Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

**Комплексна бакалаврська дипломна робота на тему:**

# «ІОТ-СИСТЕМА МОНІТОРИНГУ ПОКАЗНИКІВ СТАНУ ПРИРОДНИХ ВОДОЙМ. МОБІЛЬНИЙ ДОДАТОК АНАЛІЗУ ТА ВІЗУАЛІЗАЦІЇ ДАНИХ СПОСТЕРЕЖЕНЬ»

Виконав: студент 4 курсу, групи 2ІСТ-20б спеціальності 126 – «Інформаційні системи та технології»
_____Володимир ПАНАСЮК
Керівник: к.т.н., доц. каф. САІТ
_____Дмитро ПРОЦЕНКО
«31» 05 2024 р.
Рецензент: к.т.н., доц. каф. КН
_____Володимир ОЗЕРАНСЬКИЙ
«10» 06 2024 р.

**Допущено до захисту**

Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«03» 06 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 126 – «Інформаційні системи та технології»
Освітньо-професійна програма – Прикладні інформаційні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
"08" 05 2024 року

**ЗАВДАННЯ**
**НА КОМПЛЕКСНУ БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ**
**СТУДЕНТУ**
Панасюку Володимиру Романовичу

1. Тема роботи: «IoT-система моніторингу показників стану природних водойм. Мобільний додаток аналізу та візуалізації даних спостережень»
керівник роботи: Проценко Д. П., к.т.н., доц. каф. САІТ
затверджені наказом ВНТУ від «11» 05 2024 року №80
2. Термін подання студентом роботи 31.05
3. Вихідні дані до роботи:
1) Хмарні сервіси IoT: Sigfox Cloud, ThingSpeak.
2) Дані зібрані з пристрою для вимірювання якості поверхневих вод.
4. Зміст текстової частини:
1) загальна характеристика об'єкту дослідження. Аналіз інформаційних систем для моніторингу стану природних водойм.
2) проектування IoT-системи моніторингу. Розробка мобільного додатку для аналізу та візуалізації даних спостережень за станом природних водойм.
3) результати експериментальних досліджень.
5. Перелік ілюстративного матеріалу:
1) Архітектура IoT системи моніторингу стану природних водойм;
2) Модель даних IoT системи моніторингу стану природних водойм;
3) Архітектура ASP.NET Core Web APIs;
4) Алгоритм конвертування даних;
5) Місце проведення експеременту.

6. Консультанти розділів роботи:

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 11.03

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва та зміст етапу | Термін виконання | | Примітка |
|---|---|---|---|---|
| | | початок | закінчення | |
| 1 | Аналіз предметної області. | 11.03 | 31.03 | |
| 2 | Аналіз інформаційних систем для моніторингу стану природних водойм. Концепція мобільного додатку аналізу та візуалізації даних спостережень. | 01.04 | 15.04 | |
| 3 | Вибір мови програмування та середовища розробки, та дизайну додатка. | 16.04 | 30.04 | |
| 4 | Обробка результатів експерименту. | 01.05 | 15.05 | |
| 5 | Оформлення матеріалів до захисту БДР | 16.05 | 31.05 | |

Студент _____ Володимир ПАНАСЮК

Керівник роботи _____ Дмитро ПРОЦЕНКО

# АНОТАЦІЯ

Бакалаврська дипломна робота складається з 73 сторінок формату А4, на яких є 42 рисунка, 1 таблиці, список використаних джерел містить 24 найменування.

Метою даної роботи є розробка IoT-системи моніторингу показників стану природних водойм, яка дозволить автоматизувати процес збору даних про якість води та забезпечити їх оперативну візуалізацію та аналіз.

В роботі наведено розроблення IoT-системи моніторингу показників стану природних водойм, що включає мобільний додаток для аналізу та візуалізації даних.

Ключові слова: інформаційна система, IoT, моніторинг, поверхневі води, Sigfox, ThingSpeak.

# ABSTRACT

The Bachelor's thesis consists of 73 pages of A4 format, containing 42 figures, 1 table, and a list of references with 24 entries.

The objective of this thesis is the development of an IoT-based monitoring system for natural water bodies, which will automate the process of water quality data collection and provide their prompt visualization and analysis.

The thesis presents the development of an IoT-based monitoring system for natural water bodies, including a mobile application for data analysis and visualization.

Keywords: information system, IoT, monitoring, water surface, Sigfox, ThingSpeak.

# ЗМІСТ

# ВСТУП

У сучасному світі екологічні проблеми, пов'язані з забрудненням та деградацією природних водойм, набувають все більшої актуальності. Зростання антропогенного впливу, зміни клімату та інші фактори призводять до погіршення якості води, втрати біорізноманіття та порушення екосистем. Для ефективного управління водними ресурсами та своєчасного реагування на негативні зміни необхідні сучасні інструменти моніторингу та аналізу стану водойм.

Інтернет речей (IoT) відкриває нові можливості для вирішення цих завдань. IoT-системи дозволяють збирати дані про різноманітні параметри води (температура, pH, рівень кисню, наявність забруднюючих речовин тощо) в режимі реального часу, передавати їх на віддалений сервер для обробки та аналізу. Мобільні додатки, своєю чергою, забезпечують зручний доступ до цих даних, їх візуалізацію та інтерактивну взаємодію з користувачами.

**Актуальність теми** дослідження обумовлена потребою в розробці інноваційних рішень для моніторингу стану природних водойм, які б поєднували в собі переваги IoT-технологій та мобільних додатків. Такі системи можуть значно підвищити ефективність екологічного моніторингу, забезпечити оперативне виявлення проблем та прийняття своєчасних рішень щодо їх усунення.

**Мета і задачі дослідження.** Метою даної роботи є покращення процесу моніторингу та аналізу стану природних водойм завдяки розробці масштабованої IoT-системи, яка дозволить автоматизувати збір даних про якість поверхневих вод в режимі реального часу та забезпечити їх оперативну візуалізацію і аналіз.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

– проаналізувати існуючі інформаційні системи для моніторингу стану природних водойм та визначити їх переваги та недоліки;

– розробити концепцію мобільного додатку для аналізу та візуалізації даних спостережень, що враховуватиме особливості IoT-систем та забезпечуватиме зручний доступ до інформації для користувачів;

– обрати оптимальні мову програмування, середовище розробки та дизайн додатку, які забезпечать високу продуктивність, надійність та зручність використання системи;

– розробити та реалізувати мобільний додаток для аналізу та візуалізації даних спостережень, що дозволить користувачам отримувати оперативну інформацію про стан водних об'єктів у зручному форматі;

– провести тестування розробленої системи на реальних даних моніторингу та оцінити її ефективність.

**Об’єктом дослідження** є процес моніторингу стану природних водойм за допомогою IoT-технологій.

**Предметом дослідження** є IoT-система моніторингу показників стану природних водойм та мобільний додаток для аналізу та візуалізації даних.

**Публікації.** За результатами даної роботи була зроблена доповідь на тему «IoT система моніторингу поверхневих вод» на VI Міжнародній науковій студентській конференції Молодіжної наукової ліги (2024) з публікацією тез [1].

Робота виконувалась на замовлення науково-дослідної лабораторії автоматизованих систем управління в енергетиці та транспорті (НДЛ АСУЕТ) кафедри САІТ Вінницького національного технічного університету.

# 1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ОБ'ЄКТУ ДОСЛІДЖЕННЯ. АНАЛІЗ ІНФОРМАЦІЙНИХ СИСТЕМ ДЛЯ МОНІТОРИНГУ СТАНУ ПРИРОДНИХ ВОДОЙМ

## 1.1 Загальна характеристика об'єкту дослідження

Природні водойми, такі як річки, озера, водосховища та моря, є невід'ємною частиною біосфери Землі та відіграють ключову роль у підтримці екологічної рівноваги. Вони є джерелом питної води, забезпечують умови для розвитку різноманітних форм життя, використовуються для зрошення сільськогосподарських угідь, промисловості, рекреації та інших потреб. Стан природних водойм безпосередньо впливає на якість життя людини та стан навколишнього середовища.

Оцінка стану природних водойм є складним та багатогранним процесом, що вимагає комплексного підходу та врахування різноманітних факторів. Одним з ключових аспектів є моніторинг якості води, який включає в себе визначення фізичних, хімічних та біологічних показників.

Фізичні показники, такі як температура, прозорість, колір, електропровідність, мутність та солоність, характеризують загальний стан води та її придатність для різних видів використання [2]. Температура води впливає на розчинність газів, швидкість хімічних реакцій та життєдіяльність водних організмів. Прозорість води визначає глибину проникнення сонячного світла, що є важливим для фотосинтезу водних рослин та розвитку фітопланктону. Колір води може свідчити про наявність органічних речовин або забруднень. Електропровідність характеризує вміст розчинених солей у воді, а мутність - наявність завислих частинок. Солоність є важливим показником для морських та деяких континентальних водойм.

Хімічні показники відображають рівень забруднення води та її вплив на живі організми. Вміст розчиненого кисню є критичним для дихання риб та інших водних тварин. Значення pH визначає кислотність або лужність води, що впливає на хімічні процеси та життєдіяльність організмів. Біохімічна та хімічна потреба

в кисні характеризують кількість органічних речовин у воді та ступінь її забруднення. Концентрація нітратів та фосфатів свідчить про рівень евтрофікації водойми, тобто надмірного збагачення її поживними речовинами, що може призвести до цвітіння води та інших негативних наслідків. Наявність важких металів та інших токсичних речовин є серйозною загрозою для здоров'я людини та екосистем.

Біологічні показники дають змогу оцінити екологічний стан водойми та її здатність до самоочищення. Наявність та різноманітність фітопланктону, зоопланктону, риб, водоростей та інших водних організмів, а також індекси біорізноманіття відображають стан екосистеми та її стійкість до зовнішніх впливів. Зміни у складі та чисельності водних організмів можуть бути індикаторами забруднення або інших негативних змін у водоймі.

Гідрологічні показники, такі як рівень води, швидкість течії та витрата води, є важливими для оцінки водного балансу та управління водними ресурсами. Зміни рівня води можуть бути спричинені природними факторами, такими як опади та випаровування, або антропогенними факторами, такими як водозабір та регулювання стоку. Швидкість течії впливає на розподіл тепла, кисню та поживних речовин у водоймі. Витрата води є важливим показником для розрахунку водного балансу та планування водокористування.

Згідно Водного кодексу України, води (водні об’єкти) є виключною власністю народу України і надаються тільки у користування. Води України є її національним багатством, однією з природних основ економічного розвитку і соціального добробуту. Водні відносини регулюються цим Кодексом та іншими нормативно-правовими актами, прийнятими на його основі. Україна здійснює управління водним фондом, забезпечує раціональне використання і охорону вод та відтворення водних ресурсів, бере участь у транскордонному співробітництві у галузі використання і охорони вод та відтворення водних ресурсів, вживає заходів щодо запобігання шкідливій дії вод та ліквідації її наслідків. [3]

Однак, зростаюче антропогенне навантаження, пов'язане з промисловим виробництвом, сільськогосподарською діяльністю, урбанізацією та іншими факторами, призводить до забруднення водних об'єктів. Забруднення води

негативно впливає на якість питної води, стан водних екосистем та здоров'я людей.

Забруднення природних вод – це негативна зміна їх фізичних, хімічних та біологічних характеристик, спричинена діяльністю людини. Таке забруднення шкодить здоров'ю людей та інших живих організмів, а також обмежує використання води для різних потреб. Водойми вважаються забрудненими, коли якість їхньої води погіршується через промислові та побутові стоки, що робить її непридатною для певних видів використання. Забруднення водних ресурсів є глобальною проблемою, яка значно зменшує кількість доступної прісної води на Землі. Якість води впливає на стан різних реципієнтів (тваринний світ, рослинність, ґрунти, сільське, лісове і рибне господарства, транспорт, промислове виробництво, житлово-комунальна служба тощо) та перш за все на стан здоров’я населення [4].

Тому моніторинг якості води є важливим інструментом для оцінки стану водних об'єктів, виявлення джерел забруднення та розробки ефективних заходів щодо їх охорони та відновлення. Державний моніторинг вод здійснюється з метою: забезпечення збирання, обробки, зберігання, узагальнення й аналізу інформації про стан водних об’єктів, прогнозування його змін та розроблення науково обґрунтованих рекомендацій для прийняття рішень у галузі використання, охорони вод та відтворення водних ресурсів.[5]

Традиційні методи моніторингу якості води, що базуються на ручному відборі проб та лабораторному аналізі, є дорогими, трудомісткими та не забезпечують оперативного отримання даних. У зв'язку з цим виникає необхідність розробки нових, більш ефективних та доступних методів моніторингу стану водних об'єктів. Одним з перспективних напрямків є використання Інтернету речей (IoT) для створення систем автоматизованого збору, передачі та аналізу даних про якість води.

IoT-системи моніторингу якості води дозволяють здійснювати безперервний контроль за станом водних об'єктів, що забезпечує своєчасне виявлення забруднень та вжиття необхідних заходів. Такі системи складаються з мережі датчиків, які встановлюються у водоймах та вимірюють різні параметри

якості води, такі як температура, pH, розчинений кисень, електропровідність, каламутність та інші. Дані з датчиків передаються на віддалений сервер через бездротові мережі зв'язку, де вони обробляються, аналізуються та візуалізуються у зручному для користувача вигляді.

