користувачам отримати доступ до свого особистого облікового запису. Зазвичай, на сторінці входу користувачеві пропонується ввести свої облікові дані, такі як електронна пошта або ім'я користувача та пароль.

Ці дані перевіряються на відповідність зареєстрованим обліковим записам в системі. Якщо дані введені вірно, користувачу надається доступ до особистого кабінету або іншого захищеного функціоналу. У разі невірних облікових даних або відсутності облікового запису користувачу може бути відображена повідомлення про помилку або запропоновано повторити спробу входу.

Четверта — сторінка керування опаленням, є важливою частиною системи дистанційного керування опаленням і надає користувачу можливість контролювати і налаштовувати параметри опалювальної системи. На ній розміщено різноманітні елементи інтерфейсу, такі як кнопки та текстові поля, за допомогою яких користувач може встановлювати бажану температуру в окремих кімнатах своєї оселі.

Крім того, на сторінці відображені інформаційні дані, наприклад, поточна температура в окремих кімнатах. Завдяки сторінці керування опаленням користувач може зручно та ефективно керувати режимами опалення та створювати комфортні умови в своєму приміщенні.

П’ята — сторінка учасників сім’ї, яка дозволяє користувачу вести список учасників своєї сім'ї. На цій сторінці можна додавати нові записи з інформацією про кожного члена сім'ї, такі як ім'я, вік, стать, контактні дані тощо.

Крім того, користувач може редагувати, видаляти або переглядати існуючі записи. Також дозволяє зручно організовувати та відстежувати дані про членів сім'ї, що сприяє кращій комунікації та взаємодії всередині родини.

Шоста — сторінка оплати послуг, дозволяє користувачам здійснювати оплату за надані послуги. На цій сторінці користувачі можуть вибрати тип послуги, яку вони бажають оплатити, вказати необхідну кількість або період, а також здійснити оплату за допомогою різних доступних методів, таких як кредитні картки, електронні платіжні системи тощо.

Крім того, сторінка відображатиме інформацію про попередні платежі, історію транзакцій і надавати можливість завантаження квитанцій або отримання підтверджень оплати. Це дозволяє забезпечити зручний та безпечний процес оплати послуг для користувачів вебдодатку.

Сьома — сторінка сповіщень, є важливим елементом вебдодатку, який дозволяє користувачам отримувати різноманітні сповіщення та повідомлення. На цій сторінці користувачі можуть переглядати сповіщення про важливі події, такі як нові повідомлення, оновлення, запрошення або нагадування. Сторінка включає фільтри та сортування, що дозволяють швидко знайти потрібну інформацію.

Додатково було інтегровано модуль обробки даних на базі штучного інтелекту (ШІ), який аналізує отримані сповіщення та надає персоналізовані рекомендації. Наприклад, ШІ може пріоритезувати сповіщення або пропонувати дії, що оптимізують взаємодію користувача із системою.

Також реалізовано інтеграцію з погодними сервісами, яка дозволяє відображати сповіщення, пов’язані з погодними умовами, такими як попередження про шторм, зміну температури або інші погодні явища. Це розширює функціональність сторінки та додає практичну користь для користувачів.

Крім того, сторінка дозволяє взаємодіяти зі сповіщеннями: позначати їх як прочитані, видаляти або зберігати для подальшого використання. Завдяки цим функціям сторінка сповіщень забезпечує зручний спосіб отримувати, аналізувати та керувати сповіщеннями, підвищуючи ефективність комунікації та взаємодії у межах веб-додатку.

Восьма — сторінка налаштувань, являє собою одну із найважливіших сторінок. Тут здійснюється налаштовування різних опцій та особистих налаштування відповідно до своїх потреб і вимог. На цій сторінці користувачі можуть змінювати особисті дані, такі як ім'я, електронну пошту або пароль., зберігати зміни та виходити з акаунту.

Сторінка налаштувань забезпечує користувачам можливість контролювати та налаштовувати додаток згідно з їхніми актуальними персональними даними, що надасть змогу налаштовувати різні аспекти додатку відповідно до їхніх потреб і вподобань.

Об’єднані дані, отримані від сенсорів, погодних сервісів (рисунок 3.2) та користувача, обробляються за допомогою алгоритмів штучного інтелекту (ШІ). ШІ виконує аналіз, прогнозування та генерує рекомендації на основі отриманих даних. Це дозволяє системі адаптуватися до змін і забезпечувати високу точність результатів.

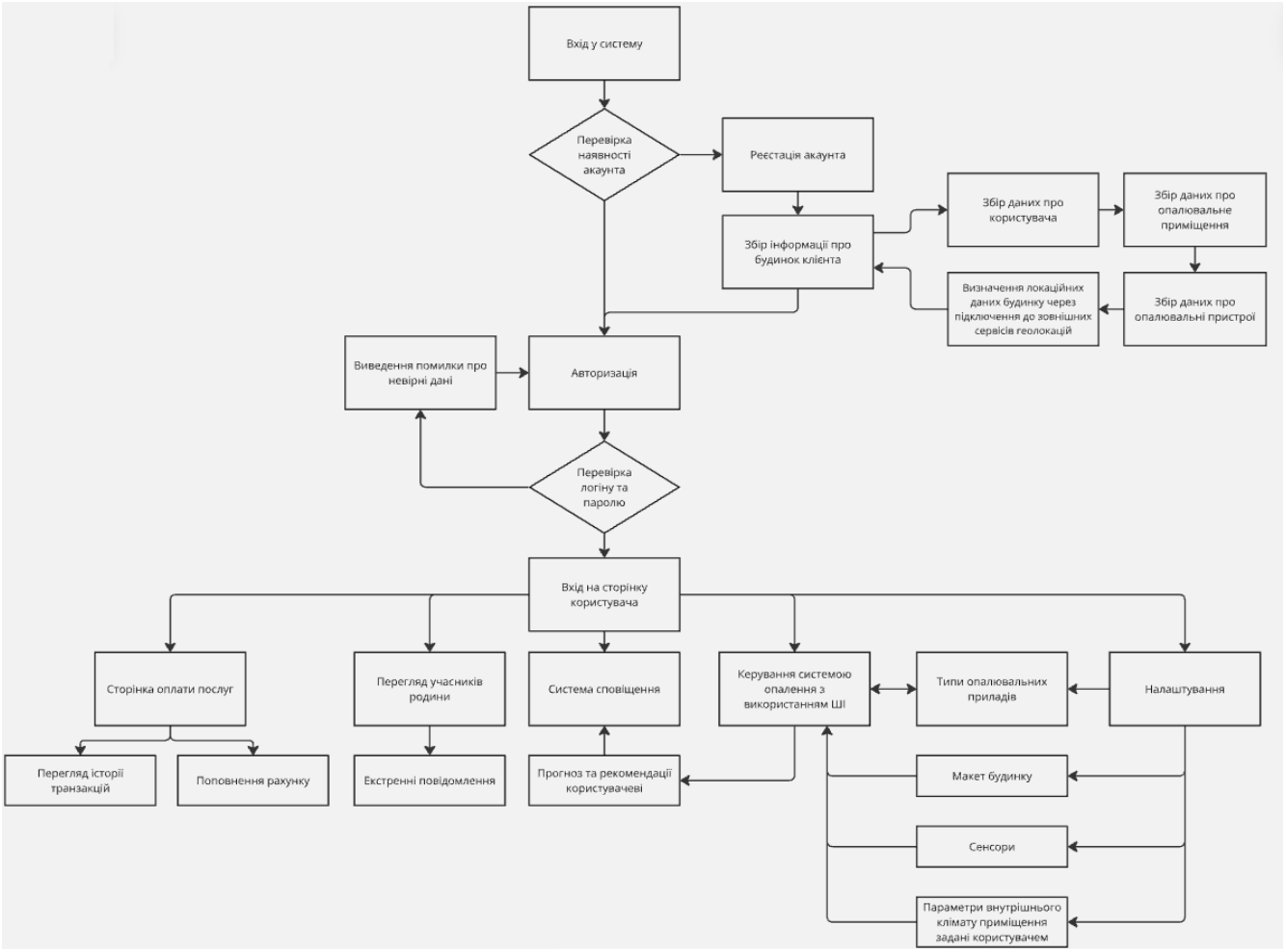

Рисунок 3.1 — Структура системи дистанційного керування системи опалення

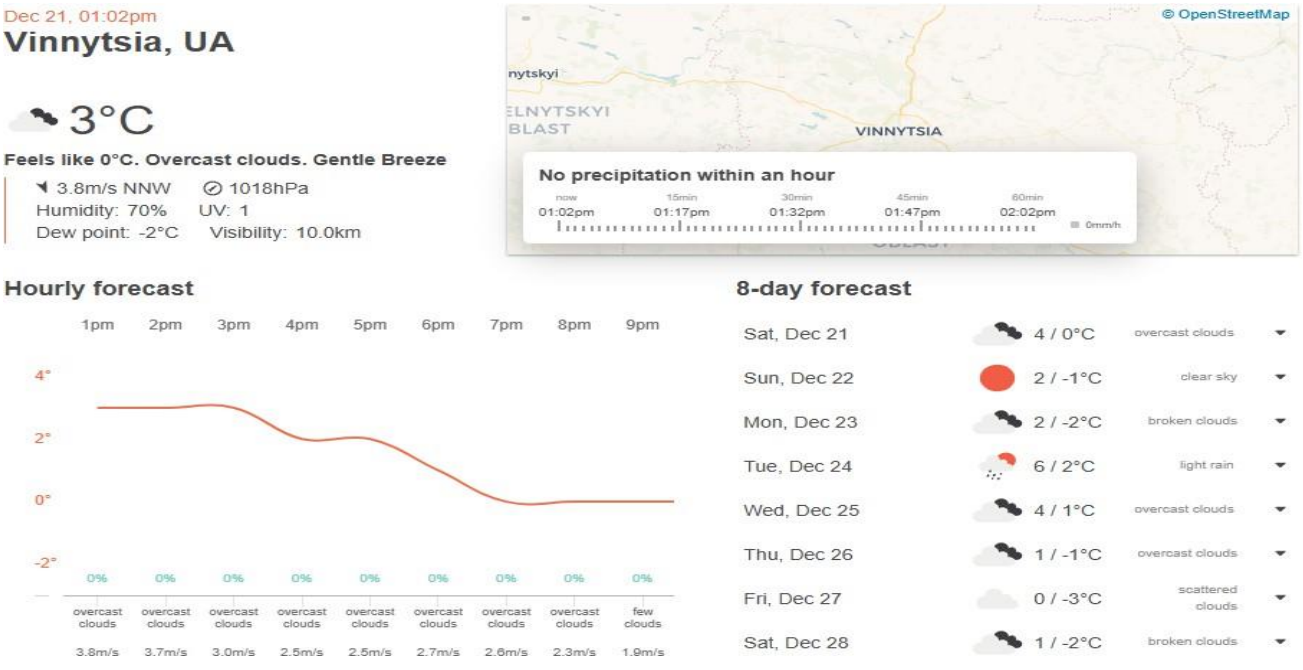

Рисунок 3.2 — Приклад роботи OpenWeatherMap

Результатом роботи системи на рівні обробки даних є управління пристроями та налаштуваннями. Це означає, що система може автоматично регулювати роботу пристроїв, наприклад, змінювати інтенсивність освітлення, вмикати чи вимикати обігрівачі залежно від прогнозу температури або включати зрошувальну систему при виявленні низького рівня вологості.

Останній етап на рівні обробки даних — надання прогнозів та рекомендацій для користувача. Ці прогнози можуть включати, наприклад, інформацію про необхідність змінити налаштування системи, передбачення несприятливих погодних умов або рекомендації щодо оптимізації ресурсів.

Важливою частиною архітектури є інтеграція з зовнішнім сервісом OpenWeatherMap. Ця інтеграція забезпечує доступ до глобальних даних про погодні умови, які доповнюють локальні вимірювання сенсорів. Наприклад, у сільському господарстві дані про очікувані опади можуть допомогти зменшити витрати на полив.

Ще одним важливим аспектом схеми є її модульна структура. Розподіл на три рівні дозволяє легко розширювати або змінювати функціонал системи. Наприклад, можна додати нові сенсори або інтегрувати інші зовнішні сервіси без значного впливу на існуючу архітектуру.

Завдяки використанню Wi-Fi та Ethernet система може працювати в різних умовах, від невеликих приміщень до великих виробничих площ. Мережевий рівень забезпечує стабільний зв’язок між усіма компонентами, що є важливим для передачі даних у реальному часі.

Рівень пристроїв дозволяє працювати з великою кількістю сенсорів, які можуть бути розташовані у різних місцях. Це забезпечує високу масштабованість системи та можливість збору даних з великої території.

Інтеграція ШІ у систему робить її не лише інформативною, але й автономною. Наприклад, ШІ може виявляти аномалії в даних і повідомляти про можливі проблеми ще до їх виникнення. Це дозволяє уникати збоїв у роботі обладнання або інших критичних ситуацій.

Система також підтримує введення даних користувачем, що дає змогу враховувати додаткові параметри або коригувати автоматично зібрані дані. Це підвищує гнучкість системи та дозволяє адаптувати її до унікальних потреб.

Розроблена схема демонструє чітке розмежування функцій на різних рівнях, що спрощує управління системою. Завдяки цьому можна швидко локалізувати та усунути можливі несправності на конкретному рівні без порушення роботи всієї системи.

Загалом, запропонована архітектура є універсальною і може бути адаптована до різних сфер застосування, таких як сільське господарство, логістика, управління будівлями та інші галузі. Вона забезпечує високу точність аналізу даних, автоматизацію процесів та можливість взаємодії з користувачем.

Така структура сприяє підвищенню ефективності використання ресурсів, зменшенню витрат та підвищенню екологічності процесів, що є особливо важливим у сучасних умовах. Завдяки використанню новітніх технологій система дозволяє вирішувати складні завдання та адаптуватися до мінливих умов.

3.2 Аналіз та вибір технологій для реалізації системи

3.2.1 Технології розробки серверної частини

Розробка серверної частини програмного забезпечення є важливим аспектом створення сучасних систем, оскільки саме сервер забезпечує обробку бізнес-логіки, управління даними та взаємодію з клієнтськими додатками. Вибір технологій для реалізації серверної частини залежить від вимог до продуктивності, масштабованості, безпеки та сумісності із зовнішніми системами. На цьому етапі враховуються мови програмування, фреймворки, бази даних, серверні платформи та протоколи комунікації, які разом утворюють основу для створення надійного серверного рішення.

Однією з найпоширеніших мов програмування для розробки серверної частини є JavaScript у поєднанні з платформою Node.js. Завдяки своїй подієво-орієнтованій моделі та неблокуючому вводу/виводу, Node.js забезпечує високу продуктивність і ефективність у роботі з великою кількістю одночасних запитів.

Ця технологія ідеально підходить для створення веб-додатків у реальному часі, наприклад чатів, стрімінгових сервісів або інтернет-магазинів із високим трафіком. До переваг Node.js належить велика кількість готових модулів у репозиторії npm, що дозволяє значно скоротити час розробки.

Іншим популярним вибором є Python, особливо у поєднанні з фреймворком Django або Flask. Python відомий своєю простотою, читабельністю коду та широким спектром бібліотек, що робить його ідеальним для швидкого створення прототипів і складних серверних додатків. Django забезпечує високу швидкість розробки завдяки наявності вбудованих інструментів, таких як ORM для роботи з базами даних, система автентифікації користувачів та засоби адміністрування. Flask, у свою чергу, надає більше свободи завдяки своїй мінімалістичній природі, що дозволяє розробникам обирати додаткові бібліотеки залежно від потреб проєкту.