Використання IoT-технологій для моніторингу якості води має ряд переваг порівняно з традиційними методами:

– Оперативність: Дані про якість води доступні у режимі реального часу, що дозволяє швидко реагувати на зміни стану водних об'єктів;

– Автоматизація: Процес збору та передачі даних відбувається автоматично, що знижує витрати на персонал та мінімізує ризик людських помилок;

– Масштабованість: IoT-системи легко масштабуються, що дозволяє розширювати мережу датчиків та охоплювати більшу кількість водних об'єктів;

– Доступність: Дані моніторингу доступні через веб-інтерфейс або мобільний додаток, що забезпечує зручність та доступність інформації для широкого кола користувачів;

– Економічність: Використання IoT-технологій дозволяє знизити витрати на моніторинг якості води за рахунок автоматизації процесів та зменшення потреби у ручній праці.

Таким чином, розробка IoT-систем моніторингу якості води є актуальним та перспективним напрямком, який має значний потенціал для покращення стану водних ресурсів та забезпечення екологічної безпеки.

## 1.2 Аналіз існуючих рішень для моніторингу показників стану природних водойм

Сучасні технології пропонують різноманітні рішення для моніторингу стану природних водойм, кожне з яких має свої переваги та обмеження. Традиційні методи моніторингу, такі як відбір проб води та їх лабораторний аналіз, є досить точними та надійними, але вони вимагають значних витрат часу та ресурсів, а також не дозволяють отримувати дані в режимі реального часу [6].

Крім того, лабораторний аналіз може бути обмежений кількістю параметрів, які можна виміряти, та вимагає спеціальних знань та обладнання.

З розвитком інформаційних технологій та Інтернету речей (IoT) з'явилися нові можливості для автоматизації та оптимізації процесів моніторингу. IoT-системи дозволяють збирати дані про різноманітні параметри води за допомогою сенсорів, передавати їх на віддалений сервер для обробки та аналізу, а також візуалізувати результати у зручній для користувача формі [7]. Це дозволяє отримувати більш повну та актуальну інформацію про стан водойм, а також знизити витрати на моніторинг.

Одним з найпоширеніших підходів є використання стаціонарних станцій моніторингу, які встановлюються на березі водойми або на спеціальних платформах (рис. 1.1). Такі станції оснащені різноманітними сенсорами, які вимірюють температуру, pH, рівень кисню, електропровідність, мутність та інші параметри води. Дані з сенсорів передаються на сервер через дротові або бездротові канали зв'язку [8]. Стаціонарні станції дозволяють проводити безперервний моніторинг якості води в певній точці водойми, що дає змогу виявляти зміни в режимі реального часу та оперативно реагувати на них.

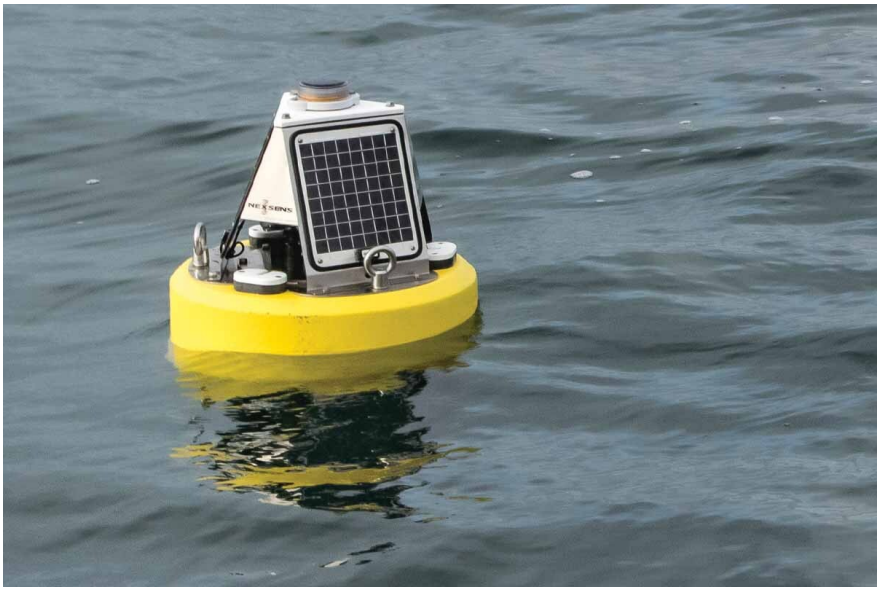


Рисунок 1.1 - Приклад використання буїв від «Adroit»

У сучасному світі існує безліч рішень для моніторингу якості води, які можна розділити на кілька основних категорій:

Традиційні методи:

– Ручний відбір проб та лабораторний аналіз: Цей метод є найбільш поширеним та використовується для визначення широкого спектру параметрів якості води. Проте, він має ряд недоліків, таких як висока вартість, трудомісткість, низька оперативність та обмежена кількість точок відбору проб;

– Автоматизовані станції моніторингу: Ці станції дозволяють здійснювати безперервний моніторинг якості води в певних точках, але їх встановлення та обслуговування є дорогим, а кількість точок моніторингу обмежена.

IoT-системи моніторингу:

– Системи на базі дротових датчиків: Такі системи забезпечують високу точність вимірювань та надійність передачі даних, але їх встановлення та обслуговування є складним та дорогим, особливо у важкодоступних місцях;

– Системи на базі бездротових датчиків: Ці системи є більш гнучкими та дешевшими у встановленні, але можуть мати обмежений радіус дії та проблеми з передачею даних у складних умовах;

– Системи на базі хмарних технологій: Такі системи дозволяють зберігати, обробляти та аналізувати дані в хмарі, що забезпечує високу масштабованість та доступність даних з будь-якого місця. Проте, вони можуть мати проблеми з безпекою даних та залежністю від інтернет-з'єднання.

Інші методи:

– Дистанційний моніторинг: Використання супутникових знімків, дронів та інших дистанційних методів дозволяє отримувати інформацію про стан водних об'єктів на великих територіях, але точність таких даних може бути обмеженою;

– Біомоніторинг: Використання живих організмів (риб, водоростей, безхребетних) як індикаторів якості води. Цей метод дозволяє оцінити вплив забруднення на водні екосистеми, але він є більш складним та трудомістким, ніж фізико-хімічні методи.

Серед найбільш відомих та використовуваних у світі IoT-платформ для моніторингу якості води можна виділити такі:

– Libelium: Іспанська компанія, що пропонує широкий спектр IoT-рішень для різних галузей, включаючи моніторинг навколишнього середовища. Їх платформа Waspmote дозволяє створювати гнучкі та масштабовані системи моніторингу якості води з використанням різних типів датчиків та бездротових технологій;

– EnviroDIY: Відкрита платформа для створення власних систем моніторингу якості води. Вона базується на апаратному забезпеченні Arduino та програмному забезпеченні з відкритим кодом. EnviroDIY дозволяє користувачам налаштовувати систему під свої потреби та інтегрувати її з іншими платформами та сервісами;

– Hobo: Компанія Onset пропонує ряд IoT-рішень для моніторингу навколишнього середовища, включаючи систему Hobo U26 Dissolved Oxygen Logger, яка дозволяє вимірювати рівень розчиненого кисню у воді. Ця система відрізняється високою точністю вимірювань та можливістю автономної роботи протягом тривалого часу;

– Sigfox: Французька компанія, що пропонує глобальну мережу низької потужності для IoT-пристроїв. Sigfox забезпечує передачу даних на великі відстані з низьким енергоспоживанням, що робить її привабливим варіантом для моніторингу якості води у віддалених та важкодоступних місцях;

– ThingSpeak: Відкрита IoT-платформа від MathWorks, яка дозволяє збирати, зберігати, аналізувати та візуалізувати дані з IoT-пристроїв. ThingSpeak має зручний інтерфейс та широкий набір

інструментів для аналізу даних, що робить її популярним вибором для розробників IoT-систем.

В Україні також існують власні розробки в галузі моніторингу якості води, такі як:

– АСУ «ЕкоІнспектор»: Єдина автоматизована система Державної екологічної інспекції України, яка призначена для збору, обробки та аналізу даних про стан навколишнього середовища, включаючи якість води;

– Система моніторингу якості вод Держводагентства: Система, розроблена Науково-дослідним інститутом проблем математичних машин та систем НАН України, яка забезпечує збір та аналіз даних про якість води у водних об'єктах України.

Проте, існуючі в Україні системи мають ряд недоліків, таких як:

– Відсутність автоматизації процесів супроводження відбору проб та вимірювань: вносяться лише результати вимірювань, що може призводити до втрати важливої інформації та ускладнювати аналіз даних;

– Недостатній облік похибок вимірювань: вимірювання одних і тих же показників у різних областях здійснюється різними приладами та за різними методиками, що може призводити до розбіжностей у результатах та ускладнювати їх інтерпретацію;

– Обмежена кількість показників, що визначаються системами: деякі системи моніторингу враховують лише обмежений набір параметрів якості води, що не дозволяє отримати повну картину стану водних об'єктів.

Таким чином, аналіз існуючих рішень для моніторингу показників стану природних водойм показує, що існує значний потенціал для вдосконалення та розвитку систем моніторингу якості води. Застосування сучасних технологій, таких як IoT, хмарні обчислення та машинне навчання, дозволить створити більш ефективні, точні та доступні системи, які забезпечать своєчасне виявлення забруднень та допоможуть покращити управління водними ресурсами.

## 1.3 Огляд технологій розробки мобільних додатків

Мобільні додатки стали невід'ємною частиною нашого життя, надаючи зручний доступ до інформації, сервісів та розваг. У контексті моніторингу стану природних водойм мобільні додатки відіграють важливу роль, дозволяючи користувачам отримувати актуальні дані про якість води, аналізувати їх та приймати обґрунтовані рішення [9]. Завдяки мобільним додаткам, дані моніторингу стають доступними широкому колу користувачів, включаючи громадськість, науковців, державні органи та інші зацікавлені сторони. Це сприяє підвищенню обізнаності про екологічні проблеми та залученню громадян до їх вирішення. Існує декілька підходів до розробки мобільних додатків, кожен з яких має свої переваги та недоліки.

Нативні додатки розробляються для конкретної мобільної платформи (iOS або Android) з використанням мов програмування та інструментів, рекомендованих виробником платформи (Swift або Objective-C для iOS, Java або Kotlin для Android) [10]. Нативні додатки мають високу продуктивність, швидкість роботи та доступ до всіх функцій пристрою, таких як камера, GPS, акселерометр тощо. Це дозволяє створювати додатки з розширеними можливостями, такими як використання геолокації для відображення даних моніторингу на карті, фотографування водойм та їх стану, а також використання сенсорів пристрою для збору додаткових даних. Однак, розробка нативних додатків вимагає значних витрат часу та ресурсів, оскільки необхідно створювати окремі версії для кожної платформи. Це пов'язано з тим, що кожна платформа має свої особливості, вимоги до дизайну та інтерфейсу користувача, а також використовує різні мови програмування та інструменти розробки. Крім того, підтримка та оновлення нативних додатків також вимагають додаткових зусиль та витрат, оскільки необхідно враховувати зміни в операційних системах та апаратних засобах пристроїв.

Кросплатформні додатки розробляються з використанням фреймворків, які дозволяють створювати єдиний код, що може бути використаний на різних платформах. Серед популярних кросплатформних фреймворків можна виділити

React Native, Flutter, Xamarin та Ionic [11]. Кросплатформні додатки мають нижчу вартість розробки та швидший час виходу на ринок, оскільки код можна використовувати повторно для різних платформ. Це особливо актуально для проектів з обмеженим бюджетом або тих, що потребують швидкого запуску. Однак, кросплатформні додатки можуть мати нижчу продуктивність та обмежений доступ до функцій пристрою порівняно з нативними додатками. Це пов'язано з тим, що вони використовують проміжний шар абстракції, який може впливати на швидкість роботи та доступ до апаратних ресурсів. Крім того, кросплатформні фреймворки можуть мати свої обмеження та особливості, які необхідно враховувати при розробці додатків.

Прогресивні веб-додатки (PWA) - це веб-сайти, які мають функціональність мобільних додатків, такі як можливість роботи в офлайн-режимі, надсилання push-повідомлень та встановлення на головний екран пристрою [12]. PWA розробляються з використанням стандартних веб-технологій (HTML, CSS, JavaScript) та можуть бути доступні на будь-якому пристрої з веб-браузером. Вони мають низьку вартість розробки та легкий доступ для користувачів, оскільки не вимагають встановлення з магазину додатків. Це особливо зручно для користувачів, які не бажають або не мають можливості встановлювати додаткові програми на свої пристрої. Однак, PWA можуть мати обмежений доступ до функцій пристрою та нижчу продуктивність порівняно з нативними та кросплатформними додатками. Це пов'язано з тим, що вони працюють у веб-браузері, який має свої обмеження щодо доступу до апаратних ресурсів та швидкості виконання коду. Крім того, PWA можуть мати проблеми з підтримкою деяких функцій, таких як геолокація та push-повідомлення, на деяких пристроях та браузерах.

Гібридні додатки поєднують в собі елементи нативних та веб-додатків. Вони розробляються з використанням веб-технологій (HTML, CSS, JavaScript), але запускаються в нативному контейнері, що дозволяє їм отримати доступ до деяких функцій пристрою [13]. Гібридні додатки є компромісом між нативними та кросплатформними додатками, поєднуючи в собі їх переваги та недоліки. Вони дозволяють використовувати єдиний код для різних платформ, але можуть

мати нижчу продуктивність та обмежений доступ до функцій пристрою порівняно з нативними додатками.

## 1.4 Обґрунтування вибору технологічного стеку для розробки мобільного додатку

Вибір технологічного стеку для розробки мобільного додатку є критичним етапом, оскільки він визначає продуктивність, масштабованість, безпеку та зручність використання майбутнього продукту. У контексті розробки IoT-системи моніторингу стану природних водойм важливо враховувати специфіку задачі та вимоги до додатку.

Серверна частина. C# та .NET Web API: C# є потужною та універсальною мовою програмування, що дозволяє створювати високонадійні та масштабовані серверні додатки [14]. .NET Web API надає зручний фреймворк для створення RESTful API, що забезпечує взаємодію між сервером та клієнтом. Цей стек технологій добре підходить для розробки серверної частини IoT-системи, оскільки він забезпечує високу продуктивність, надійність та безпеку.

Entity Framework та MSSQL Server 2022: Entity Framework є популярним ORM-фреймворком для .NET, який спрощує роботу з базами даних та дозволяє використовувати об'єктно-орієнтований підхід до моделювання даних [15]. MSSQL Server 2022 є потужною та надійною системою управління базами даних, що забезпечує високу продуктивність та масштабованість [16]. Цей стек технологій добре підходить для зберігання та обробки великих обсягів даних, що надходять з IoT-пристроїв.

Клієнтська частина (фронтенд). React 17, TypeScript та Material UI: React є популярною JavaScript-бібліотекою для створення користувацьких інтерфейсів, що дозволяє створювати динамічні та інтерактивні веб-додатки [17]. TypeScript є надмножиною JavaScript, що додає статичну типізацію та інші можливості, які

підвищують надійність та зручність розробки. Material UI є бібліотекою компонентів React, що реалізує дизайн Material Design від Google, забезпечуючи привабливий та зручний інтерфейс користувача. Цей стек технологій добре підходить для створення адаптивних веб-додатків, які можуть працювати на різних пристроях та платформах.

Порівняння з іншими технологіями відображено на таблиці 1.1.

Таблиця 1.1 – Порівняння технологій розробки мобільних додатків

| Технологія | Переваги | Недоліки |
|---|---|---|
| Нативні додатки | Висока продуктивність, доступ до всіх функцій пристрою | Висока вартість розробки, необхідність створення окремих версій для кожної платформи |
| PWA | Низька вартість розробки, легкий доступ для користувачів | Обмежений доступ до функцій пристрою, нижча продуктивність |
| Flutter | Швидкий час розробки, гарний зовнішній вигляд віджетів | Великий розмір додатків, обмежений доступ до нативних функцій |
| Xamarin | Єдиний код для різних платформ, доступ до нативних API | Висока складність розробки, проблеми з продуктивністю на деяких платформах |
| Обраний стек | Зручність розробки, висока продуктивність, масштабованість, адаптивність, кросплатформність, знайомі інструменти | Відсутність доступу до специфічних функцій пристрою (наприклад, AR) |

Обґрунтуємо вибір. Зручність розробки та підтримки: Використання єдиного технологічного стеку (.NET) для серверної та клієнтської частини спрощує розробку та підтримку додатку, оскільки розробники можуть використовувати знайомі інструменти та підходи.

Висока продуктивність та масштабованість: .NET Web API та MSSQL Server 2022 забезпечують високу продуктивність та масштабованість, що дозволяє обробляти великі обсяги даних та підтримувати велику кількість користувачів.

Адаптивність та кросплатформність: React, TypeScript та Material UI дозволяють створювати адаптивні веб-додатки, які можуть працювати на різних пристроях та платформах, забезпечуючи зручний доступ до даних моніторингу для широкого кола користувачів.

# 2 ПРОЕКТУВАННЯ ІОТ СИСТЕМИ МОНІТОРИНГУ. РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ АНАЛІЗУ ТА ВІЗУАЛІЗАЦІЇ ДАНИХ СПОСТЕРЕЖЕНЬ ЗА СТАНОМ ПРИРОДНИХ ВОДОЙМ

## 2.1 Розробка концепції IoT системи моніторингу показників стану природних водойм

IoT-система моніторингу стану природних водойм являє собою комплекс взаємопов'язаних компонентів, що забезпечують збір, передачу, обробку та аналіз даних про різноманітні параметри води. Основною метою такої системи є отримання актуальної та достовірної інформації про стан водойм, що дозволяє своєчасно виявляти проблеми та приймати обґрунтовані рішення щодо їх усунення.

Основні компоненти IoT-системи моніторингу (рис. 2.1):

– Сенсори: Пристрої, що вимірюють різноманітні параметри води, такі як температура, pH, рівень кисню, електропровідність, мутність, солоність, концентрація забруднюючих речовин тощо. Сенсори можуть бути різного типу: електрохімічні, оптичні, акустичні, термічні тощо. Вибір сенсорів залежить від конкретних завдань моніторингу та вимог до точності вимірювань;

– Мікроконтролери: Пристрої, що керують роботою сенсорів, збирають дані з них, обробляють їх та передають на віддалений сервер. Мікроконтролери можуть бути вбудовані в сенсори або встановлені окремо. Вони забезпечують автономну роботу системи та дозволяють реалізувати різноманітні алгоритми обробки даних;

– Комунікаційні модулі: Пристрої, що забезпечують передачу даних з мікроконтролерів на віддалений сервер. Комунікаційні модулі можуть використовувати різні технології передачі даних: Wi-Fi, Bluetooth, LoRaWAN, GSM тощо. Вибір технології залежить від відстані між сенсорами та сервером, вимог до енергоспоживання та вартості обладнання;

– Сервер: Комп'ютер або хмарна платформа, що приймає дані з сенсорів, зберігає їх у базі даних, обробляє та аналізує. Сервер може реалізувати різноманітні функції, такі як візуалізація даних, формування звітів, прогнозування змін стану водойм, сповіщення користувачів про перевищення нормативних значень параметрів води тощо;

– Клієнтський додаток: Програмне забезпечення, що дозволяє користувачам отримувати доступ до даних моніторингу, аналізувати їх та візуалізувати у зручній формі. Клієнтський додаток може бути реалізований у вигляді веб-додатку, мобільного додатку або десктопного додатку.

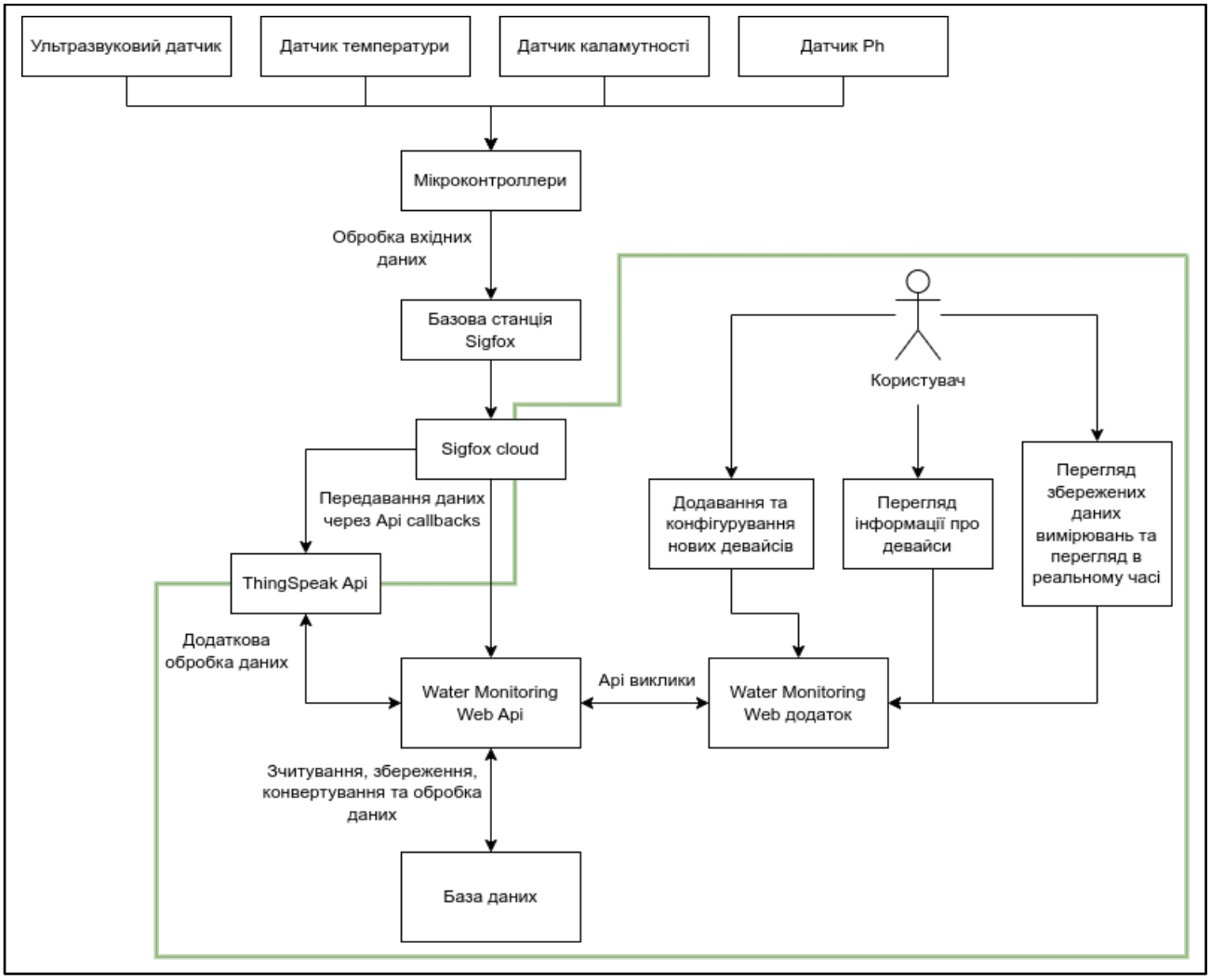



Рисунок 2.1 – Архітектура IoT системи моніторингу стану природних водойм

Переваги використання IoT-системи моніторингу:

– Автоматизація процесу збору даних: IoT-система дозволяє автоматизувати процес збору даних, що знижує витрати на моніторинг та мінімізує вплив людського фактору на результати вимірювань;

– Отримання даних у режимі реального часу: IoT-система забезпечує отримання даних про стан водойм у режимі реального часу, що дозволяє оперативно реагувати на зміни та приймати своєчасні рішення;

– Комплексний аналіз даних: IoT-система дозволяє збирати дані про різноманітні параметри води, що дає змогу проводити комплексний аналіз стану водойм та виявляти взаємозв'язки між різними факторами;

– Зручний доступ до даних: Клієнтський додаток забезпечує зручний доступ до даних моніторингу для широкого кола користувачів, що підвищує обізнаність про екологічні проблеми та сприяє залученню громадян до їх вирішення;

– Можливість прогнозування: Завдяки аналізу великих обсягів даних, IoT-система може виявляти тенденції та прогнозувати зміни стану водойм, що дозволяє вживати профілактичних заходів та запобігати негативним наслідкам.

## 2.2 Проектування бази даних

База даних є ключовим компонентом IoT-системи моніторингу стану природних водойм, оскільки вона забезпечує зберігання, обробку та доступ до великих обсягів даних, що надходять з сенсорів. Правильно спроектована база даних дозволяє ефективно зберігати та отримувати дані, забезпечує цілісність та узгодженість даних, а також оптимізує продуктивність системи.

Для даного проекту було обрано Microsoft SQL Server 2022 як СУБД. Цей вибір обумовлений наступними факторами:

– Висока продуктивність та масштабованість: MSSQL Server 2022 забезпечує високу продуктивність та масштабованість, що дозволяє

обробляти великі обсяги даних та підтримувати велику кількість користувачів [18];

– Надійність та безпека: MSSQL Server 2022 має вбудовані механізми забезпечення надійності та безпеки даних, такі як резервне копіювання, відновлення після збоїв, шифрування даних та контроль доступу;

– Широкий спектр функцій: MSSQL Server 2022 підтримує широкий спектр функцій для обробки та аналізу даних, включаючи вбудовані засоби машинного навчання, що може бути корисним для аналізу даних моніторингу та прогнозування стану водойм;

– Інтеграція з .NET: MSSQL Server 2022 легко інтегрується з .NET платформою та Entity Framework, що спрощує розробку серверної частини додатку.

Модель даних визначає структуру бази даних та взаємозв'язки між її елементами. Для IoT-системи моніторингу стану природних водойм було розроблено наступну модель даних (рис. 2.2).