У корпоративних додатках часто використовуються Java та фреймворк Spring Boot. Java вважається стандартом для створення масштабованих серверних додатків завдяки своїй стабільності, високій продуктивності та можливостям інтеграції з іншими системами. Spring Boot спрощує створення Java-додатків, автоматизуючи конфігурацію, що дозволяє зосередитися на бізнес-логіці. Завдяки вбудованій підтримці REST API, інтеграції з базами даних та системам обробки повідомлень, Spring Boot є одним із найкращих виборів для створення серверних додатків високого рівня складності.

У сфері високонавантажених систем усе більшої популярності набуває використання Golang (Go). Ця мова розроблена спеціально для створення продуктивних і масштабованих серверних додатків. Завдяки своїй простоті та можливостям роботи з паралелізмом Go є чудовим вибором для проєктів, що вимагають максимальної швидкодії, наприклад API для фінансових операцій або ігрових серверів. Крім того, Go має потужну стандартну бібліотеку, що дозволяє створювати серверні додатки без використання зовнішніх залежностей.

У деяких проєктах, зокрема пов’язаних із аналізом даних або машинним навчанням, використовується R або Julia. Ці мови забезпечують ефективну обробку

великих обсягів інформації та інтеграцію з різноманітними бібліотеками для аналізу даних, що робить їх придатними для створення серверних частин спеціалізованих додатків.

Окрім вибору мови програмування, важливу роль у розробці серверної частини відіграє інфраструктура. Використання контейнеризації за допомогою Docker дозволяє ізолювати серверні додатки в незалежні середовища, що спрощує їхнє розгортання, масштабування та тестування. У поєднанні з оркестраторами контейнерів, такими як Kubernetes, це забезпечує надійну основу для побудови хмарних серверних додатків.

Для зберігання даних серверна частина може використовувати різні типи баз даних залежно від потреб проєкту. Реляційні бази даних, такі як PostgreSQL чи MySQL, забезпечують структуроване зберігання даних і підтримку складних запитів SQL. NoSQL-бази, як-от MongoDB чи Cassandra, дозволяють працювати з неструктурованими даними й масштабуватися горизонтально. Для серверів реального часу також можуть використовуватися системи зберігання даних у пам’яті, такі як Redis або Memcached, які забезпечують швидкий доступ до тимчасових даних.

Забезпечення безпеки є ще одним критично важливим аспектом розробки серверної частини. Сервер повинен гарантувати захист даних користувачів і запобігати потенційним атакам, таким як SQL-ін'єкції, XSS або CSRF. Для цього використовуються бібліотеки для автентифікації, такі як JWT (JSON Web Token), а також сервіси авторизації, такі як OAuth 2.0.

Таким чином, вибір технологій для розробки серверної частини залежить від специфіки проєкту, вимог до продуктивності та масштабованості, а також доступних ресурсів і досвіду команди розробників. Успішна реалізація серверної частини передбачає правильне поєднання технологій, що забезпечить високу ефективність і стійкість програмного забезпечення.

3.2.2 Технології розробки клієнтської частини

Розробка клієнтської частини програмного забезпечення є важливим етапом

створення сучасних додатків, адже саме клієнтська частина є тим інтерфейсом, через який користувач взаємодіє із системою. Вибір технологій для клієнтської частини визначає функціональність, продуктивність, естетику та зручність у використанні програми. У розробці клієнтської частини враховуються як технічні аспекти, так і особливості користувацького досвіду, що створює потребу в гармонійному поєднанні дизайну, програмування та інтеграції з серверною частиною.

Сучасні веб-додатки часто розробляються із застосуванням HTML, CSS і JavaScript — трьох базових технологій, які визначають структуру, стиль і функціональність клієнтської частини. HTML забезпечує створення каркасу веб-сторінок, CSS відповідає за їхній стиль і оформлення, а JavaScript додає інтерактивність і динамічність. Завдяки цим технологіям можна створювати зручні інтерфейси з високим рівнем інтерактивності, що відповідають вимогам сучасних користувачів.

Для спрощення та прискорення розробки сучасних клієнтських додатків використовуються фреймворки та бібліотеки. Серед найпопулярніших бібліотек для JavaScript — React, створений компанією Facebook. React базується на компонентах, що дозволяє розробникам розбивати інтерфейс на невеликі частини, які можуть бути повторно використані в різних місцях програми. Цей підхід спрощує підтримку коду та забезпечує високу продуктивність завдяки віртуальному DOM, який мінімізує оновлення реального DOM.

Ще одним потужним інструментом є фреймворк Angular, розроблений Google. Angular використовує TypeScript — мову програмування, що є розширенням JavaScript із підтримкою статичної типізації. Завдяки своєму модульному підходу Angular дозволяє створювати складні клієнтські додатки з чіткою структурою. Вбудовані інструменти Angular, такі як двостороннє зв’язування даних та система залежностей, значно полегшують розробку динамічних інтерфейсів.

Іншим популярним вибором є фреймворк Vue.js, який вирізняється простотою у використанні та високою продуктивністю. Vue.js поєднує найкращі

риси React і Angular, зберігаючи простоту налаштування та можливість поступового масштабування. Завдяки легкості інтеграції Vue.js часто використовується для додавання інтерактивних елементів до існуючих проєктів або для створення нових додатків із нуля.

У розробці мобільних додатків, що функціонують як клієнтська частина, активно використовуються такі технології, як Flutter, React Native і Swift. Flutter, створений Google, дозволяє розробляти додатки для Android і iOS із використанням єдиного коду на мові Dart. Завдяки своїй продуктивності та вбудованим можливостям для створення адаптивного дизайну Flutter стає популярним вибором для багатоплатформних проєктів. React Native від Facebook також дозволяє створювати кросплатформні мобільні додатки, використовуючи мову JavaScript, що спрощує розробку завдяки єдиній екосистемі. Swift, у свою чергу, є основною мовою для створення нативних додатків для платформи iOS і забезпечує високу продуктивність та інтеграцію з екосистемою Apple.

Для забезпечення адаптивності клієнтської частини, що дозволяє додатку коректно відображатися на різних пристроях, використовуються такі технології, як CSS Grid, Flexbox та Bootstrap. Адаптивний дизайн став стандартом у розробці, оскільки сучасні користувачі часто взаємодіють із додатками через різноманітні пристрої — від смартфонів до великих моніторів. Bootstrap є фреймворком для створення адаптивного дизайну, що містить готові компоненти та стилі, які значно спрощують процес розробки.

Забезпечення продуктивності клієнтської частини є ще одним ключовим аспектом. Для зменшення часу завантаження сторінок застосовуються методи оптимізації, зокрема мінімізація JavaScript і CSS файлів, використання інструментів, таких як Webpack, і кешування статичних ресурсів. У додатках із великою кількістю користувачів важливо також використовувати інструменти для моніторингу та налагодження, такі як Google Lighthouse, які дозволяють виявляти вузькі місця у продуктивності.

Інтеграція клієнтської частини із сервером здійснюється через API, що використовують протоколи HTTP або WebSocket. Найпоширенішим форматом

обміну даними є JSON, оскільки він простий у використанні та легко інтегрується з більшістю мов програмування. Інтеграція також вимагає врахування безпеки, включно з захистом від атак типу XSS та CSRF. Для цього клієнтська частина використовує токени автентифікації та верифікацію серверних відповідей.

Сучасна розробка клієнтської частини також передбачає використання інструментів для тестування. Наприклад, фреймворки Jest і Mocha забезпечують автоматизоване тестування JavaScript-коду, що дозволяє виявляти помилки на ранніх етапах розробки. Інструменти для візуального тестування, як-от Storybook, допомагають перевіряти окремі компоненти користувацького інтерфейсу в ізольованому середовищі.

Таким чином, технології розробки клієнтської частини охоплюють широкий спектр інструментів і підходів, які забезпечують створення ефективних, естетично привабливих і зручних у використанні додатків. Правильний вибір технологій залежить від вимог проєкту, технічних обмежень та очікувань кінцевих користувачів, що дозволяє створювати програмні рішення високої якості.

3.3 Вибір та обґрунтування бази даних

Вибір бази даних є одним із ключових етапів у розробці програмних систем, оскільки від того, яку технологію обрано для зберігання та управління даними, залежить не тільки ефективність роботи додатку, але й масштабованість, швидкість обробки запитів, безпека та можливості інтеграції з іншими компонентами. Процес вибору бази даних передбачає детальне дослідження вимог до системи, типів даних, обсягу інформації, частоти запитів, рівня навантаження на систему, а також особливостей майбутнього використання даних. Однією з основних категорій баз даних є реляційні та нереляційні системи. Вибір між ними залежить від архітектурних вимог проєкту, а також від специфіки роботи з даними.

Реляційні бази даних (RDBMS) є найбільш популярними і широко використовуваними для розробки додатків, що потребують чіткої структури даних і складних запитів. Вони організовують дані у вигляді таблиць, де кожна таблиця складається з рядків і стовпців. Реляційні системи підтримують стандартну мову

запитів SQL (Structured Query Language), яка дозволяє здійснювати операції з даними, такі як вибірка, вставка, оновлення та видалення. Найпоширенішими представниками реляційних баз є MySQL, PostgreSQL, Oracle Database та Microsoft SQL Server.

Реляційні бази даних є ефективними для систем, де дані мають чітко визначену структуру з обов'язковими зв'язками між різними таблицями. Оскільки система повинна працювати з величезними масивами даних, які потребують складних запитів на пошук, фільтрацію, агрегацію та обробку, реляційна база даних дає можливість забезпечити належну продуктивність і стабільність. Крім того, реляційні бази даних надають потужні засоби для підтримки транзакцій, що є критично важливим для додатків, де точність та консистентність даних мають особливе значення. Вони підтримують атомарність, узгодженість, ізоляцію та стійкість (ACID), що робить їх ідеальними для бізнес-додатків, фінансових та інших критичних систем.

У порівнянні з реляційними, нереляційні бази даних (NoSQL) краще підходять для зберігання неструктурованих або частково структурованих даних. Вони можуть бути корисними для великих даних, що часто змінюються або не мають чіткої схеми. Однак відсутність єдиної структури даних може ускладнити виконання складних запитів і знизити ефективність роботи з даними, що вимагає швидких і точних результатів. Найпоширенішими типами нереляційних баз є документоорієнтовані бази (наприклад, MongoDB), колонкові бази даних (наприклад, Cassandra), графові бази даних (наприклад, Neo4j) та бази даних з ключ-значення (наприклад, Redis).

Вибір бази даних залежить від типу системи, яку потрібно побудувати. У разі системи, яка працює з великими обсягами структурованих даних, що потребують складних запитів і гарантованої цілісності, реляційна база даних є найбільш підходящим вибором. Вона забезпечує високу швидкість обробки даних, підтримує механізми безпеки, масштабованість та дає можливість здійснювати складні аналізи і генерацію звітів, що є важливими для таких систем, як системи для обробки та аналізу даних з інтернету речей (IoT).

Одним з основних критеріїв вибору бази даних для нашої системи є необхідність ефективного управління даними, що збираються з різних джерел — сенсорів IoT, погодних сервісів, а також локаційних даних. Оскільки дані від сенсорів мають чітку структуру та потребують інтеграції з іншими даними, такими як часова інформація, координати місцезнаходження, а також зміни параметрів, реляційна база даних є найбільш оптимальним вибором. Вона дозволяє ефективно зберігати дані, легко здійснювати запити на їх обробку та аналіз, а також здійснювати масштабування за рахунок кластеризації і реплікації, що є важливим для систем, що працюють у реальному часі.

Реляційні бази даних підтримують зв’язки між таблицями через механізм зовнішніх ключів, що дозволяє зберігати логічні зв’язки між різними типами даних. Наприклад, дані про температурні показники, зібрані від різних сенсорів, можна легко об’єднати з даними про локацію будинку або погодні умови. Це дозволяє здійснювати складні запити для аналізу та прогнозування, а також інтегрувати додаткові сервіси, такі як система опалення і кондиціонування, з урахуванням умов навколишнього середовища.

У разі систем, що взаємодіють з клієнтами через інтерфейс, реляційні бази даних також забезпечують швидку і надійну обробку запитів, що дозволяє користувачам отримувати оперативні дані про стан системи, її параметри та прогнозовані зміни. Оскільки реляційні бази даних забезпечують транзакційність, це дає змогу зберігати цілісність даних під час їх обробки, що є критично важливим для систем, що працюють у режимі реального часу.

Враховуючи усі ці фактори, можна зробити висновок, що для нашої системи, що обробляє дані з інтернету речей, обираючи між реляційними і нереляційними базами даних, реляційний підхід є найбільш відповідним. Вибір реляційної бази даних забезпечить високий рівень структурованості, можливість складних запитів, підтримку транзакцій та цілісність даних, що є необхідним для ефективної роботи системи в умовах великих обсягів даних та високих вимог до надійності та масштабованості.

Таблиця users (рисунок 3.3) містить дані про користувачів системи. Кожен

запис зберігає унікальний ідентифікатор користувача («ID»), його логін («login») і зашифрований пароль («password»). Також зберігається особиста інформація, така як ім'я («first_name»), прізвище («second_name»), по батькові («last_name»), адреса проживання («address»), контактний номер телефону («phone») і місце розташування («location»). Поле «location» підтримує географічну координату (тип «POINT»), що дозволяє системі визначати місце розташування користувача для адаптації послуг на основі геолокації. Створення таблиці:

```
CREATE TABLE users (
    ID INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    login VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    second_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    address VARCHAR(100) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    location VARCHAR(100) DEFAULT NULL
);
```

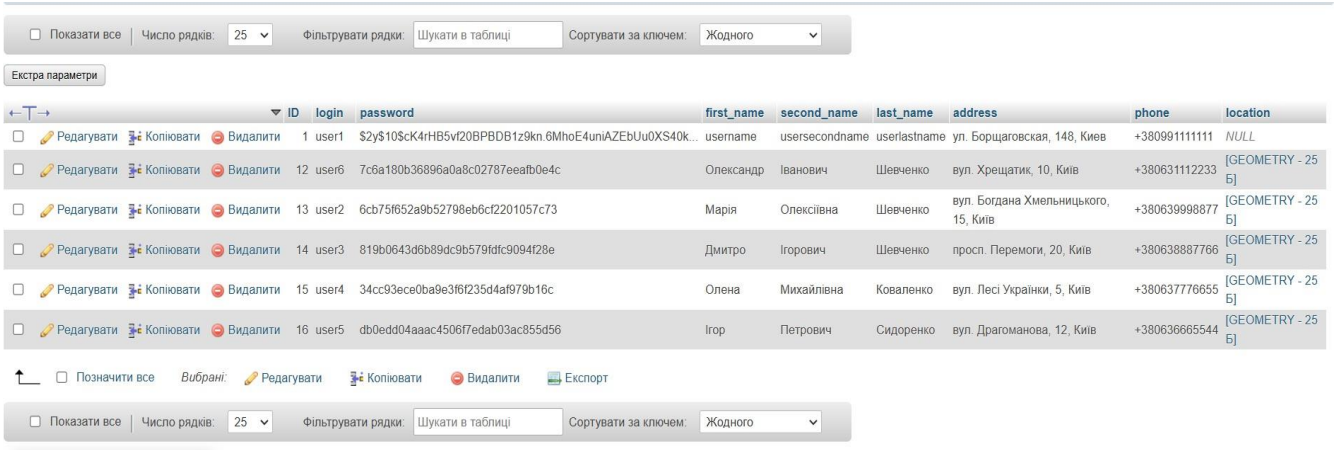

Рисунок 3.3 — Створена таблиця USERS

Таблиця active_user (рисунок 3.4) використовується для відстеження активних користувачів у системі.

Поле «user_id» зберігає ідентифікатор користувача, який має активний статус, з посиланням на таблицю «users». У разі видалення користувача всі пов'язані записи автоматично видаляються завдяки обмеженням «ON DELETE CASCADE».