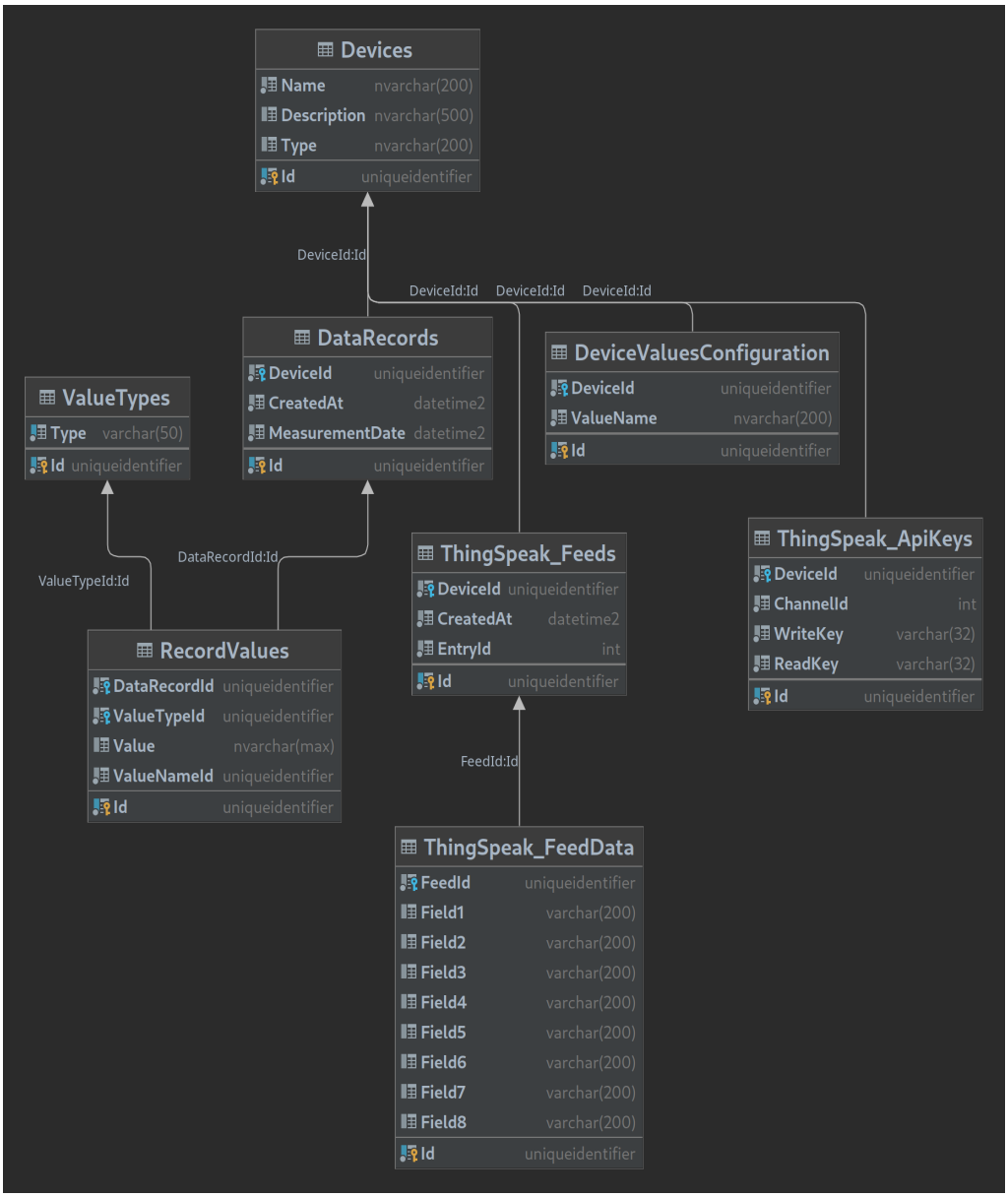



Рисунок 2.2 - Модель даних IoT системи моніторингу стану природних водойм

Опис таблиць:

– Таблиця "Devices": Зберігає інформацію про пристрої, а саме назву, опис та тип пристрою;

– Таблиця "ThingSpeak_ApiKeys": Зберігає конфігурацію для роботи з ThingSpeak Api, а саме номер каналу, ключ для запису та ключ для читання;

– Таблиця "ThingSpeak_Feeds": Зберігає дані записів отриманих з ThingSpeak, а саме дату створення запису, та EntryId, який є внутршнім ідентифікатор запису ThingSpeak;

– Таблиця "ThingSpeak_FeedData": Зберігає дані вимірювань для кожного запису, і збергіє дані в тому ж форматі що і ThingSpeak;

– Таблиця "DeviceValuesConfiguration": Зберігає конфігурацію значень пристрою, які необхідно моніторити, аналізувати та відображати;

– Таблиця "ValueTypes": Зберігає типи даних до яких необхідно приводити необроблені дані, отримані з ThingSpeak, або ж напряму з Sigfox;

– Таблиця "DataRecords": Зберігає інформацію про вимірювання, а саме дату створення та дату вимірювання;

– Таблиця "RecordValues": Зберігає інформацію про вимірювання, а саме вимірюване значення.

Взаємозв'язки між таблицями:

– Таблиця "Devices" має зв'язок "один-до-багатьох" з таблицями "ThingSpeak_ApiKeys", "ThingSpeak_Feeds", "DataRecords", "DeviceValuesConfiguration";

– Таблиця "ThingSpeak_Feeds" має зв'язок "один-до-багатьох" з таблицею "ThingSpeak_FeedData";

– Таблиця "DeviceValuesConfiguration" має зв'язок "один-до-багатьох" з таблицею "RecordValues";

– Таблиця "ValueTypes" має зв'язок "один-до-багатьох" з таблицею "RecordValues";

– Таблиця "DataRecords" має зв'язок "один-до-багатьох" з таблицею "RecordValues";

## 2.3 Розробка серверної частини

Серверна частина IoT-системи моніторингу стану природних водойм відіграє ключову роль у забезпеченні функціональності всієї системи. Вона відповідає за прийом, обробку, зберігання та надання даних, отриманих від сенсорів, а також за взаємодію з клієнтським додатком.

Для розробки серверної частини було обрано наступний технологічний стек:

1. Мова програмування: C#;
2. Фреймворк: .NET Web API;

3. ORM: Entity Framework;
4. СУБД: MSSQL Server 2022.

.NET C# Web API може бути використаний для створення backend сервісів, які збирають, обробляють та надають дані про якість води з різних джерел (датчики, лабораторні аналізи тощо). Ці сервіси можуть бути інтегровані з веб-інтерфейсами, мобільними додатками та іншими системами для забезпечення комплексного моніторингу та управління водними ресурсами (рис. 2.3).

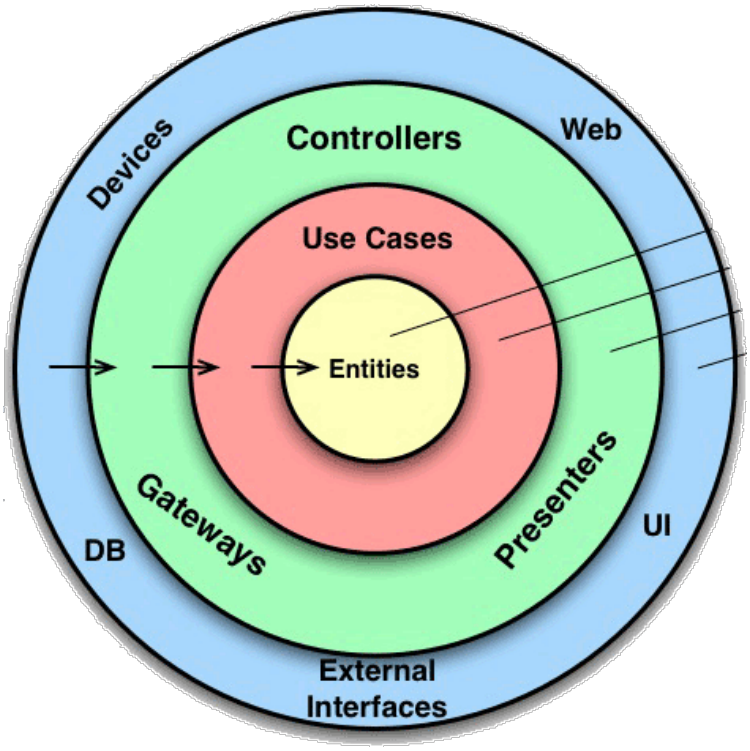
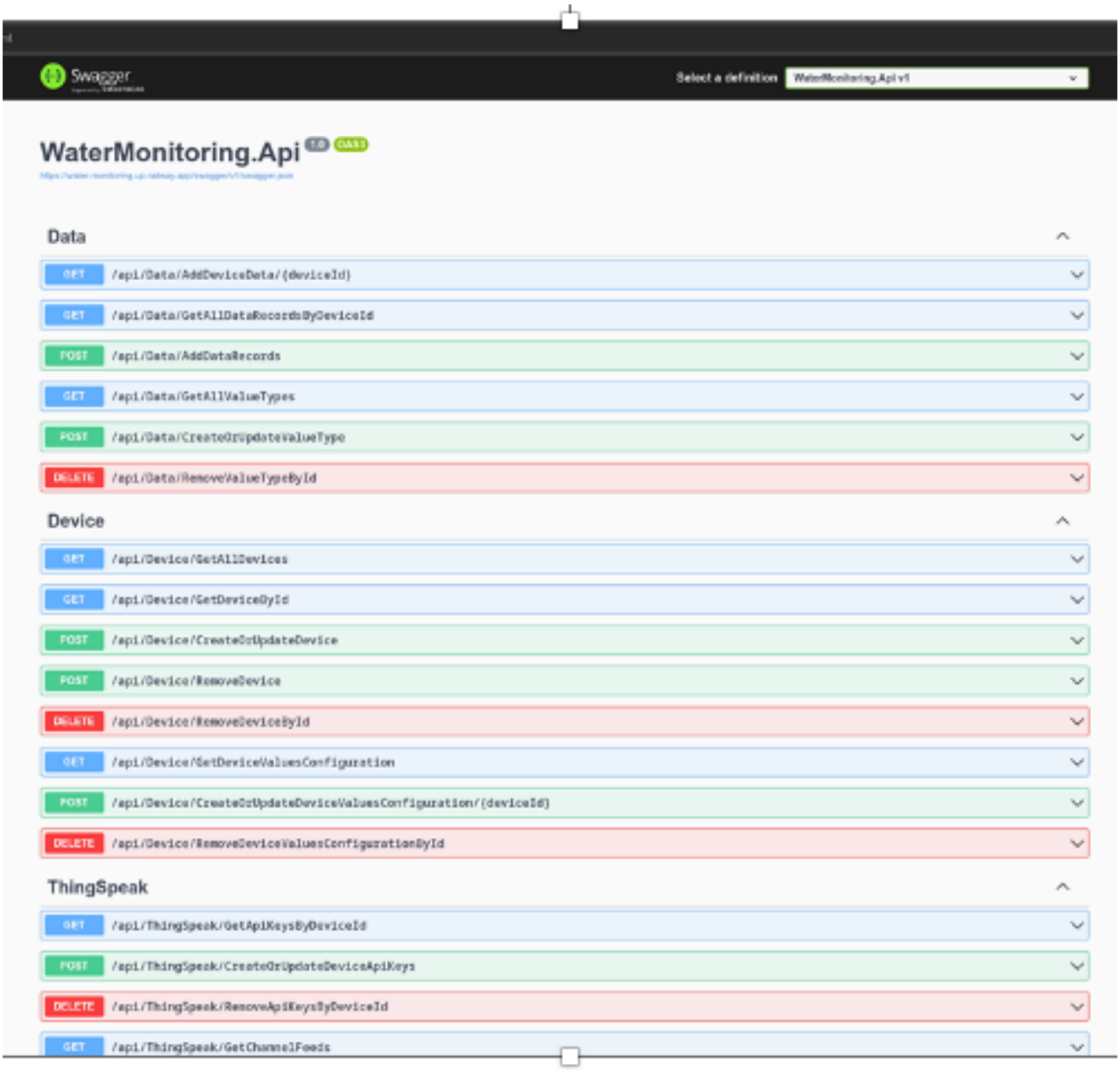


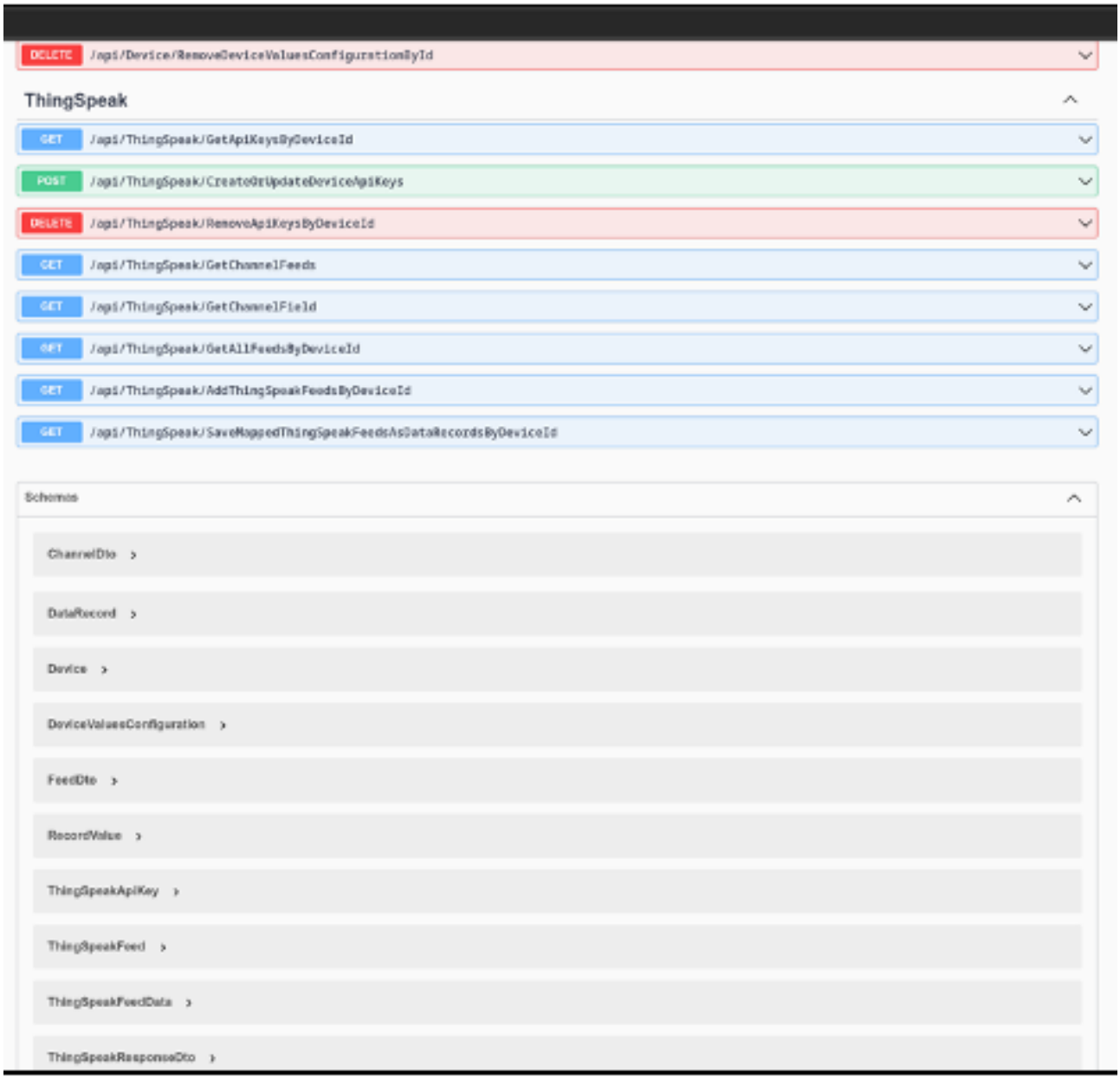

Рисунок 2.3 – Архітектура ASP.NET Core Web APIs

Серверна частина побудована за архітектурою RESTful API, що забезпечує стандартизований спосіб взаємодії між сервером та клієнтом. API надає набір ендпоінтів для виконання різних операцій з даними (рис. 2.4, 2.5), таких як:

– Отримання даних;
– Додавання даних;
– Оновлення даних;
– Видалення даних.

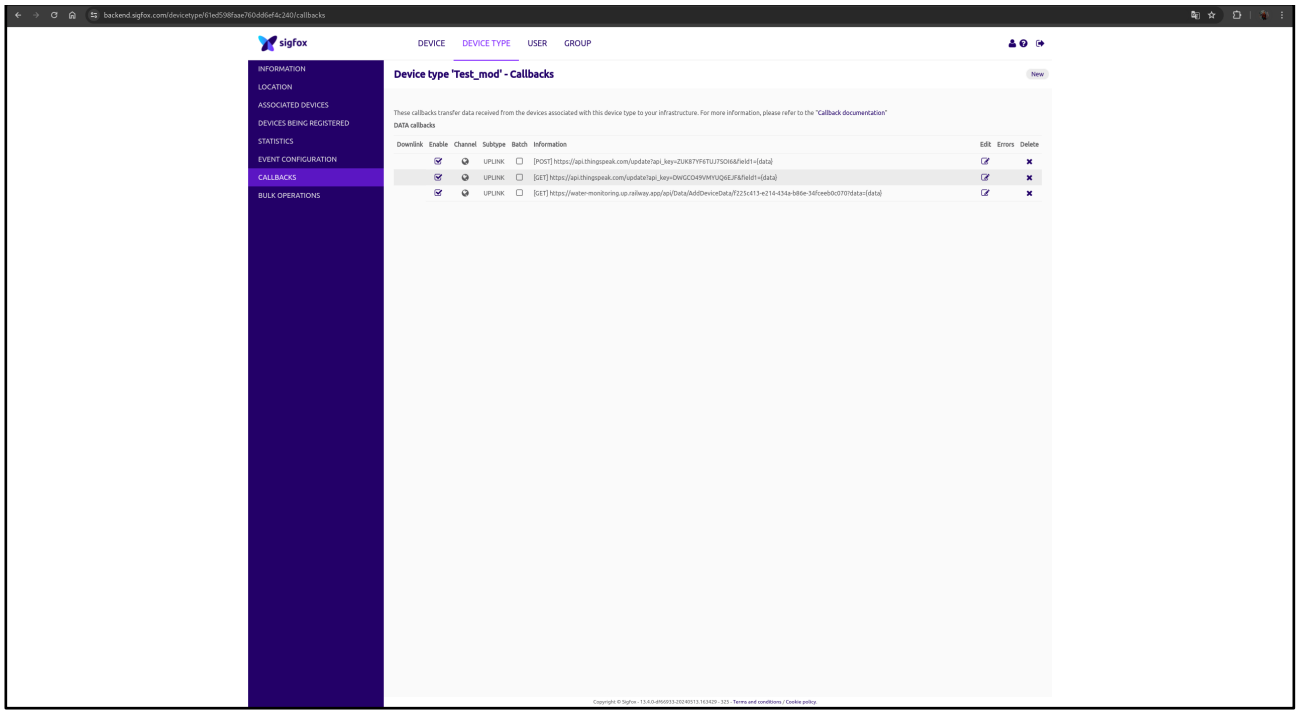


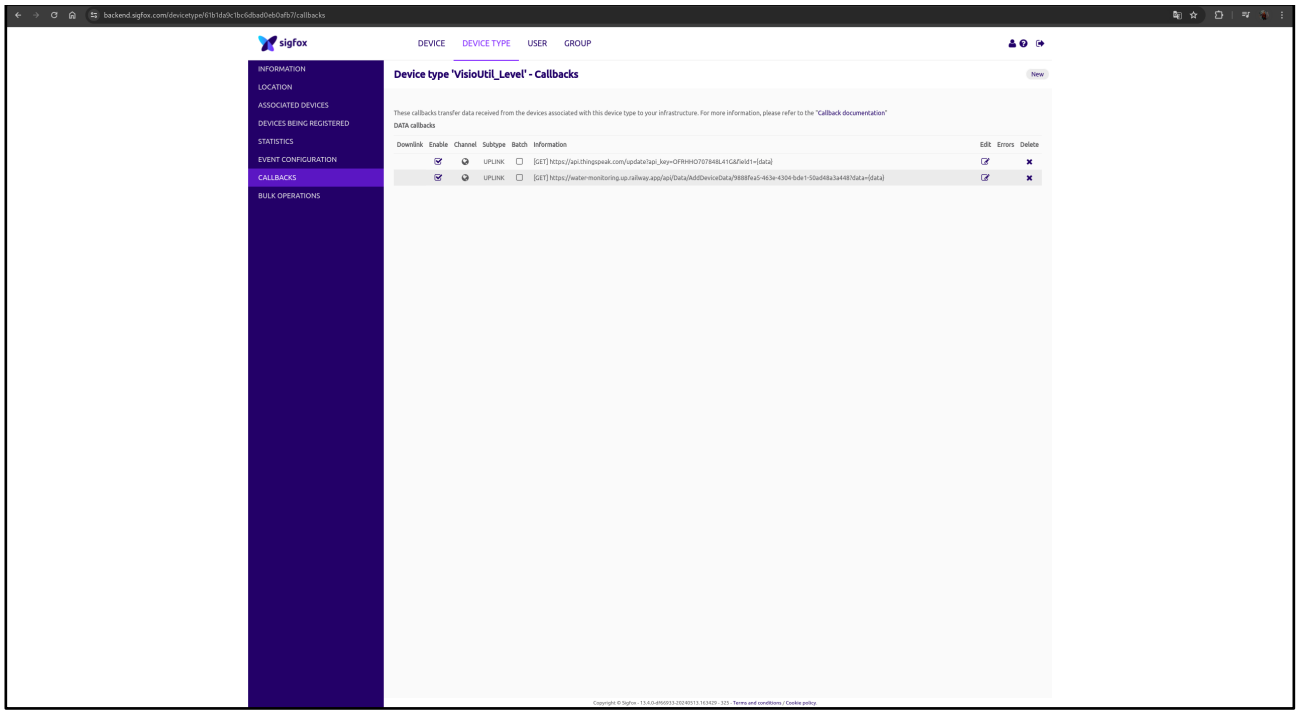

Рисунки 2.4 – Інтерфейс Swagger для виклику ендопінтів API




Рисунки 2.5 – Інтерфейс Swagger для виклику ендопінтів API (продовження)

Налаштування Api Callbacks на платформі Sigfox Backend для взаємодії з розробленим Api та сервісом ThingsSpeak (рис. 2.6 - 2.7).




Рисунки 2.6 - 2.7 – Налаштування Api Callbacks на платформі Sigfox Backend

Функціональні можливості серверної частини:

– Додавання, конфігурування та видалення пристроїв;

– Додавання, тестових даних вручну;

– Отримання та збереження необроблених даних вимірювань з ThingSpeak Api, або ж напряму з Sigfox Backend;

– Читання даних необроблених даних;

– Конвертація (рис. 2.8), обробка, збереження та читання даних;
– Читання та відображення даних в режимі реального часу;
– Видалення даних.

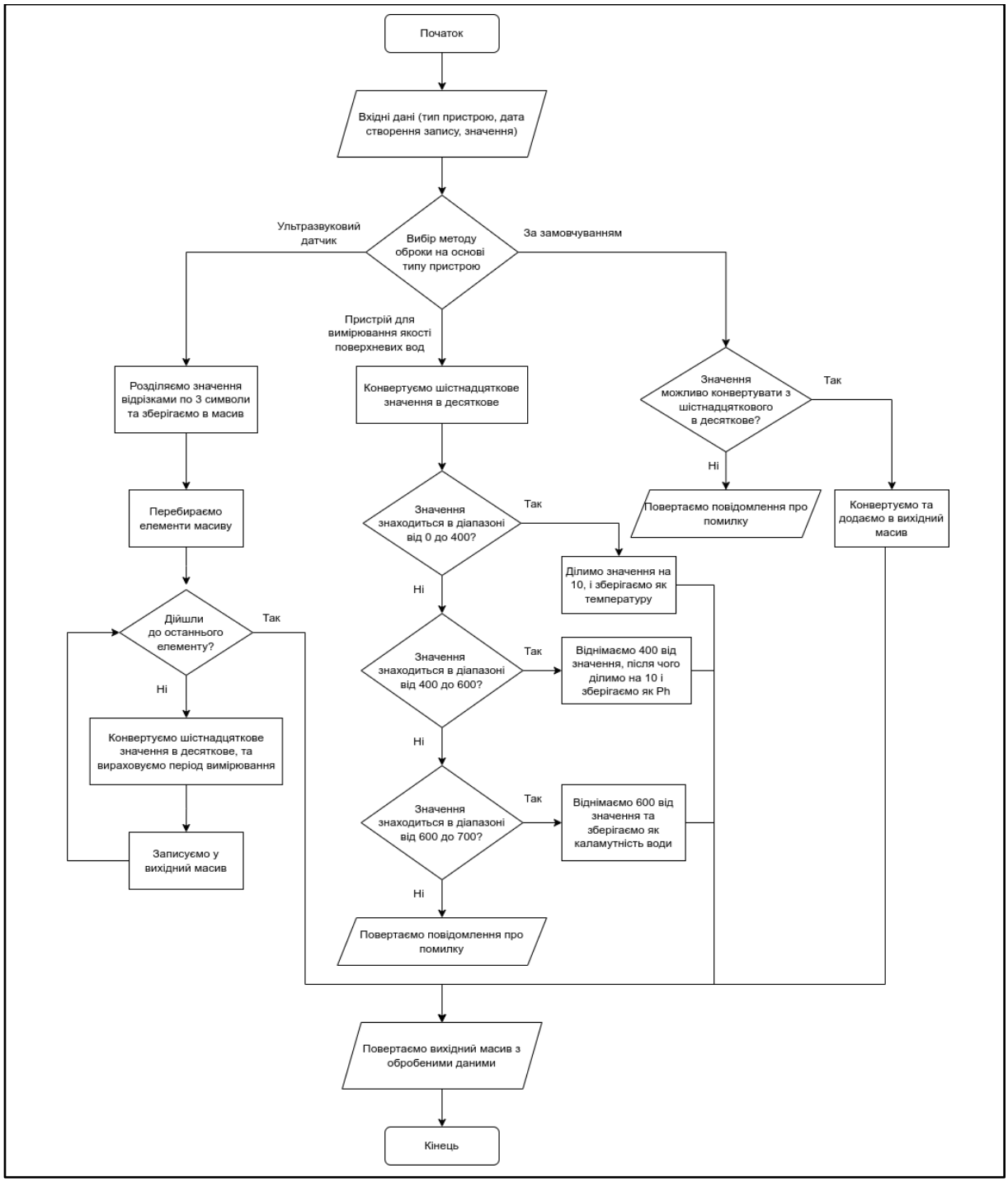



Рисунок 2.8 – Алгоритм конвертування даних

Представлений алгоритм (рис. 2.8) демонструє яким чином дані отримані з SigfoxCloud, або ж ThingSpeak будуть конвертуватися для подальшого

збереження та аналізу. Спочатку буде обиратися відповідний алгоритм для конвертації даних в залежності від типу пристрою, після чого дані будуть збережені у вигляді колекції, зі значеннями вимірювань, типом значення, а також датою і часом замірів. У випадку виникнення будь яких помилок при конвертації, буде з’являтися повідомлення про помилку. Програмна реалізація алгоритму представлена на рисунках 2.9 - 2.11.