Створення таблиці:

```
CREATE TABLE active_user (
   ID INT PRIMARY KEY AUTO_INCREMENT,
   user_id INT NOT NULL,
   FOREIGN KEY (user_id) REFERENCES users(ID) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Таблиця rooms (рисунок 3.3) зберігає дані про кімнати, які підключені до системи. Поле «roomID» є унікальним ідентифікатором кожної кімнати, а «room» містить її назву.

У таблиці також зберігається поточна температура в кімнаті («temp») та ідентифікатор користувача («user_id»), який володіє кімнатою. Це дозволяє відстежувати умови всередині кожної кімнати та адаптувати роботу системи IoT для забезпечення комфорту. Створення таблиці:

```
CREATE TABLE rooms (
   roomID INT PRIMARY KEY AUTO_INCREMENT,
   room VARCHAR(50) NOT NULL,
   temp INT NOT NULL,
   user_id INT NOT NULL,
   FOREIGN KEY (user_id) REFERENCES users(ID) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Рисунок 3.4 — Створена таблиця ROOMS

Таблиця transactions (рисунок 3.5) призначена для зберігання інформації про фінансові транзакції, пов'язані з користувачами. Поле «service» вказує, за яку послугу здійснено платіж, а «amount» — суму транзакції. Поле «date» містить дату транзакції, а «user_id» посилається на користувача, який здійснив операцію. Ці дані можуть бути використані для створення звітів про витрати або розрахунку обслуговування системи. Створення таблиці:

```
CREATE TABLE transactions (
    id INT PRIMARY KEY AUTO_INCREMENT,
    service VARCHAR(50) NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    date DATE NOT NULL,
    user_id INT NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(ID) ON DELETE CASCADE ON UPDATE CASCADE
);
```

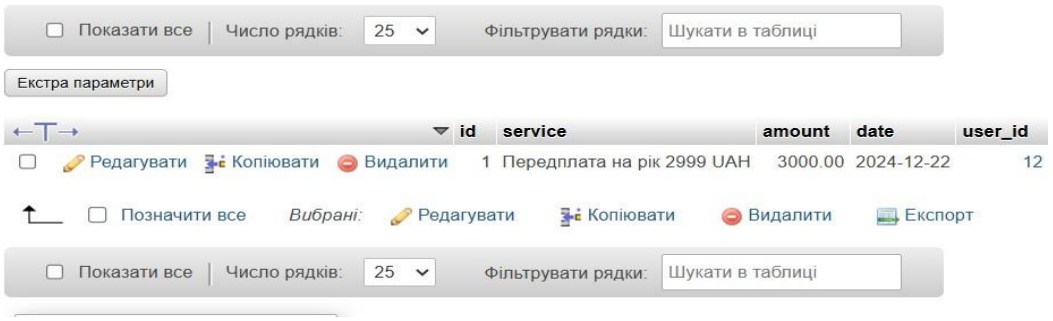

Рисунок 3.5 — Створена таблиця TRANSACTIONS

Таблиця weather_forecast (рисунок 3.6) зберігає прогнози погоди для певних локацій. Поле «location» вказує географічне місце, для якого актуальний прогноз. Поля «forecast_date» і «temperature» містять дату прогнозу та очікувану температуру відповідно.

Поле «weather_condition» описує погодні умови (наприклад, «сонячно», «дощ»), а додаткові поля «humidity» і «wind_speed» містять інформацію про вологість і швидкість вітру. Поле «created_at» дозволяє відстежувати час створення запису. Створення таблиці:

```
CREATE TABLE weather_forecast (
    id INT PRIMARY KEY AUTO_INCREMENT,
    location VARCHAR(100) NOT NULL,
    forecast_date DATE NOT NULL,
    temperature DECIMAL(5, 2) NOT NULL,
    weather_condition VARCHAR(50) NOT NULL,
    humidity INT DEFAULT NULL,
    wind_speed DECIMAL(5, 2) DEFAULT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Рисунок 3.6 — Створена таблиця WEATHER_FORECAST

Таблиця heating_control (рисунок 3.7) призначена для управління системою опалення в IoT. Поле «status» визначає поточний стан опалення (включено або вимкнено), а «action_time» фіксує час останньої дії. Поле «user_confirmation» вказує, чи підтверджено користувачем виконання дії. Таблиця дозволяє автоматизувати процеси опалення залежно від умов, таких як температура в кімнаті або погодні умови.

Створення таблиці «heating_control»:

```
CREATE TABLE heating_control (
    id INT PRIMARY KEY AUTO_INCREMENT,
    status ENUM('on', 'off') NOT NULL,
    action_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    user_confirmation BOOLEAN DEFAULT FALSE
);
```

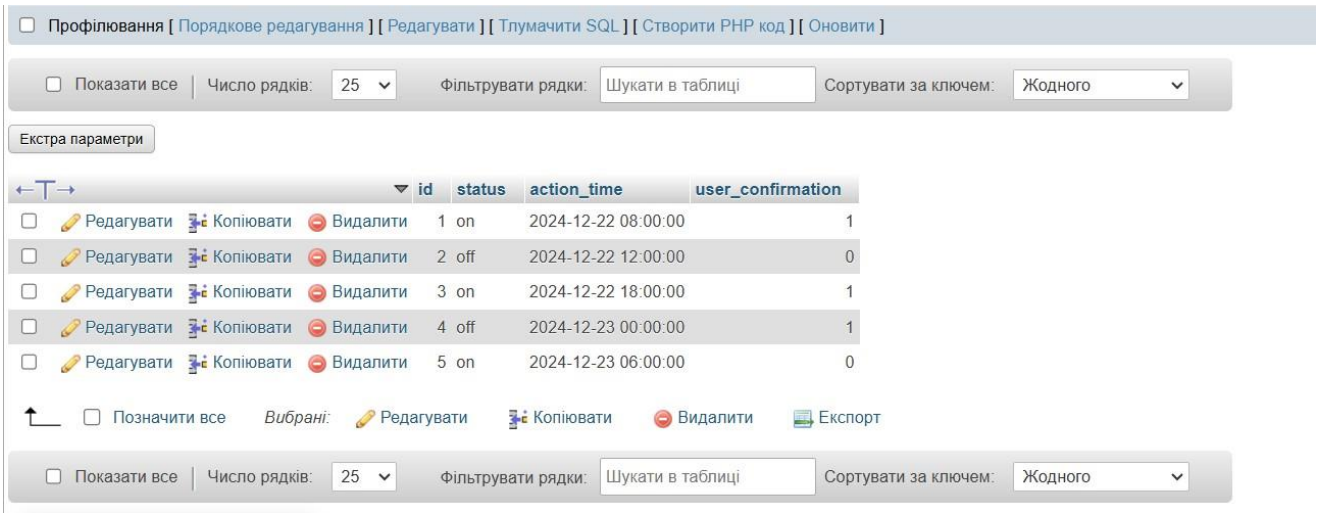

Рисунок 3.7 — Створена таблиця HEATING_CONTROL

Ця база даних забезпечує збереження ключових даних для роботи IoT-системи (рисунок 3.8). Вона підтримує управління користувачами, моніторинг кімнат, прогнозування погоди та контроль за фінансовими операціями. Використання географічних даних дозволяє адаптувати послуги до місцезнаходження користувача, а зв'язки між таблицями через зовнішні ключі («FOREIGN KEY») забезпечують узгодженість даних.

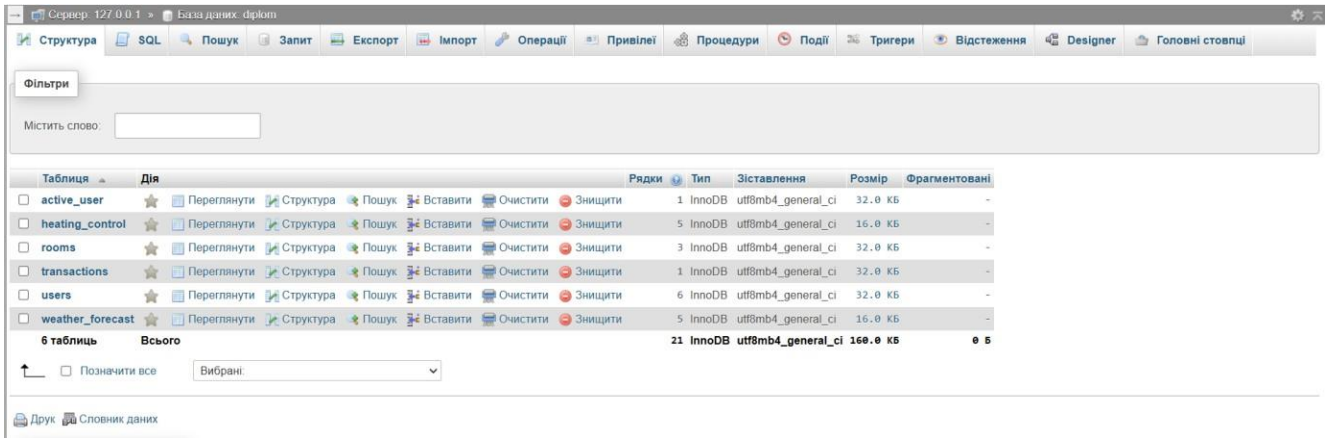

Рисунок 3.8 — Вигляд бази даних зі створеними таблицями

Система дозволяє інтегрувати автоматизовані рішення, такі як контроль температури чи генерація прогнозів погоди, завдяки даним, які зберігаються у таблицях, і забезпечує стабільну та масштабовану інфраструктуру для роботи в реальному часі.

4 ТЕСТУВАННЯ СИСТЕМИ

4.1 Тестування розробленої системи

На рисунку 4.1 зображено сторінку реєстрації, де користувач може створити свій обліковий запис. Форма реєстрації включає поля для введення імені, прізвища, адреси електронної пошти, номера телефону, адреси проживання та пароля. Реєстрація завершується підтвердженням через код, надісланий на електронну пошту або телефон.

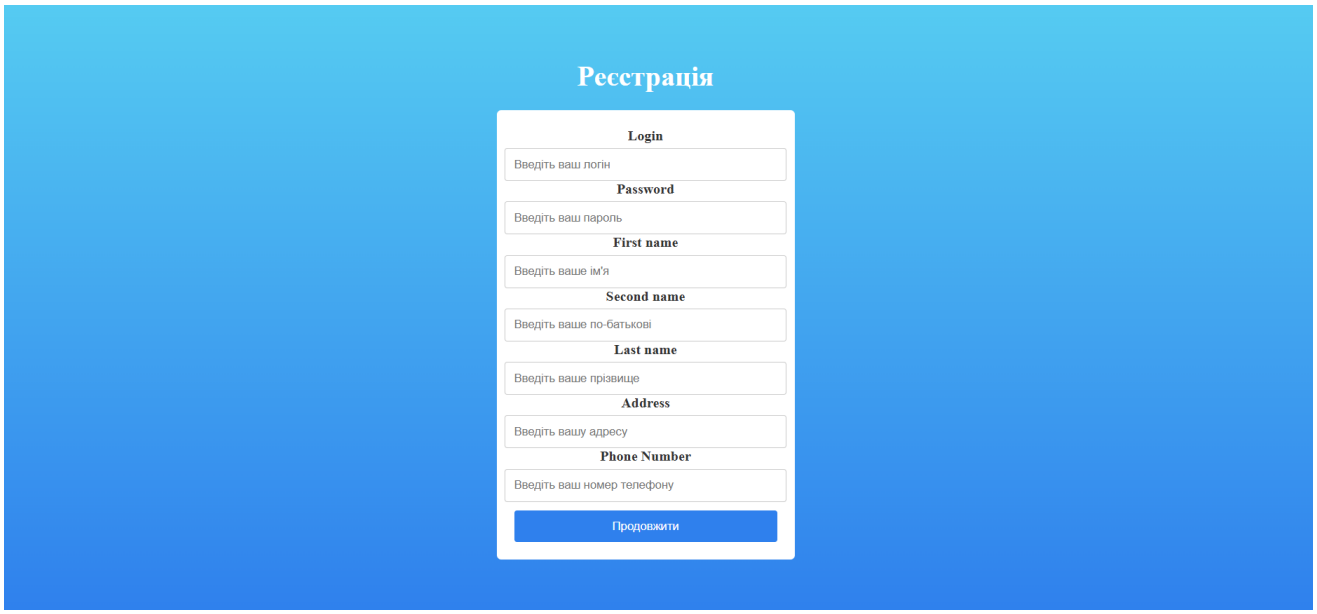

Рисунок 4.1 — Сторінка реєстрації

На рисунку 4.2 зображено сторінку входу в систему (логін). Користувач вводить свої логін та пароль у відповідні поля. У випадку забутого пароля доступна функція відновлення, яка перенаправляє користувача на сторінку для скидання пароля.

На рисунку 4.3 зображено профіль користувача, який відображає персональну інформацію: ім'я, прізвище, адреса, контактний телефон і місцезнаходження. Також є можливість редагування профілю, додавання додаткової інформації та завантаження аватару.

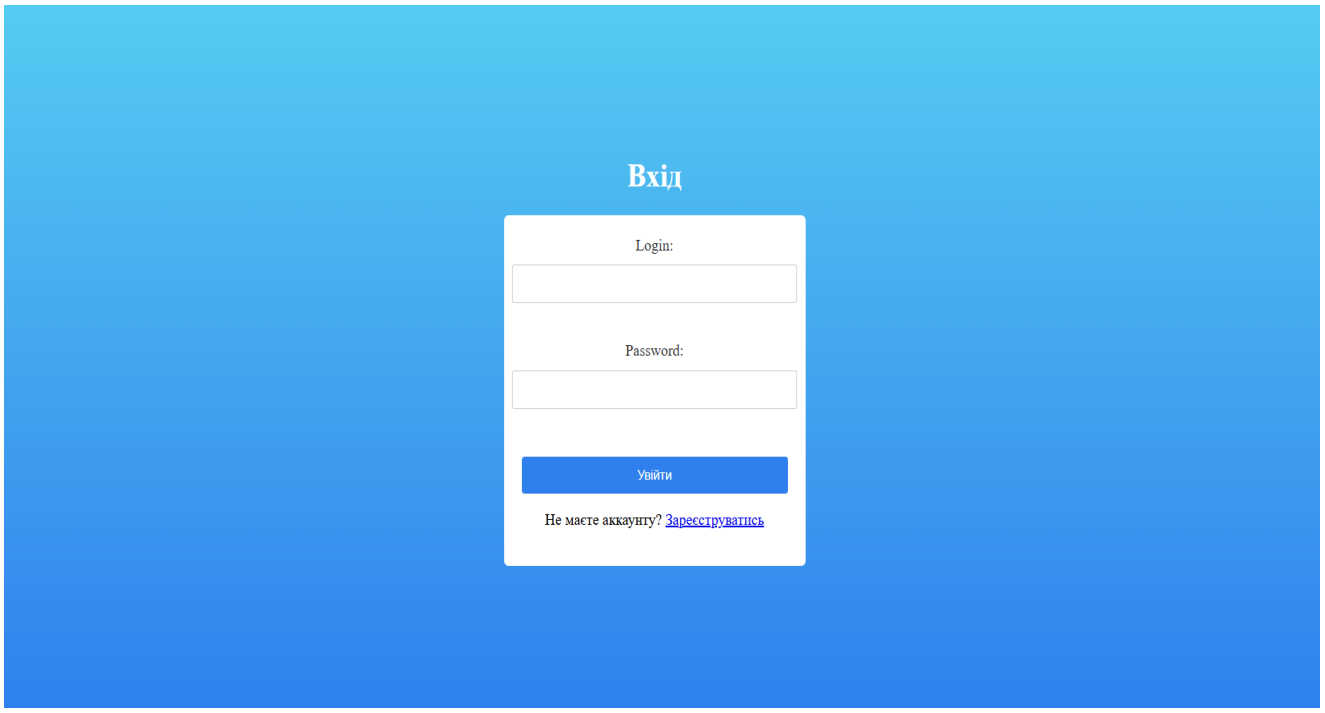

Рисунок 4.2 — Сторінка логіну

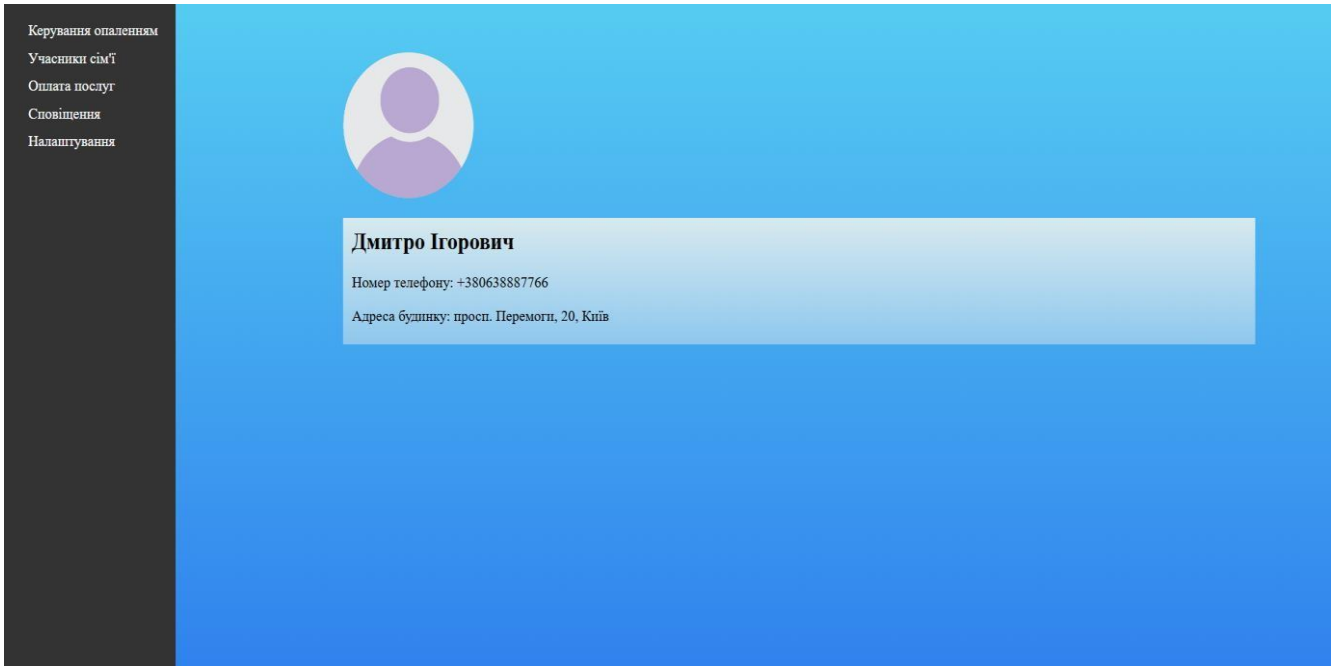

Рисунок 4.3 — Сторінка профілю користувача

На рисунку 4.4 зображено сторінку оплати, де користувач може переглядати список своїх фінансових операцій. У таблиці відображаються дата транзакції, сума та тип послуги. Також доступна функція додавання нового способу оплати або проведення миттєвого платежу.

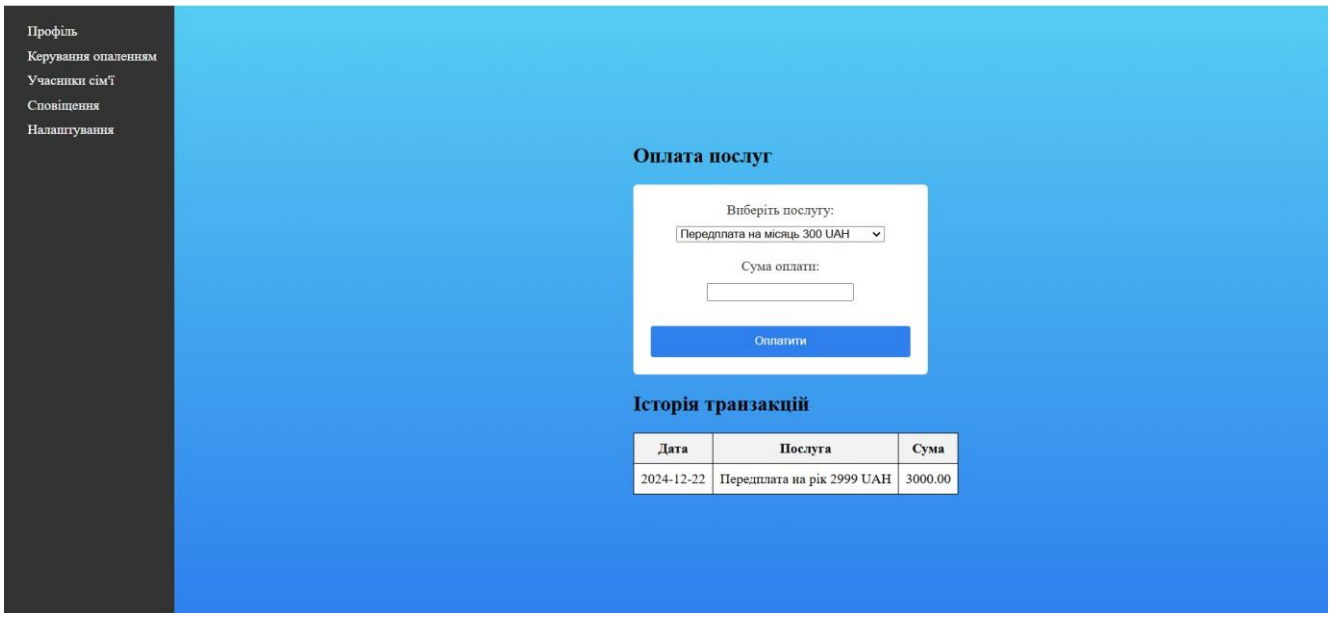

Рисунок 4.4 — Сторінка оплати послуг

На рисунку 4.5 зображено сторінку з членами родини, де користувач може додавати своїх рідних до системи. Також передбачена можливість редагування або видалення записів.

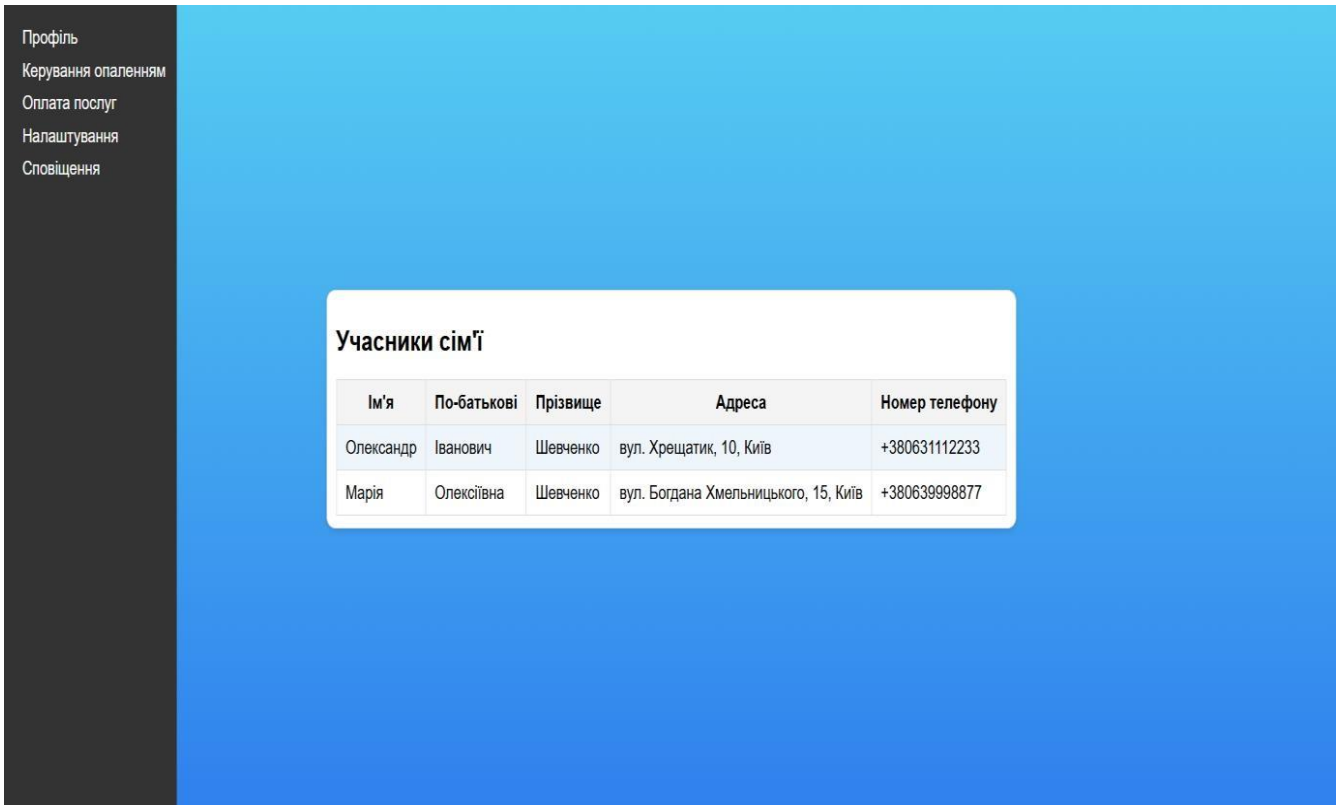

Рисунок 4.5 — Сторінка учасників родини

На рисунку 4.6 зображено сторінку інформації про будинок. Вона містить загальні характеристики будинку, такі як адреса, кількість кімнат і загальна площа. Також показано карту, яка відображає точне розташування будинку.

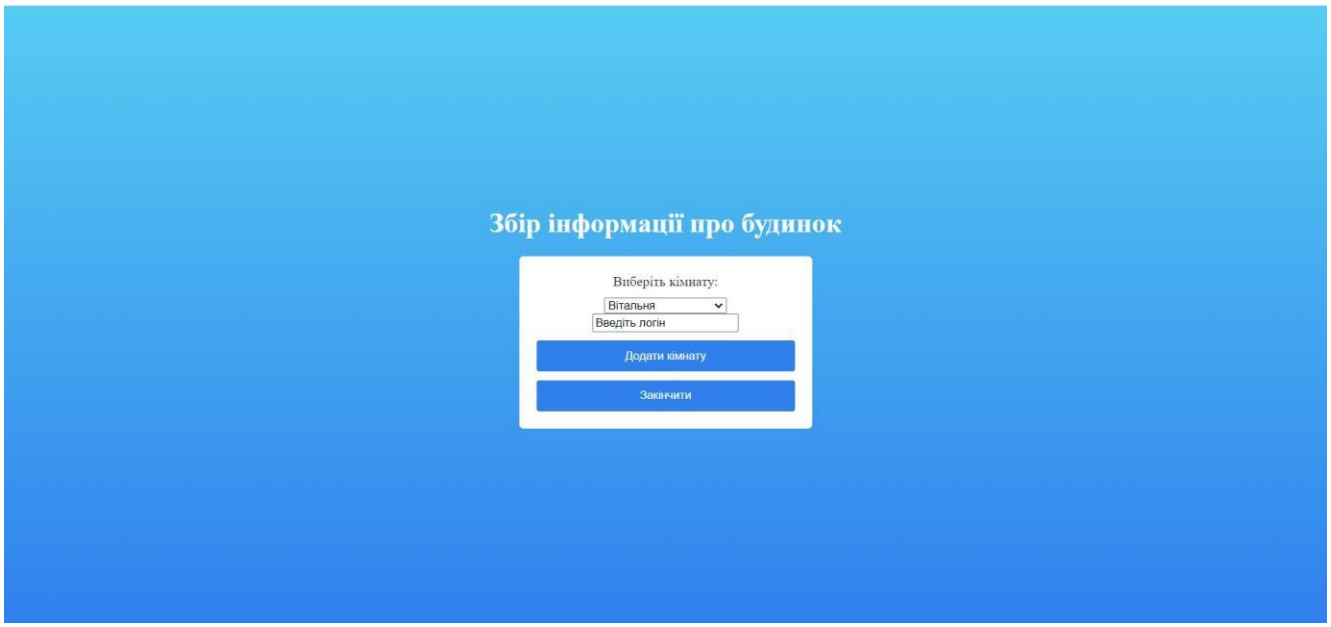

Рисунок 4.6 — Сторінка зі збором інформації про будинок

На рисунку 4.7 зображено сторінку з вибором локації будинку. Користувач може використовувати інтерактивну карту для точного встановлення місця розташування будинку. На карті доступний інструмент для переміщення маркера та автоматичного отримання координат.

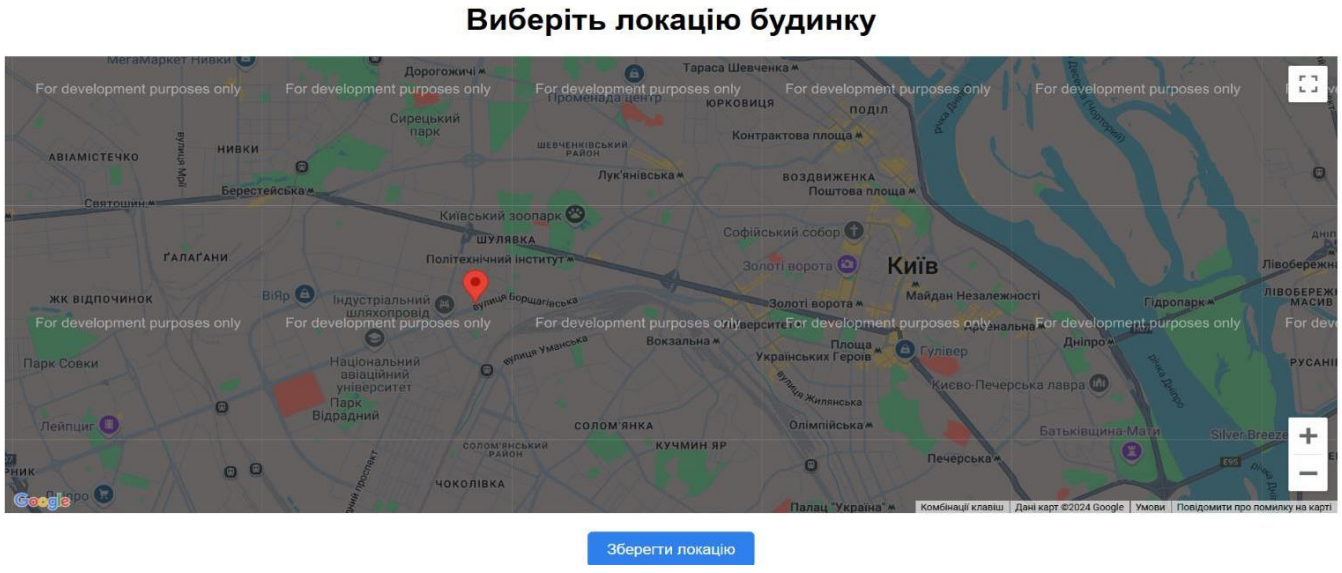

Рисунок 4.7 — Сторінка з вибором локації будинку

На рисунку 4.8 зображено сторінку з керуванням будинком, яка включає панель управління пристроями IoT. Користувач має змогу змінювати налаштування освітлення, опалення, системи безпеки тощо. Статус кожного пристрою (увімкнено/вимкнено) відображається у реальному часі.

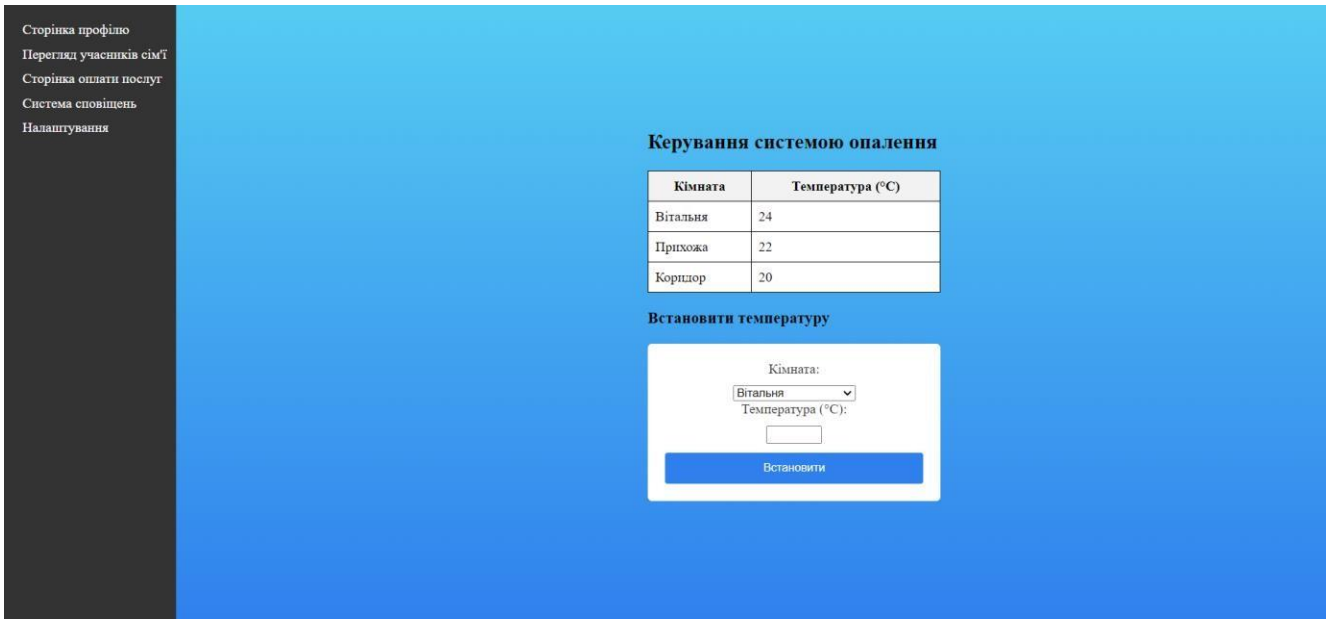

Рисунок 4.8 — Сторінка з керуванням системи опалення

На рисунку 4.9 зображено сторінку з нотифікаціями, яка відображає список сповіщень, отриманих від системи. Кожне повідомлення містить інформацію про стан пристроїв, завершені транзакції або важливі події. Повідомлення впорядковані за датою та часом.

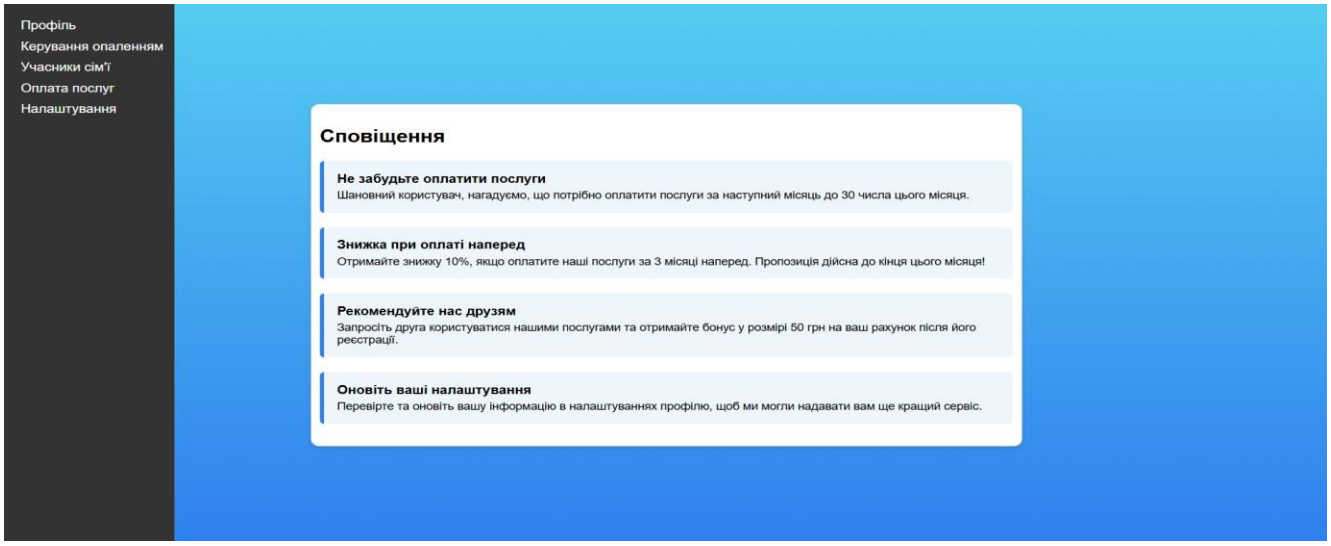

Рисунок 4.9 — Сторінка зі сповіщеннями від системи

На рисунку 4.10 зображено сторінку налаштувань, яка дозволяє користувачеві змінювати загальні параметри системи. Серед доступних функцій — вибір мови інтерфейсу, налаштування сповіщень, увімкнення автоматичних оновлень та зміна параметрів конфіденційності.

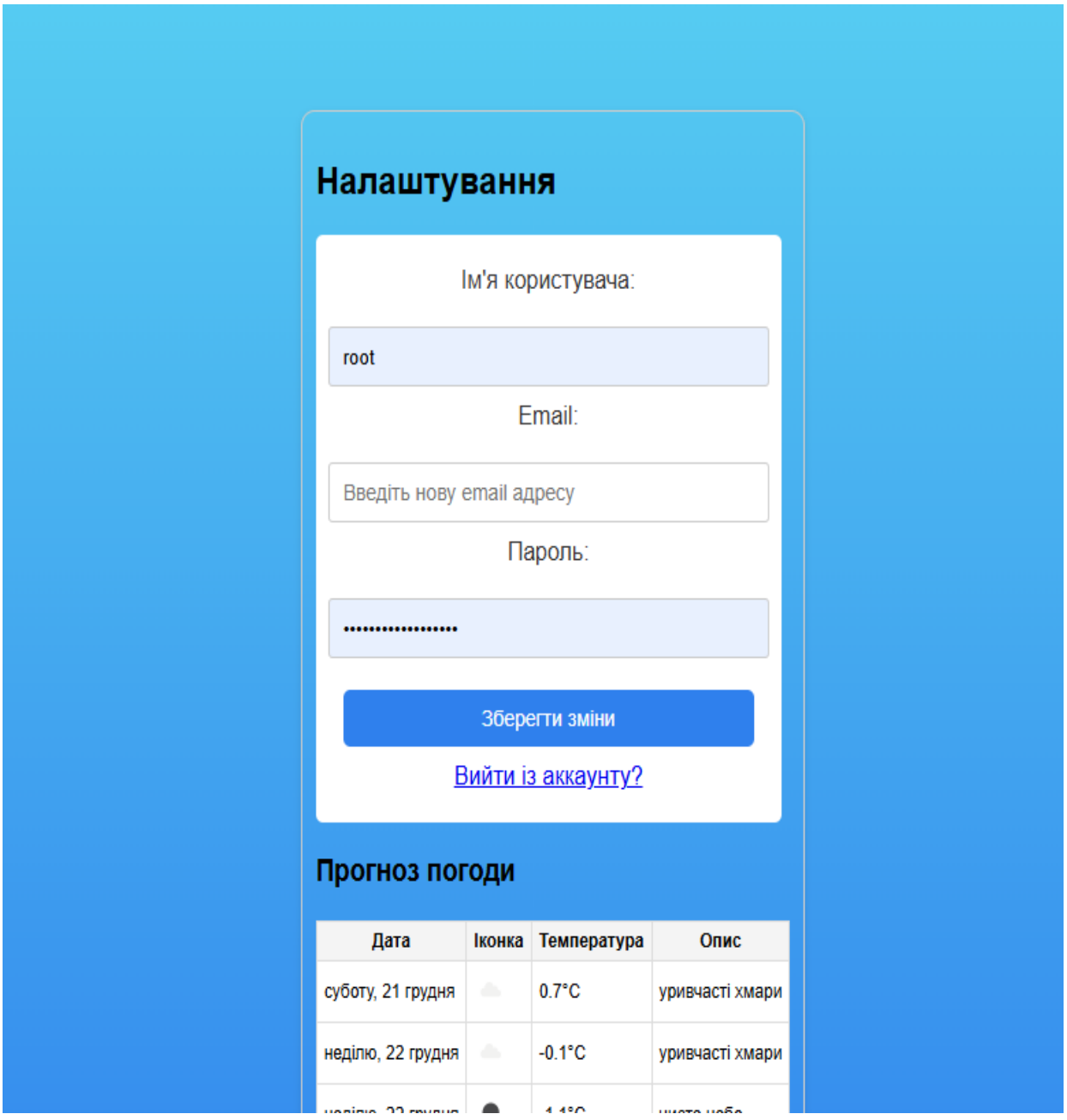

Рисунок 4.10 — Сторінка з налаштуваннями

На рисунку 4.11 зображено приклад пуш-нотифікації, яка надходить на пристрій користувача. Повідомлення може містити інформацію про зміну температури, виявлення руху чи інші важливі події.

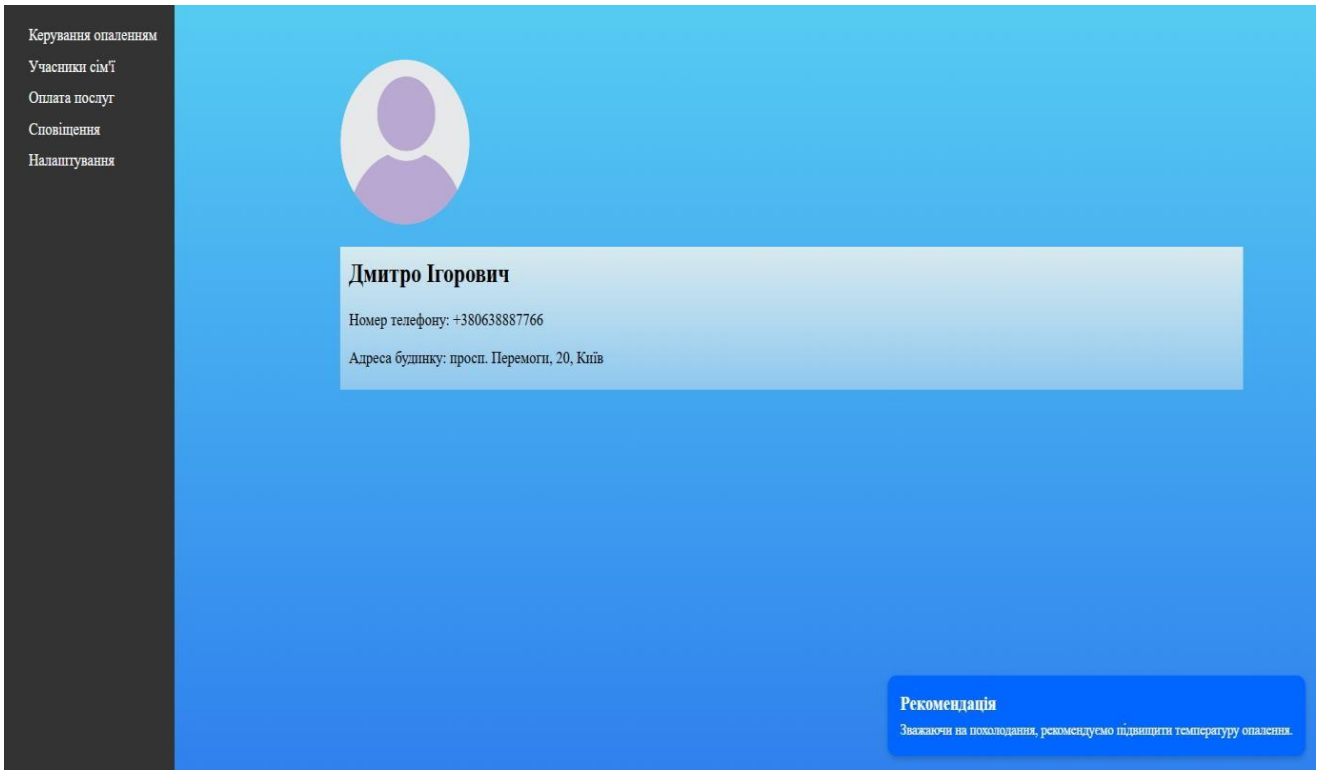

Рисунок 4.11 — Приклад пуш нотифікації

4.2 Огляд аналогів

У сучасному світі розробка систем обробки та аналізу даних з використанням IoT є однією з найбільш актуальних тем. Існує низка відомих аналогів, які використовуються в різних галузях для автоматизації, моніторингу та управління. Серед них можна виділити такі системи:

Google Cloud IoT Core є однією з провідних платформ для роботи з IoT. Вона дозволяє підключати, управляти та обробляти дані з великих мереж пристроїв IoT у хмарі.

Основними функціями є масштабованість, підтримка протоколів MQTT і HTTP, а також інтеграція з іншими сервісами Google Cloud для аналізу даних та машинного навчання.

Недоліки:

— висока залежність від хмарного з’єднання;

— складність інтеграції з локальними системами;

— висока вартість для великих мереж пристроїв.

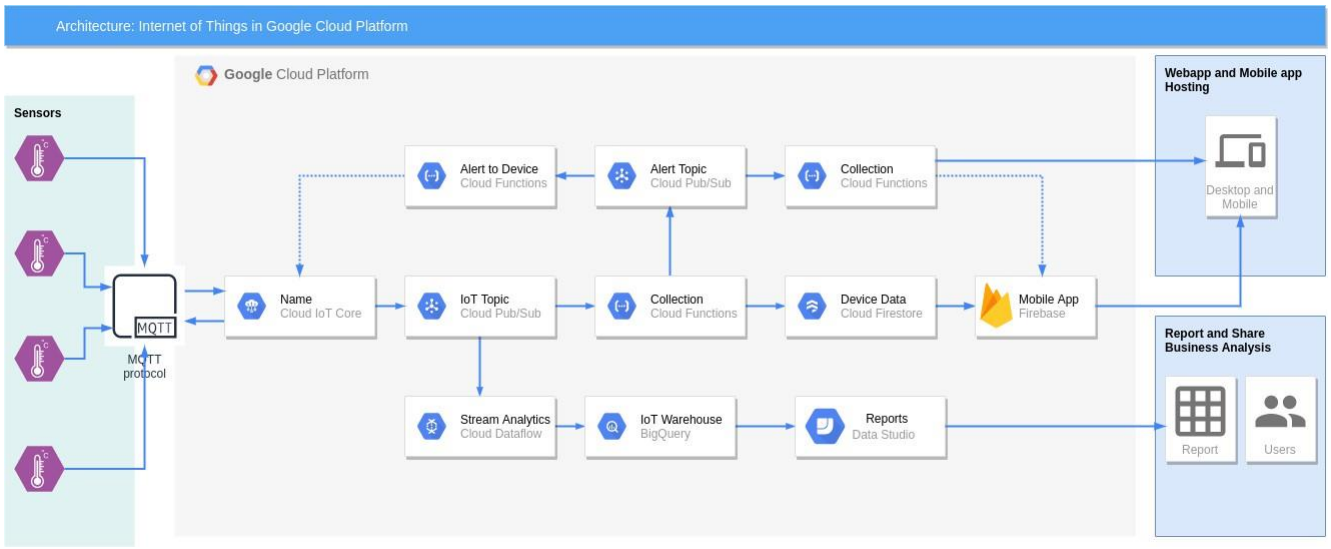

Рисунок 4.12 — Google Cloud IoT Core

Amazon AWS IoT Core забезпечує потужні можливості для управління пристроями IoT та обробки їхніх даних. Платформа дозволяє створювати "розумні" пристрої з підтримкою автоматизації та інтегрувати їх із хмарними сервісами AWS. Її ключові особливості включають функцію аналітики в реальному часі, підтримку голосового управління через Alexa та масштабованість для великих систем.

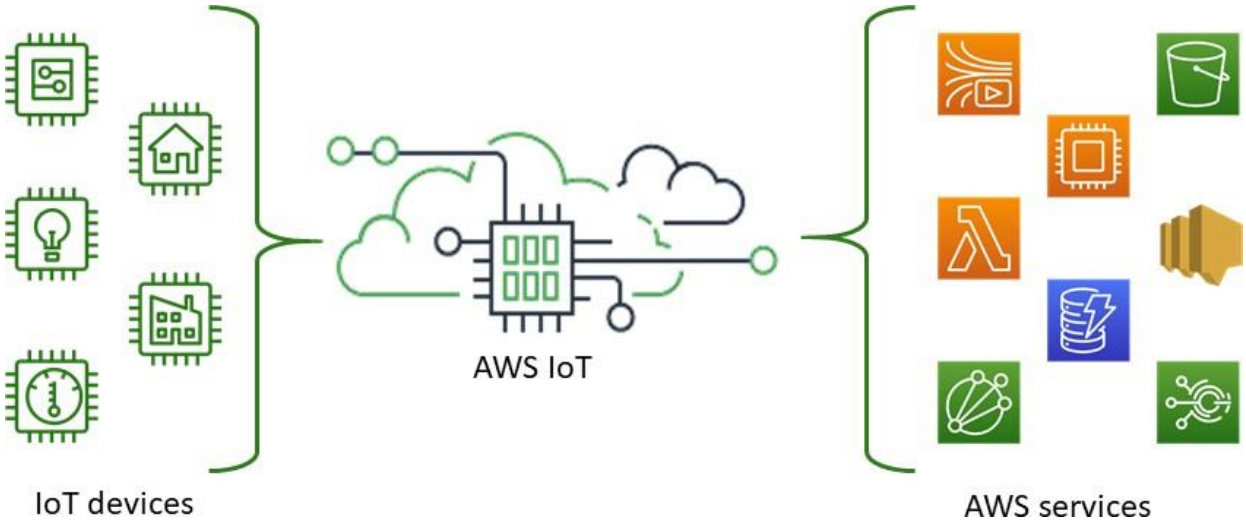

Рисунок 4.13 — Amazon AWS IoT Core

Недоліки:

— складність налаштування для новачків;

— залежність від хмарних обчислень;

— високі витрати на обслуговування та використання.

Azure IoT Hub надає можливості для підключення та моніторингу мільйонів пристроїв IoT. Платформа забезпечує двосторонній зв’язок між пристроями та хмарою, підтримує безпечне підключення та надає аналітичні інструменти для обробки даних. Інтеграція зі штучним інтелектом і машинним навчанням дозволяє створювати прогностичні моделі.

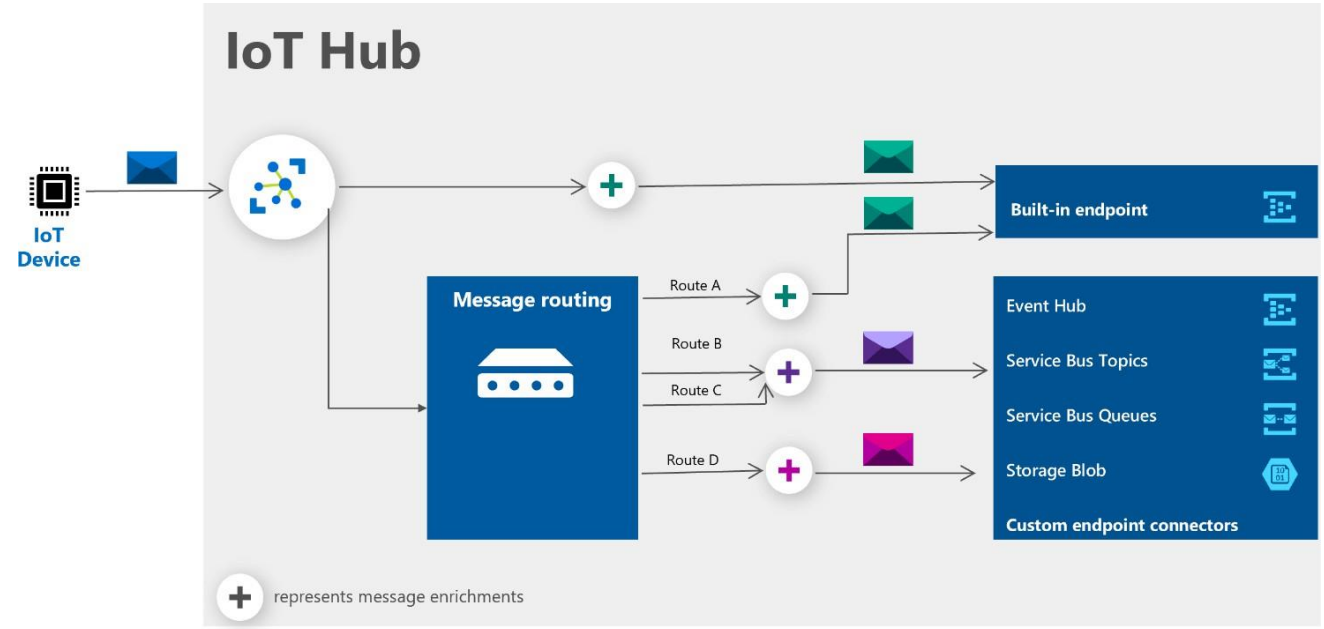

Рисунок 4.13 — Azure IoT Hub

Недоліки:

— високі вимоги до навчання користувачів;

— залежність від хмарних сервісів Microsoft;

— необхідність додаткових інструментів для роботи з локальними мережами.

ThingsBoard є платформою з відкритим вихідним кодом, яка дозволяє створювати IoT-рішення для моніторингу та управління пристроями. Вона забезпечує можливості для збирання даних, візуалізації та автоматизації процесів. ThingsBoard підтримує роботу з різними протоколами, такими як MQTT, CoAP і HTTP.

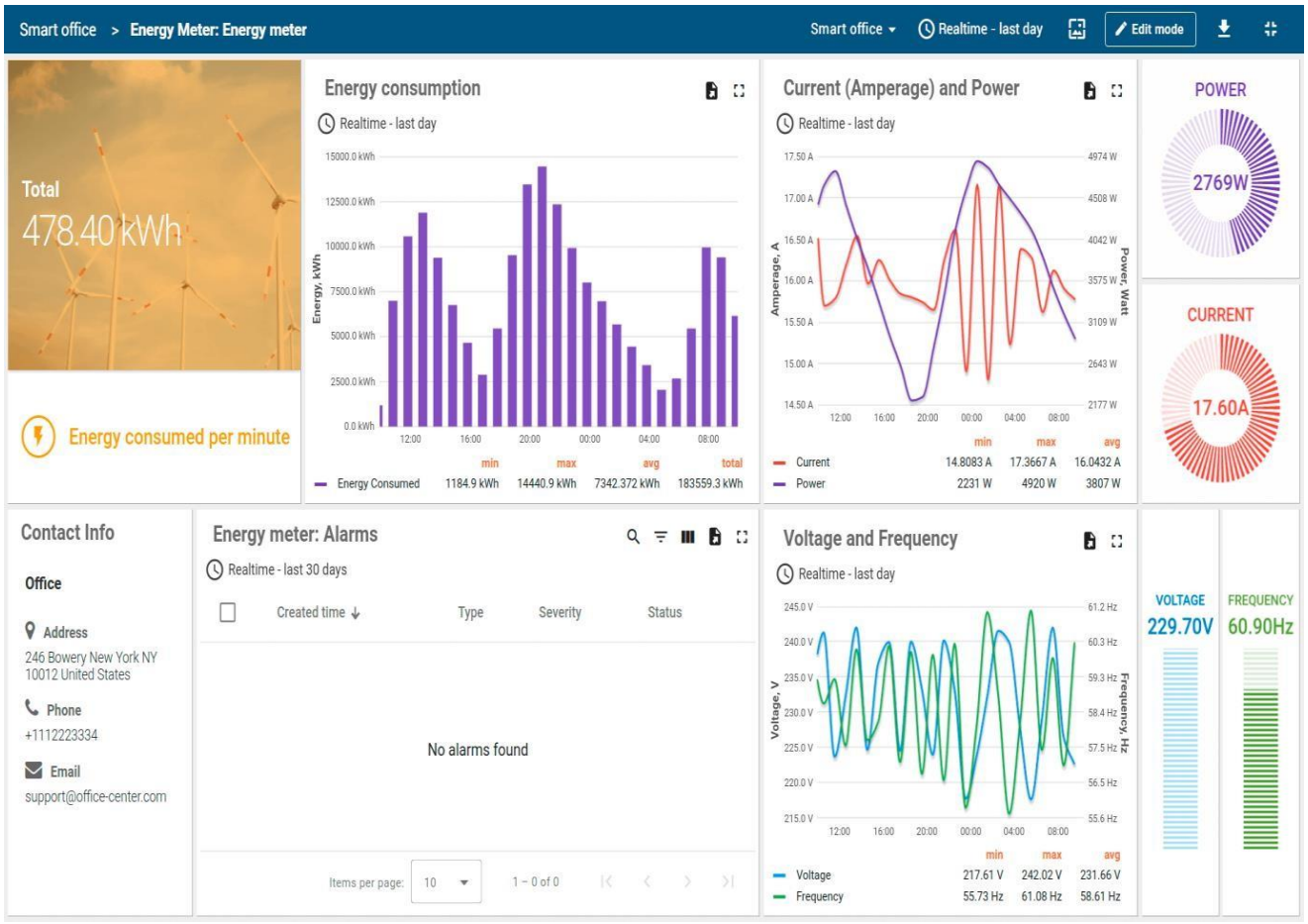

Рисунок 4.14 — ThingsBoard

Недоліки:

— обмежена функціональність у базовій версії;

— відсутність глибокої інтеграції з хмарними сервісами;

— необхідність значних ресурсів для налаштування та підтримки.

MindSphere — це IoT-платформа від Siemens, яка орієнтована на промислові рішення.

Вона забезпечує аналітику в реальному часі, оптимізацію виробничих процесів і підтримку підключення до великої кількості пристроїв. Її основною перевагою є інтеграція з обладнанням Siemens.

Недоліки:

— висока вартість;

— обмежена адаптивність для неіндустріальних рішень;

— залежність від екосистеми Siemens.

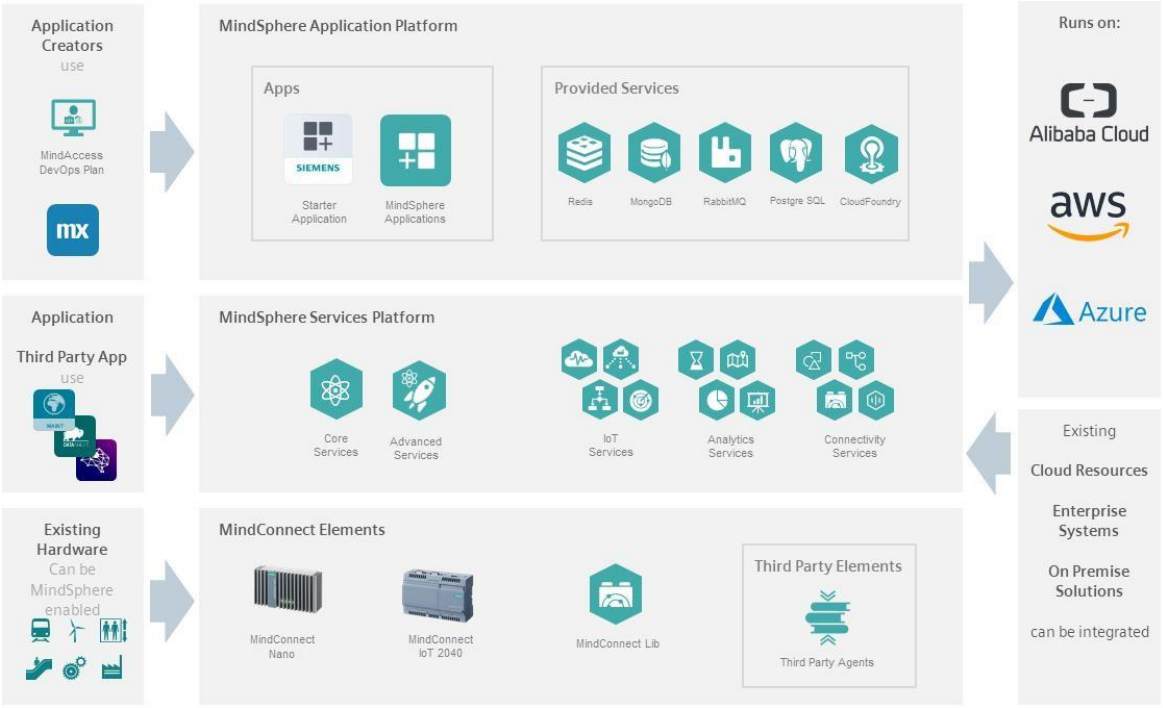

Рисунок 4.15 — MindSphere

Переваги розробленої системи над аналогами:

— гнучкість і локальність; на відміну від хмароорієнтованих платформ, таких як AWS IoT Core чи Azure IoT Hub, розроблена система має можливість локальної обробки даних; це знижує затримки передачі інформації, підвищує продуктивність у реальному часі та забезпечує більшу безпеку, оскільки дані не передаються в зовнішні хмари;

— розроблена система легко інтегрується з зовнішніми сервісами, наприклад, OpenWeatherMap для отримання даних про погоду; це дозволяє адаптувати рішення до змін зовнішнього середовища, чого немає у ThingsBoard або Siemens MindSphere;

— економічна доцільність на відміну від Google Cloud IoT Core та Amazon AWS IoT Core, система є менш витратною, оскільки зосереджена на локальній обробці даних та не потребує постійного підключення до хмари;

— підтримка автоматизації на рівні користувача — система надає зручний інтерфейс для користувачів, включаючи підтримку сценаріїв автоматизації для

домашніх, сільськогосподарських або логістичних рішень; це підвищує її універсальність порівняно з вузькоспеціалізованими платформами, такими як Siemens MindSphere;

— модульність та адаптивність – розроблена система має модульну архітектуру, що дозволяє швидко додавати нові пристрої або функціонал; це є значною перевагою у порівнянні з платформами, які вимагають складної інтеграції, наприклад, Microsoft Azure IoT Hub;

— система розроблена з урахуванням потреб локальних користувачів, включаючи підтримку української мови та адаптацію до умов експлуатації в регіоні;

— на відміну від аналогів, система оптимізована для роботи з енергоефективними пристроями, що дозволяє значно зменшити витрати на обслуговування.

Розроблена комп'ютерна система обробки та аналізу даних з використанням інтернету речей перевершує існуючі аналоги завдяки своїй гнучкості, локальній обробці даних, економічній ефективності та можливостям інтеграції з зовнішніми сервісами. Завдяки універсальності та модульності система може бути адаптована до різних галузей, забезпечуючи високу продуктивність, точність і зручність у використанні.

4.3 Технічні вимоги

Серверна частина системи вимагає процесора із щонайменше чотирма ядрами (наприклад, Intel Core i5), 16 ГБ оперативної пам'яті, SSD-накопичувача об'ємом не менше 512 ГБ і гігабітного мережевого адаптера. IoT-пристрої повинні базуватися на енергоефективних контролерах, таких як ESP32 або Raspberry Pi.