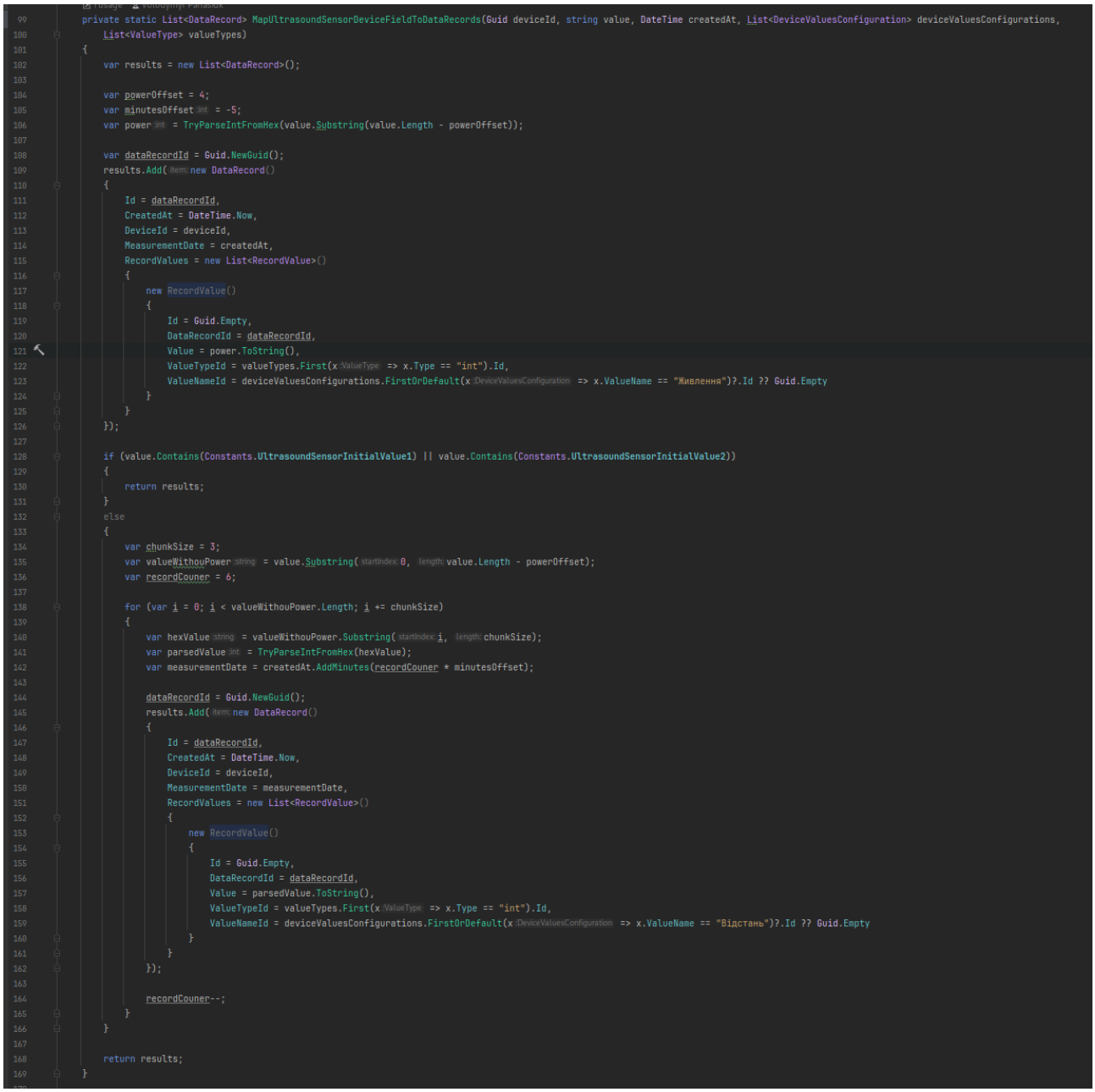
```
public static List<DataRecord> MapHexDeviceFieldToDataRecords(Guid deviceId, string deviceType, string value, DateTime createdAt, List<Devic
    List<ValueType> valueTypes)
{
    var results = new List<DataRecord>();

    if (string.IsNullOrEmpty(value))
    {
        return results;
    }

    try
    {
        switch (deviceType)
        {
            case Constants.UltrasoundSensorDeviceType:
                results = MapUltrasoundSensorDeviceFieldToDataRecords(deviceId, value, createdAt, deviceValuesConfigurations, valueTypes);
                break;
            case Constants.WaterQualitySensorDeviceType:
                results = MapWaterQualitySensorToDataRecords(deviceId, value, createdAt, deviceValuesConfigurations, valueTypes);
                break;
            default:
                var intValue = Convert.ToInt32(value, fromBase:16);
                var dataRecordId = Guid.NewGuid();
                results.Add(item:new DataRecord()
                {
                    Id = dataRecordId,
                    CreatedAt = DateTime.Now,
                    DeviceId = deviceId,
                    MeasurementDate = createdAt,
                    RecordValues = new List<RecordValue>()
                    {
                        new RecordValue()
                        {
                            Id = Guid.Empty,
                            DataRecordId = dataRecordId,
                            Value = intValue.ToString(),
                            ValueTypeId = valueTypes.First(x:ValueType => x.Type == "int").Id,
                            ValueNameId = deviceValuesConfigurations.First().Id
                        }
                    }
                });
                break;
        }

        return results;
    }
    catch (Exception e)
    {
        return results;
    }
}
```

Рисунок 2.9 – Код вибору алгоритму конвертації відповідно до типу пристрою



```
private static List<DataRecord> MapUltrasoundSensorDeviceFieldToDataRecords(Guid deviceId, string value, DateTime createdAt, List<DeviceValuesConfiguration> deviceValuesConfigurations,
    List<ValueType> valueTypes)
{
    var results = new List<DataRecord>();

    var powerOffset = 4;
    var minutesOffset = -5;
    var power = TryParseIntFromHex(value.Substring(value.Length - powerOffset));

    var dataRecordId = Guid.NewGuid();
    results.Add(new DataRecord()
    {
        Id = dataRecordId,
        CreatedAt = DateTime.Now,
        DeviceId = deviceId,
        MeasurementDate = createdAt,
        RecordValues = new List<RecordValue>()
        {
            new RecordValue()
            {
                Id = Guid.Empty,
                DataRecordId = dataRecordId,
                Value = power.ToString(),
                ValueTypeId = valueTypes.First(x => x.Type == "int").Id,
                ValueNameId = deviceValuesConfigurations.FirstOrDefault(x => x.ValueName == "Живлення")?.Id ?? Guid.Empty
            }
        }
    });

    if (value.Contains(Constants.UltrasoundSensorInitialValue1) || value.Contains(Constants.UltrasoundSensorInitialValue2))
    {
        return results;
    }
    else
    {
        var chunkSize = 3;
        var valueWithouPower = value.Substring(startIndex: 0, length: value.Length - powerOffset);
        var recordCouner = 6;

        for (var i = 0; i < valueWithouPower.Length; i += chunkSize)
        {
            var hexValue = valueWithouPower.Substring(startIndex: i, length: chunkSize);
            var parsedValue = TryParseIntFromHex(hexValue);
            var measurementDate = createdAt.AddMinutes(recordCouner * minutesOffset);

            dataRecordId = Guid.NewGuid();
            results.Add(new DataRecord()
            {
                Id = dataRecordId,
                CreatedAt = DateTime.Now,
                DeviceId = deviceId,
                MeasurementDate = measurementDate,
                RecordValues = new List<RecordValue>()
                {
                    new RecordValue()
                    {
                        Id = Guid.Empty,
                        DataRecordId = dataRecordId,
                        Value = parsedValue.ToString(),
                        ValueTypeId = valueTypes.First(x => x.Type == "int").Id,
                        ValueNameId = deviceValuesConfigurations.FirstOrDefault(x => x.ValueName == "Відстань")?.Id ?? Guid.Empty
                    }
                }
            });

            recordCouner--;
        }
    }

    return results;
}
```

Рисунок 2.10 – Код алгоритму конвертації значень ультразвукового сенсору

```
1 usage  Volodymyr Panasiuk
private static List<DataRecord> MapWaterQualitySensorToDataRecords(Guid deviceId, string value, DateTime createdAt, List<DeviceValuesConfiguration> deviceValuesConfigurations,
    List<ValueType> valueTypes)
{
    var dataRecordId = Guid.NewGuid();
    var results = new List<DataRecord>()
    {
        new DataRecord()
        {
            Id = dataRecordId,
            CreatedAt = DateTime.Now,
            DeviceId = deviceId,
            MeasurementDate = createdAt,
            RecordValues = new List<RecordValue>()
        }
    };

    var intValue = TryParseIntFromHex(value);
    if (intValue is > 0 and <= 400)
    {
        var temperature :double = (double)intValue / 10;
        results[0].RecordValues?.Add( item: new RecordValue()
        {
            Id = Guid.NewGuid(),
            DataRecordId = dataRecordId,
            Value = temperature.ToString(),
            ValueTypeId = valueTypes.First(x :ValueType => x.Type == "double").Id,
            ValueNameId = deviceValuesConfigurations.FirstOrDefault(x :DeviceValuesConfiguration => x.ValueName == "Температура")?.Id ?? Guid.Empty
        });
    }
    else if (intValue is > 400 and <= 600)
    {
        var ph :double = (double)(intValue - 400) / 10;
        results[0].RecordValues?.Add( item: new RecordValue()
        {
            Id = Guid.NewGuid(),
            DataRecordId = dataRecordId,
            Value = ph.ToString(),
            ValueTypeId = valueTypes.First(x :ValueType => x.Type == "double").Id,
            ValueNameId = deviceValuesConfigurations.FirstOrDefault(x :DeviceValuesConfiguration => x.ValueName == "Ph")?.Id ?? Guid.Empty
        });
    }
    else if (intValue is > 600 and <= 700)
    {
        var turb :int = intValue - 600;
        results[0].RecordValues?.Add( item: new RecordValue()
        {
            Id = Guid.NewGuid(),
            DataRecordId = dataRecordId,
            Value = turb.ToString(),
            ValueTypeId = valueTypes.First(x :ValueType => x.Type == "int").Id,
            ValueNameId = deviceValuesConfigurations.FirstOrDefault(x :DeviceValuesConfiguration => x.ValueName == "Каламутність")?.Id ?? Guid.Empty
        });
    }

    return results;
}

3 usages  Volodymyr Panasiuk
private static int TryParseIntFromHex(string value)
{
    try
    {
        return Convert.ToInt32(value, fromBase: 16);
    }
    catch (Exception e)
    {
        return 0;
    }
}
```

Рисунок 2.11 - Код алгоритму конвертації значень пристрою для вимірювання якості поверхневих вод

Клієнтський додаток взаємодіє з сервером через RESTful API, відправляючи запити на отримання, додавання, оновлення або видалення даних. Сервер обробляє запити та повертає відповіді у форматі JSON. Клієнтський додаток використовує отримані дані для відображення інформації про стан водойм, аналізу даних та формування звітів.

## 2.4 Розробка клієнтської частини

Клієнтська частина IoT-системи моніторингу стану природних водойм є інтерфейсом, через який користувачі взаємодіють з системою. Вона забезпечує відображення даних моніторингу, інструменти для їх аналізу та візуалізації, а також можливість налаштування сповіщень та інших функцій.

Для розробки клієнтської частини було обрано наступний технологічний стек:

– Бібліотека/Фреймворк: React 17;
– Мова програмування: TypeScript;
– Бібліотека компонентів: Material UI;
– Пакет для роботи з Api: Axios.

Клієнтська частина побудована за компонентною архітектурою, що дозволяє розділити інтерфейс на окремі незалежні блоки, які можна легко повторно використовувати та тестувати. Кожен компонент відповідає за відображення певної частини інтерфейсу та обробку відповідних подій.

Основні компоненти клієнтської частини.

HomePage: Головна сторінка додатку, яка відображає інформацію про проект, а також його архітектуру (рис. 2.12).