Для програмного забезпечення передбачається використання Python для серверної частини, JavaScript для інтерфейсу користувача та SQL для роботи з базою даних. База даних повинна підтримувати роботу з географічними координатами (тип `POINT`). Використання фреймворків, таких як Django або

Flask, дозволить забезпечити стабільну роботу серверної частини, тоді як React або Vue.js відповідають за інтуїтивний інтерфейс користувача.

У системі передбачена інтеграція з хмарними сервісами, такими як AWS або Google Cloud, для забезпечення зберігання великих обсягів даних і підтримки їх обробки. Водночас система повинна функціонувати локально для забезпечення швидкого реагування та збереження конфіденційності.

Для надійності передбачено резервне копіювання даних та механізми автоматичного відновлення після збоїв.

Тестування системи включає перевірку коректності роботи за різних сценаріїв, зокрема перевантаження мережі або відсутності з’єднання. Також проводиться оцінка ефективності інтеграції з IoT-пристроями, хмарними сервісами та алгоритмами обробки даних.

Система має відповідати стандартам безпеки, енергоефективності та забезпечувати стабільну роботу в реальних умовах експлуатації. Уся документація повинна містити детальні інструкції для розробників і користувачів, опис архітектури, API та основних функціональних модулів.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку комп’ютерної системи для обробки та аналізу даних з використанням інтернету речей.

Особливістю розробки є розширення функціональних можливостей комп'ютерної системи, а саме прогнозування роботи системи опалення та кондиціювання за рахунок застосування нейромережі та зовнішніх погодних сервісів, що дозволяє підвищити швидкодію системи та покращити комфорт користувача.

Аналогом може бути контролер GoogleNestHubMaxChalk – 12 437 грн.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 5.1.

Таблиця 5.1 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненністьконцепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продукту в реальних умовах
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження табл. 5.1

Ринкові переваги					
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження табл. 5.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 5.2

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	3	3
Наявність аналогів на ринку	3	3	4
Цінова політика	4	4	4
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	4	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	3	3
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	4	4	3
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	4	4	4
Супровідна документація	3	4	3
Сума	43	41	42
Середньоарифметична сума балів	(43+41+42) / 3 = 42		

За даними таблиці 5.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 5.3.

Таблиця 5.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 - 10	Низький
11-20	Нижче середнього
21- 30	Середній
31- 40	Вище середнього
41 -48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок розширення функціональних можливостей комп'ютерної системи, а саме прогнозування роботи системи опалення та кондиціювання за рахунок застосування нейромережі та зовнішніх погодних сервісів, що дозволяє підвищити швидкодію системи та покращити комфорт користувача.

5.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t, \quad (5.1)$$

де M — місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ – число робочих днів за місяць, 23 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 5.4.

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

Додаткова заробітна плата розробників, які брати участь в розробці обладнання/програмного продукту.

Таблиця 5.4 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	38500	1673,91	35	58586,957
Програміст	36000	1565,22	35	54782,609
Всього				113369,57

Додаткову заробітну плату прийнято розраховувати як 13,5% від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 13,5\ \% / 100\% \quad (5.2)$$

$$З_д = (113369,57 \cdot 13,5\% / 100\ \%) = 15304,89 \text{ (грн.)}$$

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\% / 100\% \quad (5.3)$$

$$Н_з = (113369,57 + 15304,89) \cdot 22\ \% / 100\% = 28308,38 \text{ (грн.)}$$

Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \quad (5.4)$$

де Ц – балансова вартість обладнання, грн.;

Т — термін корисного використання обладнання згідно податкового законодавства, років

$t_{вик}$ — термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 75999 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання — 1,52 міс.

$$A_{обл} = \frac{75999}{2} \times \frac{1,52}{12} = 4818,78\ грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 5.5.Так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, $B_{нем.ак.} = 9957+6049 = 16006$ грн.

Таблиця 5.5 — Амортизаційні відрахування на матеріальні та нематеріальні ресурси для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія	75999	2	1,52	4818,777
Офісне обладнання (меблі)	42000	4	1,52	1331,522
Приміщення	1500000	20	1,52	9510,870
Всього				15661,17

Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії

для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (5.5)$$

де В – вартість 1 кВт-години електроенергіїдля 1 класу підприємства з ПДВ в 2024 році для Вінницької області за даними Енера-Вінниця, В = 10,42 грн./кВт;

П – встановлена потужність обладнання, кВт. П = 0,3кВт;

Ф – фактична кількість годин роботи обладнання, годин.

$К_п$– коефіцієнт використання потужності, $К_п$ = 0,9.

$В_е = 0,9 \cdot 0,3 \cdot 8 \cdot 35 \cdot 10,42 = 787,752$ (грн.)

Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.6)$$

Де $Н_{ів}$ — норма нарахування за статтею «Інші витрати».

$I_в$ = 113369,57 * 78% / 100% = 88428,26 (грн.)

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та

рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$H_{нзв} = (З_о + З_р) \cdot \frac{H_{нзв}}{100\%}, \quad (5.7)$$

Де $H_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$H_{нзв}$ = 113369,57 * 145% / 100 % = 164386 (грн.)

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$B_{заг}$ =113369,57+15304,89+28308,38+15661,17+16006+787,75+88428,26+
+164386 = 442251,89грн.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{B_{заг}}{\eta} \text{ (грн)}, \quad (5.8)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η = 0,1; технічного проектування, то η = 0,2; розробки конструкторської документації, то η = 0,3; розробки технологій, то η = 0,4; розробки дослідного зразка, то η = 0,5; розробки промислового зразка, то η = 0,7; впровадження, то η = 0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$$ЗВ = 442251,89 / 0,5 = 884504 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

— вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

— зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

— кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

— визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об'єднати, враховуючи визначені ситуації з відповідними умовами.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.9)$$

Де $\pm \Delta Ц_o$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N — кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_o = Ц_б \pm \Delta Ц_o$;

$Цб$ — вартість програмного продукту у році до впровадження результатів розробки;

ΔN — збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

p — коефіцієнт, який враховує рентабельність продукту;

ϑ — ставка податку на прибуток, у 2024 році $\vartheta = 18\%$.

Припустимо, що при прогнозованій ціні 7200 грн. за одиницю, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на *300* грн. Кількість одиниць

реалізованої продукції також збільшиться: протягом першого року – на 5000шт., протягом другого року –на 4000 шт., протягом третього року на 3000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П_1 = (0*300 + (7200 + 300)*5000)* 0{,}8333* 0{,}23) * (1 - 0{,}18) = 5657999{,}774 \text{ грн.}$$

$$\Delta П_2 = (0*300 + (7200 + 300)*(5000+4000)* 0{,}8333* 0{,}23) * (1 - 0{,}18) = 10608749{,}576 \text{ грн.}$$

$$\Delta П_3 = (0*300 + (7200 + 300)*(5000+4000+3000)* 0{,}8333* 0{,}23) * (1 - 0{,}18) = 14144999{,}434 \text{ грн.}$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 30411748,78грн.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ –ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}05 \ldots 0{,}15$;

t – період часу (в роках).

Збільшення прибутку ми отримаємо, починаючи з першого року:

$$ПП = (5657999{,}774/(1+0{,}1)^1)+(10608749{,}576/(1+0{,}1)^2)+(14144999{,}434/$$

$/(1+0,1)^3)$= 5143636,16 + 8767561,633 + 10627347,43 = 24538545,22грн.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (5.11)$$

де $k_{інв}$– коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв} = 2...5$, але може бути і більшим;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$PV = 2 * 884504 = 1769007,55$ *грн.*

Тоді абсолютний економічний ефект $E_{абс}$або чистий приведений дохід *(NPV, NetPresentValue)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП – PV, \quad (5.12)$$

$$E_{абс} = 24538545,22 – 1769007,55 = 22769537,67 \text{ грн.}$$

Оскільки $E_{абс} > 0$то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої но-рми дохідності

(IRR, InternalRateofReturn) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.13}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{1 + 22769537,67/1769007,55} - 1 = 1,403$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{5.14}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні *d*= (0,09…0,14);

f показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{min} = 0,14 + 0,05 = 0,19.$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \tag{5.15}$$

$T_{ок}$ = 1 / 1,403 = 0,71 р.

Оскільки $T_{ок}$ < 3-х років,а саме термін окупності рівний 0,71роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 884504гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розрахованоперіод окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є високо конкурентоспроможним. Період окупності складе близько 0,71роки.

ВИСНОВКИ

Під час виконання магістерської роботи було спроектовано архітекту комп'ютерної системи для обробки та аналізу даних з використанням технологій Інтернету речей (IoT) та вдосконалено метод обробки даних шляхом інтеграції штучного інтелекту, регресійних моделей та зовнішніх погодних сервісів. Модель охоплює всі рівні системи: пристроїв, мережі та обробки даних. Особливу увагу приділено інтеграції сенсорів, збиранню даних у реальному часі та забезпеченню їх передачі через стандартні мережеві протоколи, такі як Wi-Fi та Ethernet. Модель є базовою основою для подальшої реалізації масштабованої та функціональної IoT-системи.

Визначено та протестовано ефективні алгоритми для обробки великих масивів даних у реальному часі. Основну увагу було приділено алгоритмам машинного навчання, що дозволяють виконувати аналіз даних, виявлення закономірностей та прогнозування. Також проведено оптимізацію методів попередньої обробки даних для прискорення роботи системи. Запропоновані алгоритми забезпечують швидку обробку інформації навіть за умов значного навантаження, що важливо для реального використання в динамічних середовищах.

Досліджено можливості інтеграції IoT-систем з хмарними технологіями для забезпечення масштабованості. Обрано та протестовано сервіси, які дозволяють зберігати та аналізувати дані у хмарних середовищах. Особливу увагу приділено підвищенню безпеки передачі даних між IoT-пристроями та хмарними платформами. Інтеграція хмарних технологій дозволила значно збільшити продуктивність системи, знизити витрати на локальне обладнання та забезпечити доступ до даних з будь-якої точки світу.

Проведено оцінку практичної ефективності розробленої системи у різних галузях, зокрема в логістиці, медицині та концепції «розумного міста». У логістиці система дозволяє автоматизувати моніторинг умов перевезення товарів (температура, вологість, місцезнаходження), що сприяє зниженню втрат і підвищенню ефективності доставки. У медицині розроблена система може

застосовуватися для моніторингу стану пацієнтів у режимі реального часу, що підвищує швидкість реагування на критичні ситуації. У «розумному місті» IoT-рішення забезпечують ефективне управління енергоспоживанням, транспортною інфраструктурою та екологічним моніторингом.

Таким чином, усі поставлені завдання виконано, а поставлена мета роботи щодо розробки комп’ютерної системи для обробки та аналізу даних з використанням IoT досягнута. Розроблена концептуальна модель та алгоритми демонструють високу ефективність і надійність, а запропоновані методи інтеграції хмарних технологій та практичні сценарії підтверджують перспективність впровадження таких систем у різних галузях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Сліденко Д.О. Система для обробки та аналізу даних з використанням інтернету речей / Д.О. Сліденко, О. С. Городецька, О. В. Войцеховська // Тези доповіді. LI науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії. Вінниця 2023 р. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/view/17561/15656.

2. Комп’ютерна система обробки та аналізу даних з використанням інтернету речей / Д. О. Сліденко, О. С. Городецька, Д. В. Куклій // Матеріали Молодь в науці: дослідження, проблеми, перспективи (МН-2025), 2025 р. [Електронний ресурс] — https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/22878/18887

3. Create System Model of Solution [Електронний ресурс]. Режим доступу: https://docs.idew.org/internet-of-things-project/iot-project-outline/3-3-create-system-model-of-solution

4. Технології інтернету речей. [Електронний ресурс].Режим доступу: https://ela.kpi.ua/bitstream/123456789/42078/1/Zhurakovskyi_B_Zeniv_Tehnologii_internet_rechey.pdf

5. Інтернет речей [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82_%D1%80%D0%B5%D1%87%D0%B5%D0%B9

6. About Remote Access Service [Електронний ресурс]. Режим доступу: https://learn.microsoft.com/en-us/windows/win32/rras/about-remote-access-service

7. Переваги використання веб-застосунків [Електронний ресурс]. Режим доступу: https://business.diia.gov.ua/en/cases/tehnologii/so-take-hmarni-tehnologii-i-ak-voni-mozut-dopomogti-vasomu-pidpriemstvu

8. Google Nest [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Google_Nest

9. Ecobee [Електронний ресурс]. Режим доступу: https://www.achrnews.com/articles/144928-ecobee-announces-new-smart-thermostat-integration-with-alarmcom-based-security-and-smart-home-solutions

10. Using the ecobee Web Portal [Електронний ресурс]. Режим доступу: https://support.ecobee.com/s/articles/Using-the-ecobee-Web-Portal

11. Google Nest Hub 2nd Gen - Smart Home Device with Google Assistant - Chalk [Електронний ресурс]. Режим доступу: https://www.thesource.ca/en-ca/smart-home-household/smart-home/voice-assistants-smart-speakers/google-nest-hub-2nd-gen---smart-home-device-with-google-assistant---chalk/p/108093365

12. Difference Between Ecobee and Nest [Електронний ресурс]. Режим доступу: http://www.differencebetween.net/technology/difference-between-ecobee-and-nest/

13. Архітектура програмного забезпечення [Електронний ресурс]. Режим доступу: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya

14. Розробка клієнт-серверного застосунку на C++ [Електронний ресурс]. Режим доступу: https://ekmair.ukma.edu.ua/server/api/core/bitstreams/148bf8ee-33fd-49fb-a24d-a45e94b8603b/content

15. SOAP vs REST. What's the Difference? [Електронний ресурс]. Режим доступу: https://appmaster.io/blog/soap-vs-rest

16. Types of Databases [Електронний ресурс]. Режим доступу: https://www.javatpoint.com/types-of-databases

17. HTML: Мова для створення веб-сторінок [Електронний ресурс]. Режим доступу: https://w3schoolsua.github.io/index.html#gsc.tab=0

18. Future Of Web Technologies And Programming [Електронний ресурс]. Режим доступу: https://www.sencha.com/blog/future-of-web-technologies-and-programming/

19. What is XAMPP? and How to Install XAMPP? [Електронний ресурс]. Режим доступу: https://www.devopsschool.com/blog/what-is-xampp-and-how-to-install-xampp/

20. Google Chrome [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Google_Chrome

ДОДАТОК А

Технічне завдання

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. О.Д. Азаров
"18" вересня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

«Комп'ютерна система для обробки та аналізу даних з використанням інтернету речей»

08–54.МКР.017.00.000 ПЗ

Науковий керівник к.т.н., доц., каф. ОТ
Городецька О.С.
Магістрант групи 1КІ–23м
Сліденко Д.О.

Вінниця 2024

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

Підставою для розробки даної магістерської кваліфікаційної роботи є наказ ВНТУ №_310_ від «17» 09 2024 року та рішення засідання кафедри обчислювальної техніки (протокол №___від «_____»___2024 року).

2 Мета і призначення МКР

2.1 Мета роботи — аналіз сучасного стану і тенденцій розвитку зарубіжних і вітчизняних розробок у сфері обробки та аналізу даних із використанням технологій інтернету речей (IoT). Також метою є створення концептуальної моделі комп'ютерної системи, яка дозволяє інтегрувати сенсорні мережі, алгоритми штучного інтелекту та зовнішні сервіси для ефективного управління і прогнозування на основі зібраних даних.

2.2 Призначення розробки — комп'ютерна система для обробки та аналізу даних із використанням IoT призначена для автоматичного збору даних із сенсорів, інтеграції з хмарними сервісами, прогнозування на основі штучного інтелекту та генерування рекомендацій. Система забезпечує можливості автоматизації, моніторингу, адаптивного керування пристроями та підтримки рішень у таких галузях, як логістика, сільське господарство, «розумний дім» і медицина.

3 Вихідні дані для виконання МКР

3.1 Розробка концептуальної моделі комп'ютерної системи для обробки та аналізу даних з використанням IoT;

3.2 Визначення ефективних алгоритмів для обробки великих масивів даних у реальному часі;

3.3 Дослідження можливостей інтеграції IoT з хмарними технологіями для масштабованості систем;

3.4 Оцінка практичної ефективності впровадження таких систем у конкретних галузях (наприклад, логістика, медицина, «розумне місто»).

4 Вимоги до виконання МКР

Основною вимогою є використання методів обробки даних із застосуванням

технологій інтернету речей (IoT) для автоматичного збору, аналізу та обробки інформації в режимі реального часу. Система повинна забезпечувати інтеграцію сенсорних даних із центральним контролером для виконання завдань моніторингу, прогнозування та автоматичного керування пристроями.

Розроблені алгоритми мають враховувати динамічні зміни параметрів середовища, адаптуватися до змінних умов роботи та забезпечувати високу точність аналізу даних.

Усі компоненти системи повинні відповідати вимогам енергоефективності, стабільності роботи та безперебійної взаємодії між пристроями, що входять до IoT-інфраструктури. Система повинна бути масштабованою та придатною для використання в різних галузях, таких як «розумне місто», логістика чи медицина.

5. Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в таблиці А.1.

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Огляд і аналіз джерел інформації	20.09.2024	04.10.2024	Розділ 1
2	Теоретичні дослідження технологій побудови системи та вибір засобів реалізації	05.10.2024	18.10.2024	Розділ 2
3	Розробка комп'ютерної системи для обробки та аналізу даних з використанням інтернету речей	19.10.2024	01.11.2024	Розділ 3
4	Розробка і тестування комп'ютерної системи	02.11.2024	16.11.2024	Розділ 4
5	Економічна частина	17.11.2024	25.11.2024	Розділ 5

6. Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами, нормоконтроль про відповідність оформлення МКР діючим вимогам.

7. Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора

8. Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлювання МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104—2006 «Єдина система конструкторської документації. Основні написи»;

— Методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 «Комп’ютерна інженерія» (освітня програма «Комп’ютерна інженерія»). Кафедра обчислювальної техніки ВНТУ 2022;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ—03.02.02—П.001.01:21».

ДОДАТОК Б

UML–діаграми роботи системи

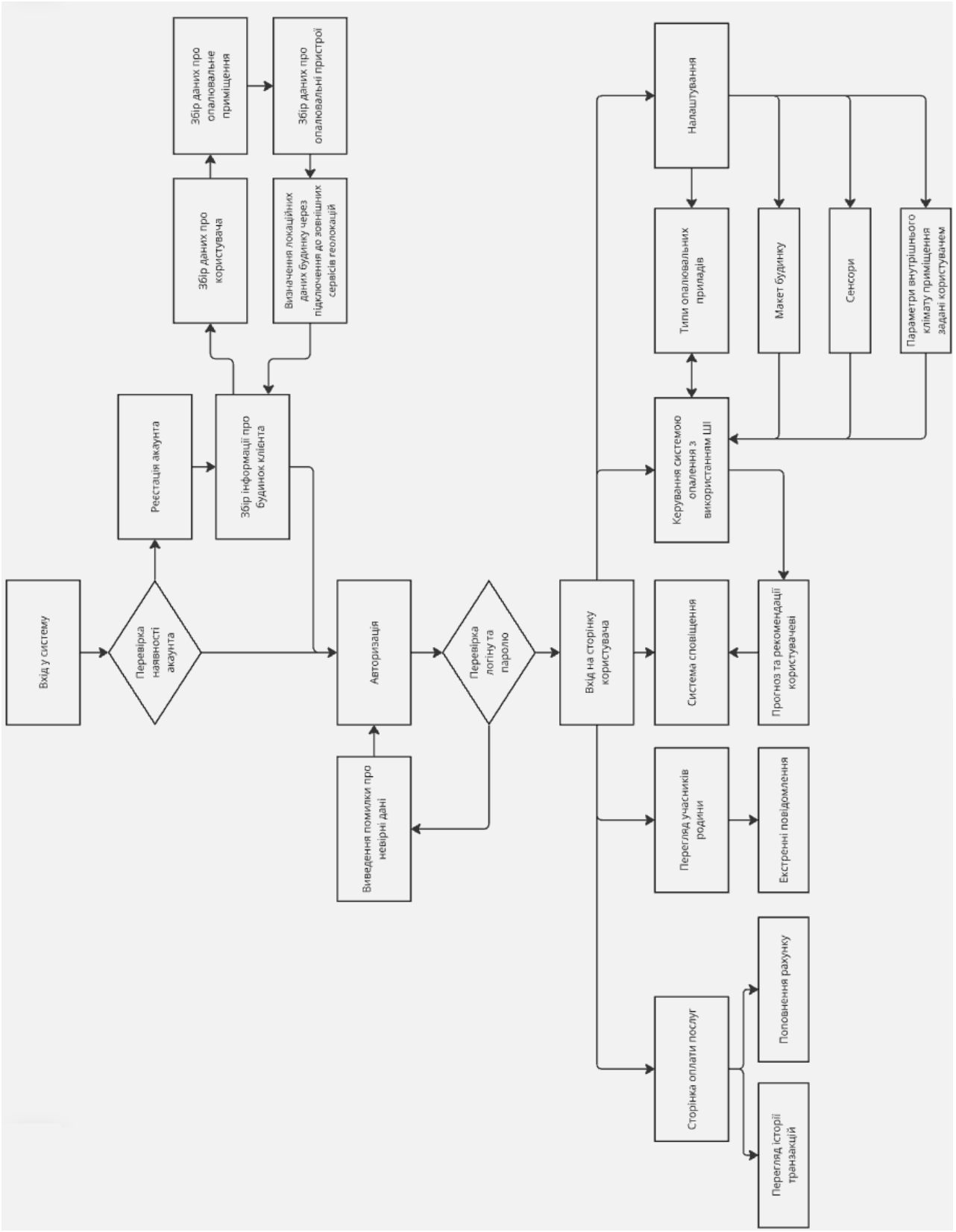

Рисунок Б.1 — UML–діаграми роботи системи

ДОДАТОК В

Структура рівнів реалізованої системи

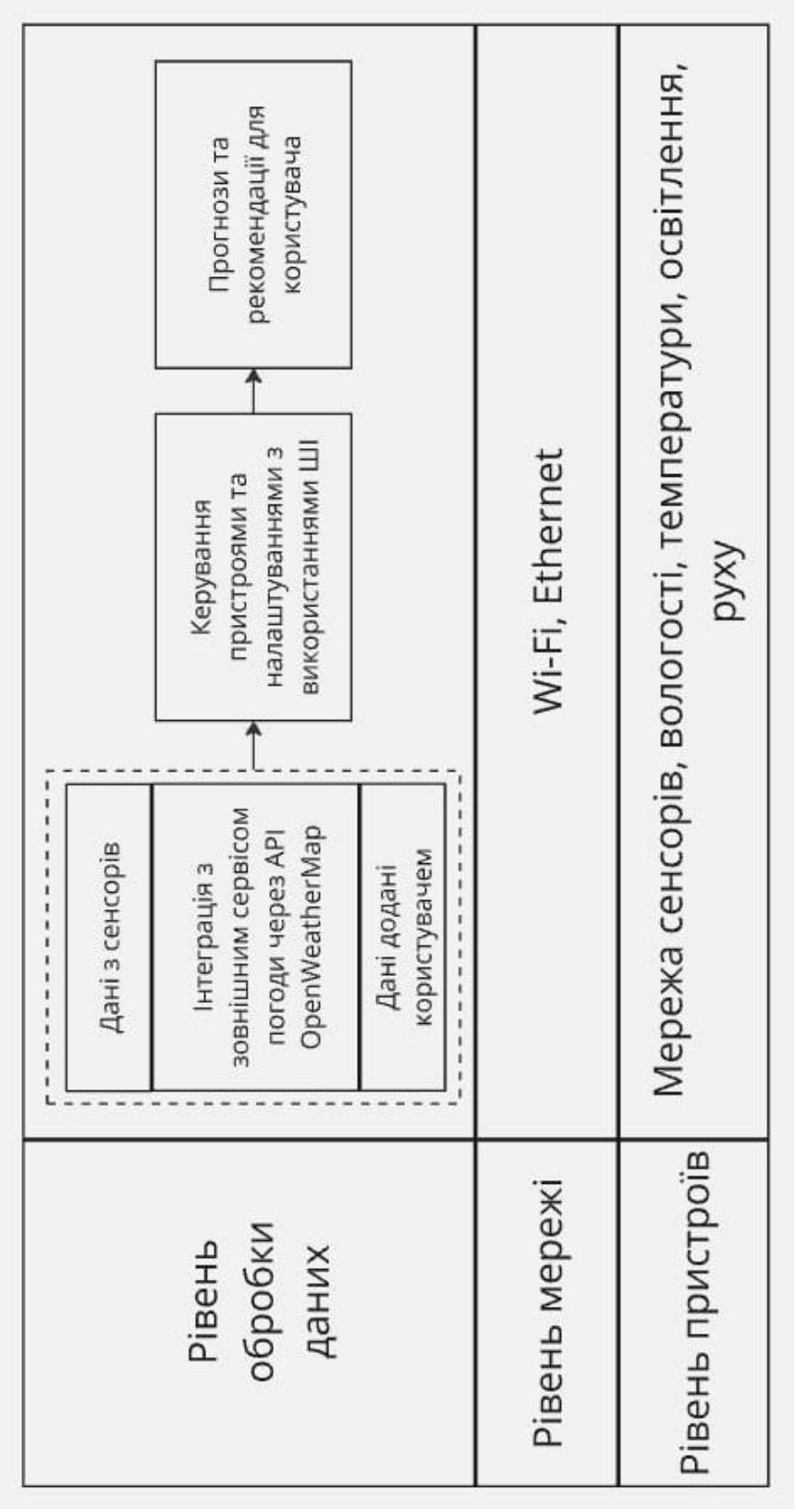

Рисунок В.1 — Ілюстрація рівнів структури реалізованої системи

ДОДАТОК Г

Лістинг модулю зі штучним інтелектом