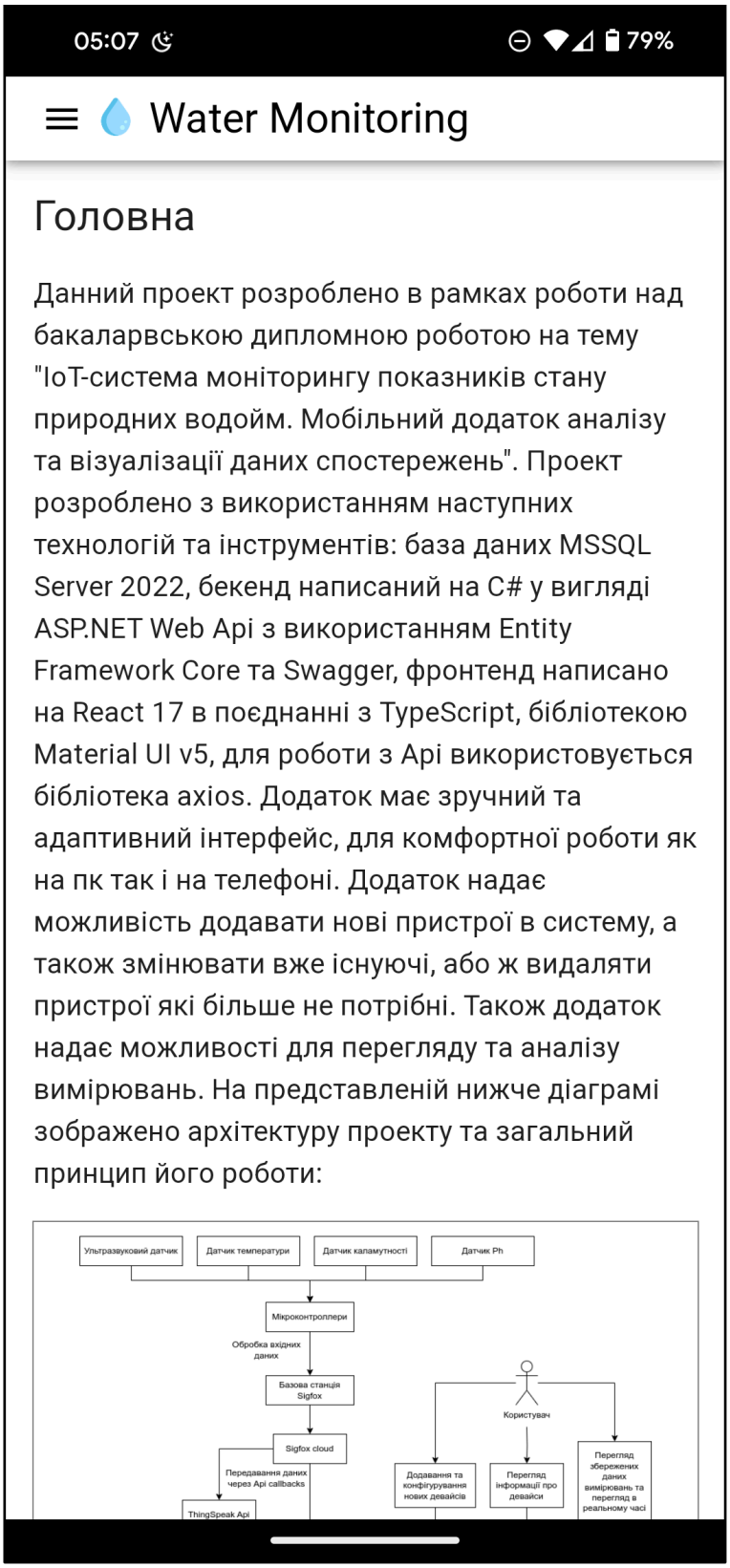
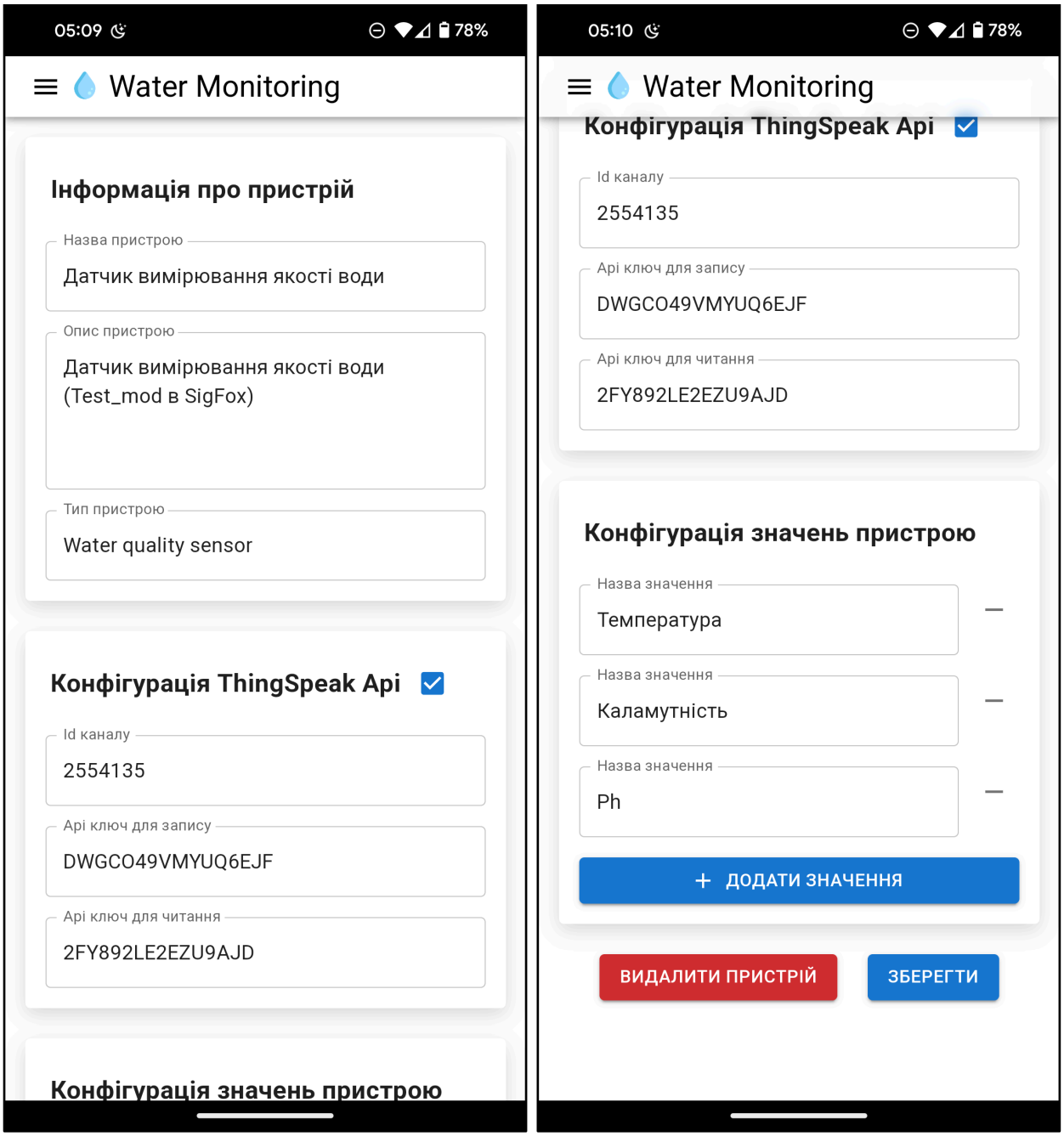


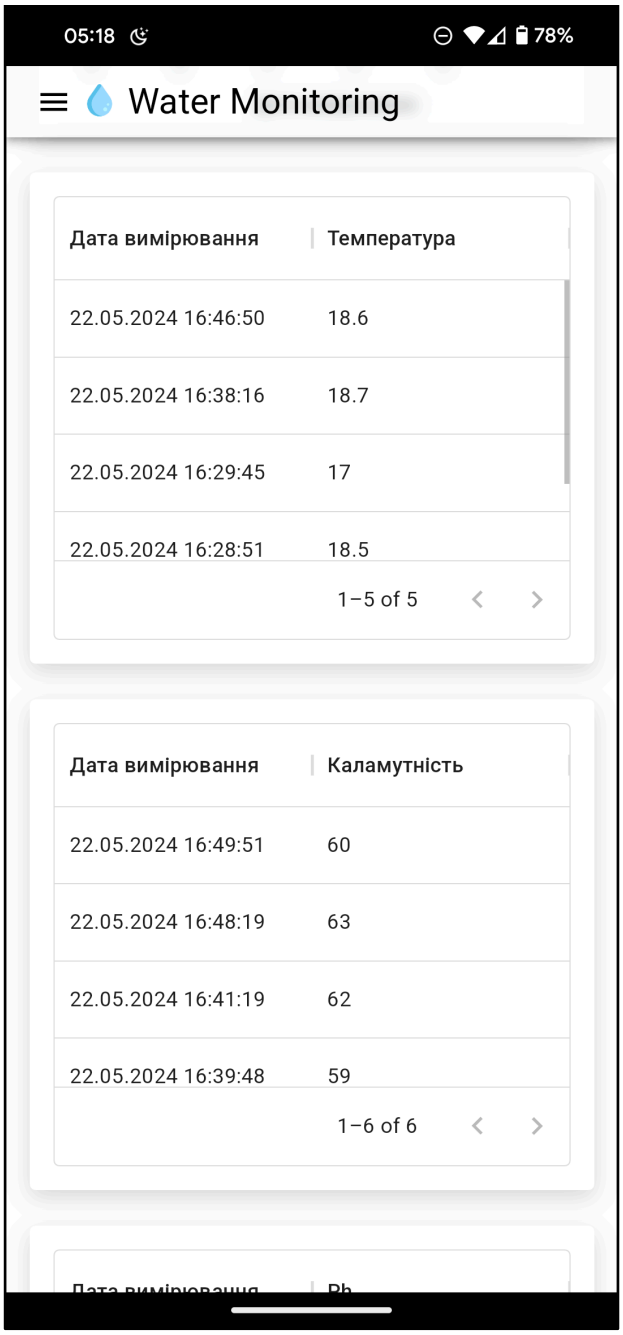

Рисунок 2.12 - Головна сторінка додатку

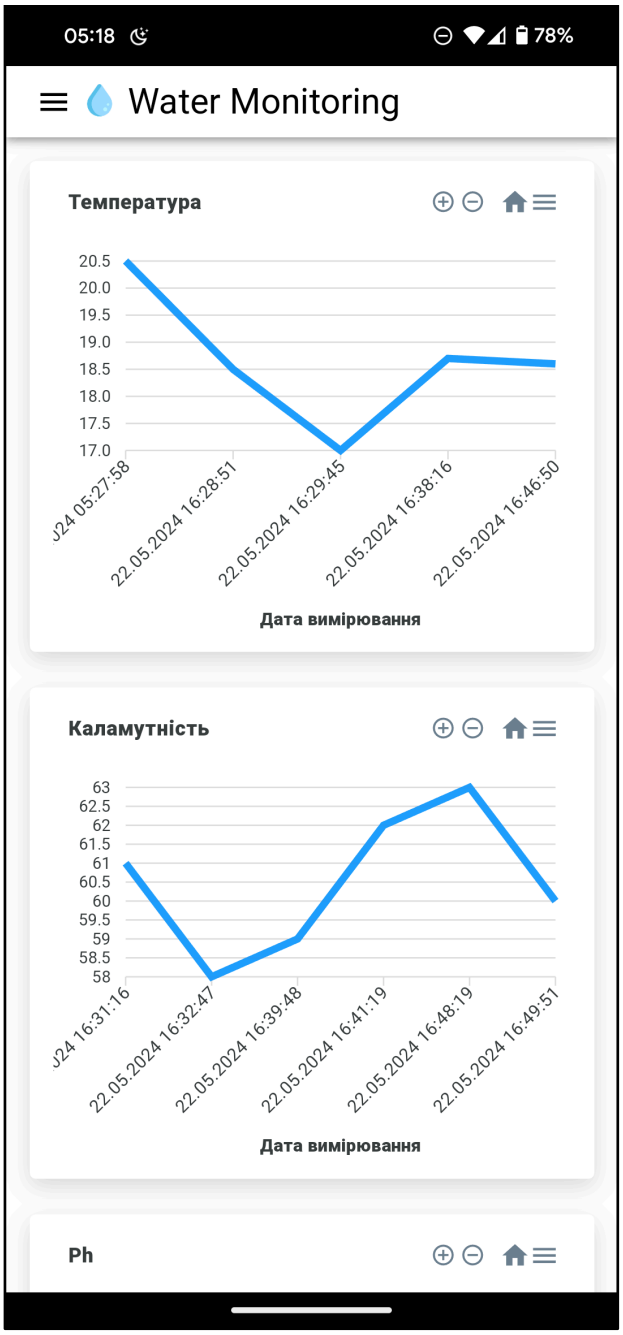
DevicePage: Сторінка для перегляду інформації про пристрій, її зміни, конфігурації Api ключів для роботи з ThingSpeak та конфігурації значень пристрою, а також видалення та додавання пристроїв (рис 2.13 - 2.14).




Рисунки 2.13 - 2.14 – Сторінка для додавання, редагування та видалення пристроїв

DeviceDataPage: Сторінка для перегляду оброблених результатів вимірювань, у вигляді графіків та таблиць, з можливостями для експорту цих даних у вигляді зображення або CSV файлів (рис. 2.15-2.17).