```
import json
from sklearn.ensemble import RandomForestClassifier
import numpy as np
with open('weather_data.json', 'r') as f:
    weather_data = json.load(f)
temperature = weather_data['temperature']
humidity = weather_data['humidity']
wind_speed = weather_data['wind_speed']
X = np.array([
    [10, 80, 10],  # 10°C, 80% вологості, 10 км/год
    [20, 70, 5],   # 20°C, 70% вологості, 5 км/год
    [15, 75, 12],  # 15°C, 75% вологості, 12 км/год
    [5, 90, 15]    # 5°C, 90% вологості, 15 км/год
])
# Цільова змінна: 1 - увімкнути опалення, 0 - не потрібно
y = np.array([1, 0, 0, 1])
model = RandomForestClassifier()
model.fit(X, y)
new_input = np.array([[temperature, humidity, wind_speed]])  # Нові дані
prediction = model.predict(new_input)
# Зберігаємо результат в файл
result = "Увімкнути опалення" if prediction == 1 else "Не потрібно вмикати опалення"
with open('prediction_result.txt', 'w') as f:
```

```
            f.write(result)
```

ДОДАТОК Д

Лістинг модулю з вибором локації та відображенням мапи

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Вибір локації</title>
  <style>
    #map {
      height: 500px;
      width: 100%;
    }
    body {
      font-family: Arial, sans-serif;
      text-align: center;
    }
    .button-container {
      margin-top: 20px;
    }
    button {
      padding: 10px 20px;
      font-size: 16px;
      background-color: #2F80ED;
      color: white;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }
  </style>
</head>
```

```
<body>

<h1>Виберіть локацію будинку</h1>

<div id="map"></div>

<div class="button-container">
  <button id="save-location">Зберегти локацію</button>
</div>
<script>
  let map;
  let marker;
  const userId = new URLSearchParams(window.location.search).get('user_id');
  function initMap() {
    const ukraine = { lat: 48.3794, lng: 31.1656 };
    map = new google.maps.Map(document.getElementById('map'), {
      center: ukraine,
      zoom: 6,
      restriction: {
        latLngBounds: {
          north: 52.0,
          south: 44.0,
          west: 22.0,
          east: 40.0
        },
        strictBounds: true
      }
    });
    map.addListener('click', (e) => {
      if (marker) {
        marker.setMap(null);
      }
```

```
            marker = new google.maps.Marker({
                position: e.latLng,
                map: map
            });
        });
    }
    document.getElementById('save-location').addEventListener('click', () => {
        if (!marker) {
            alert('Будь ласка, виберіть локацію на карті.');
            return;
        }
        const position = marker.getPosition();
        const lat = position.lat();
        const lng = position.lng();

        fetch('php/save_location.php', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ user_id: userId, latitude: lat, longitude: lng })
        })
        .then(response => response.text())
        .then(data => alert(data))
        .catch(error => console.error('Помилка:', error));
    });
</script>
<script async defer
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAD2aVX9f6vHrLZOd05JVgVPC60fMRLPt8&callback
=initMap"></script>

</body>
</html>
```

ДОДАТОК Е

Лістинг сторінки оплати

```
<!DOCTYPE html>
<html>
<head>
  <title>Оплата послуг</title>
  <link rel="stylesheet" type="text/css" href="css/styles.css">
  <style>
    table {
      border-collapse: collapse;
      width: 100%;
      background-color: white;
    }
    th, td {
      border: 1px solid black;
      padding: 8px;
    }
    th {
      background-color: #f2f2f2;
    }
  </style>
</head>
<body>
  <div class="navbar">
    <ul>
      <li><a href="profile.html">Профіль</a></li>
      <li><a href="temp_system.php">Керування опаленням</a></li>
      <li><a href="family_members.php">Учасники сім'ї</a></li>
      <li><a href="notifications.html">Сповіщення</a></li>
      <li><a href="settings.html">Налаштування</a></li>
    </ul>
  </div>
```

```
<div class="content">
    <h2>Оплата послуг</h2>
    <form action="process_payment.php" method="post">
        <label for="service">Виберіть послугу:</label>
        <select name="service" id="service">
            <option value="service1">Передплата на місяць 300 UAH</option>
            <option value="service2">Передплата на 6 місяців 1599 UAH</option>
            <option value="service3">Передплата на рік 2999 UAH</option>
        </select>
        <br>
        <label for="amount">Сума оплати:</label>
        <input type="number" name="amount" id="amount" min="0" step="0.01" required>
        <br>
        <input type="submit" value="Оплатити">
    </form>

    <h2>Історія транзакцій</h2>
    <table>
        <thead>
            <tr>
                <th>Дата</th>
                <th>Послуга</th>
                <th>Сума</th>
            </tr>
        </thead>
        <tbody>
            <?php
            $servername = "localhost";
            $username = "root";
            $password = "";
            $dbname = "diplom";
```

```
            $conn = new mysqli($servername, $username, $password, $dbname);

            if ($conn->connect_error) {
               die("Connection failed: " . $conn->connect_error);
            }
                  $sql = "SELECT date, service, amount FROM transactions";
                  $result = $conn->query($sql);
                  if ($result->num_rows > 0) {
               while ($row = $result->fetch_assoc()) {
                  echo "<tr>";
                  echo "<td>" . $row["date"] . "</td>";
                  echo "<td>" . $row["service"] . "</td>";
                  echo "<td>" . $row["amount"] . "</td>";
                  echo "</tr>";
               }
            } else {
               echo "<tr><td colspan='3'>Поки що немає жодної транзакції.</td></tr>";
            }
            $conn->close();
            ?>
         </tbody>
      </table>

   </div>
</body>
</html>
```

ДОДАТОК Ж

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Комп'ютерна система для обробки та аналізу даних з використанням інтернету речей

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Turnitin

Оригінальність 90% Схожість 10%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Turnitin щодо роботи.

Автор роботи _____ Сліденко Д.О.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Городецька О.С.
(підпис) (прізвище, ініціали)