Рисунки 2.15 - 2.16 – Представлення результатів вимірювань у вигляді графіків та таблиць

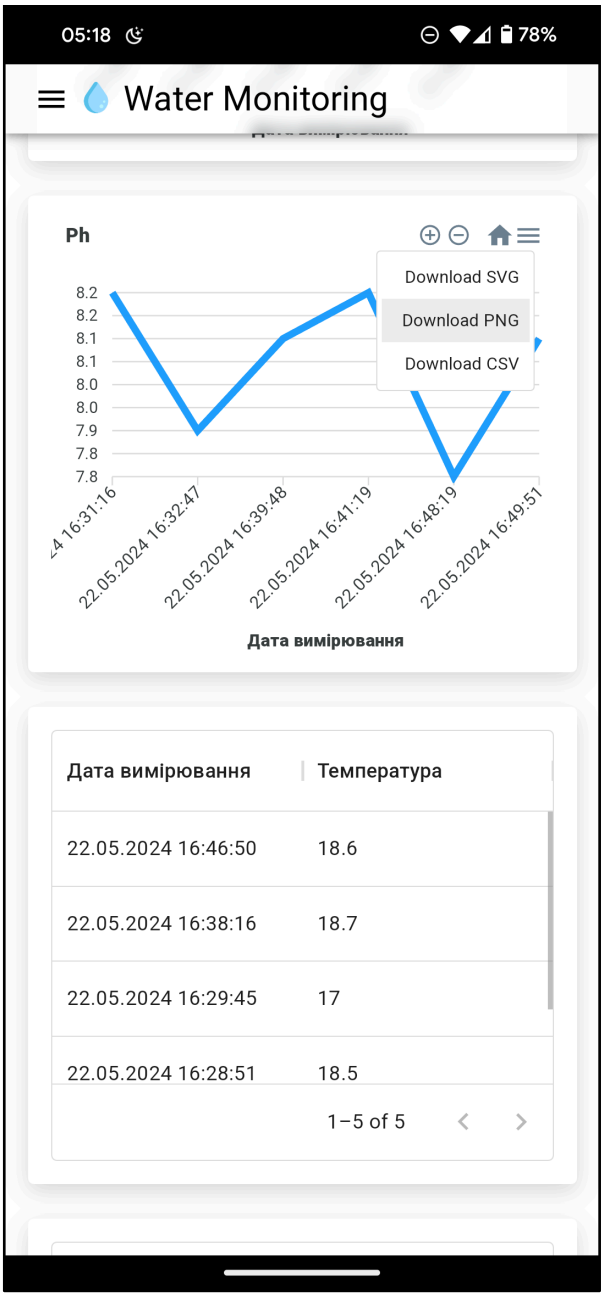



Рисунок 2.17 – Можливість експорту даних у вигляді зображень, або CSV файлів

NavigationComponent та RootLayout: Компоненти необхідний для навігації по додактку (рис. 2.18).

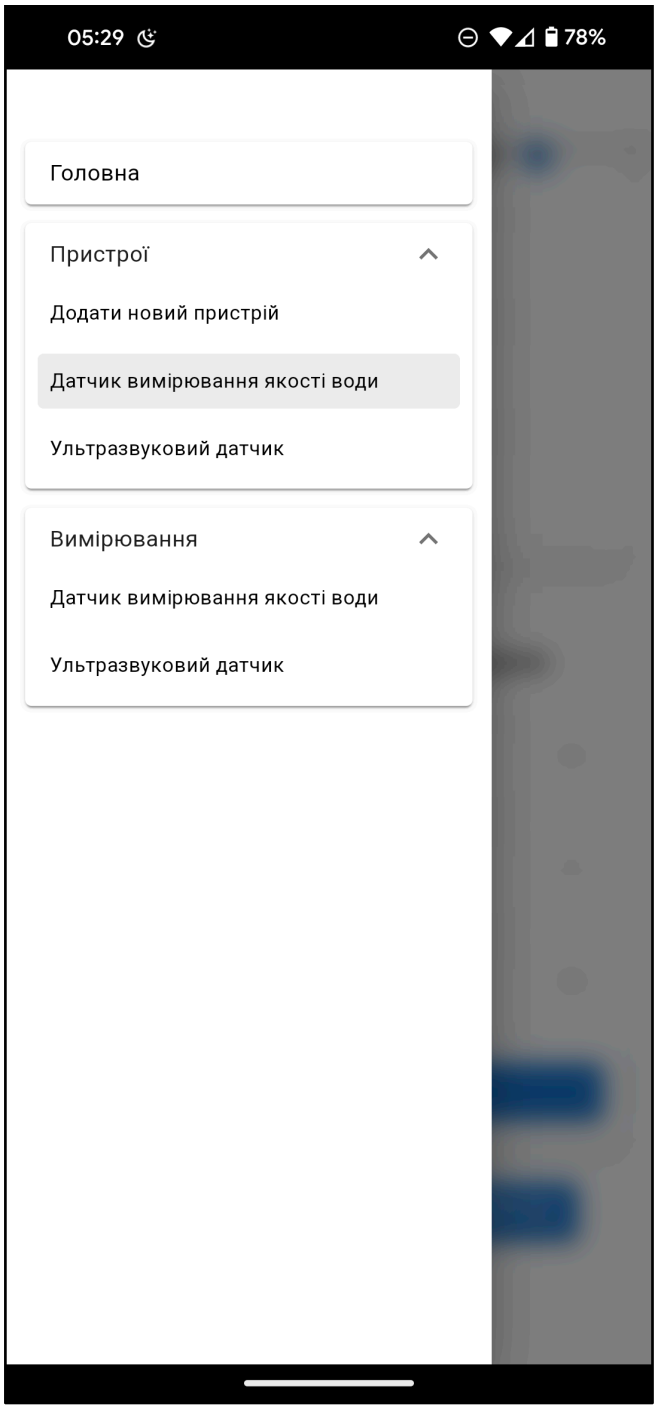



Рисунок 2.18 – Навігація по додатку

Функціональні можливості клієнтської частини:

– Відображення даних: Відображення даних моніторингу у вигляді таблиць та графіків;

– Аналіз даних: Можливість фільтрації, сортування та групування даних за різними критеріями, а також порівняння даних з різних періодів часу;

– Експорт даних: Можливість експорту даних у різні формати (CSV, PNG, SVG) для подальшого аналізу;

– Робота з пристроями: Можливість додавання нових пристроїв, їх конфігурування та видалення.

Клієнтська частина забезпечує інтуїтивно зрозумілий та зручний інтерфейс для взаємодії з користувачем. Користувач може вибирати пристрої зі списків, переглядати детальну інформацію про них та налаштовувати параметри відображення даних.

Клієнтська частина розроблена з використанням React 17, TypeScript та Material UI, тому її дизайн є адаптивним, що дозволяє додатку, працювати на різних пристроях та платформах (комп'ютери, планшети, смартфони) з різними розмірами екранів та операційними системами (рис. 2.19 - 2.20).

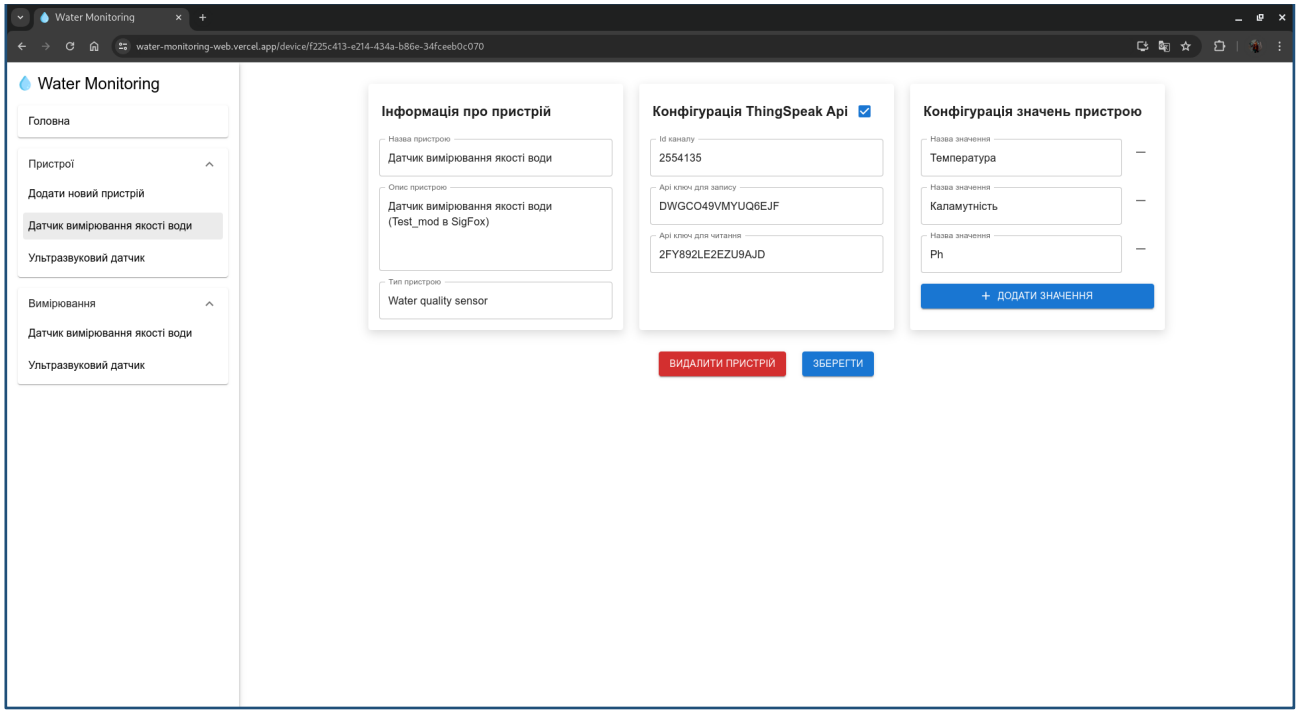



Рисунок 2.19 – Зовнішній вигляд сторінки для додавання, редагування та видалення пристроїв на комп’ютері

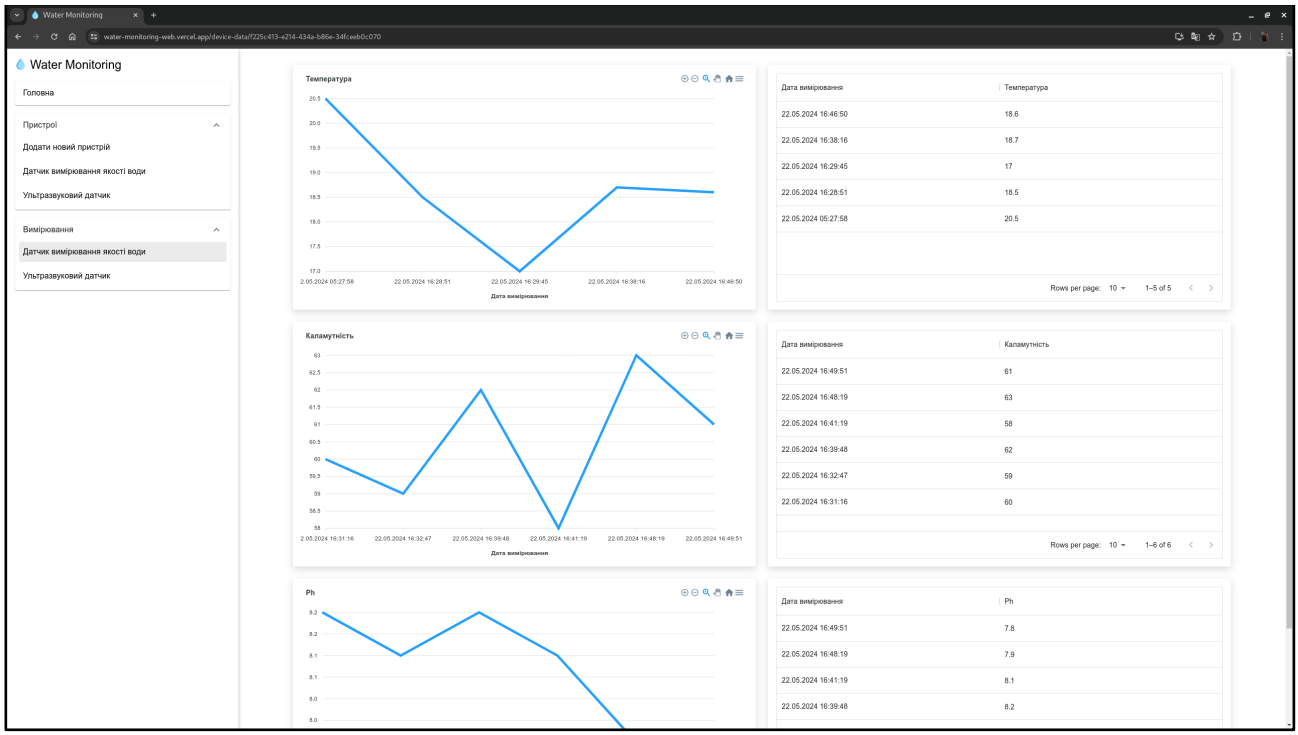



Рисунок 2.20 – Зовнішній вигляд сторінки для перегляду результатів вимірювань у вигляді графіків та таблиць на комп’ютері

## 2.5 Розгортання та хостинг додатку

Успішна реалізація IoT-системи моніторингу стану природних водойм включає не лише розробку програмного забезпечення, а й його ефективне розгортання та підтримку у робочому стані. Для забезпечення безперервної інтеграції та безперервного розгортання (CI/CD) мобільного додатку було обрано комплекс сервісів та інструментів, які дозволяють автоматизувати процеси розробки, тестування та розгортання, забезпечуючи швидке та надійне оновлення додатку.

Основою для керування версіями коду та спільної роботи над проектом є розподілена система контролю версій Git [19]. Git дозволяє відстежувати зміни в коді, створювати гілки для розробки нових функцій, об'єднувати зміни з різних гілок та відновлювати попередні версії коду у разі потреби. GitHub, як веб-сервіс, надає зручний інтерфейс для роботи з Git-репозиторіями (рис. 2.21), а також додаткові інструменти для спільної роботи, такі як пул-реквести, issues та project

boards [20]. Пул-реквести дозволяють розробникам пропонувати зміни до коду та обговорювати їх з іншими членами команди. Issues використовуються для відстеження завдань та помилок, а project boards - для візуалізації процесу розробки та планування роботи.

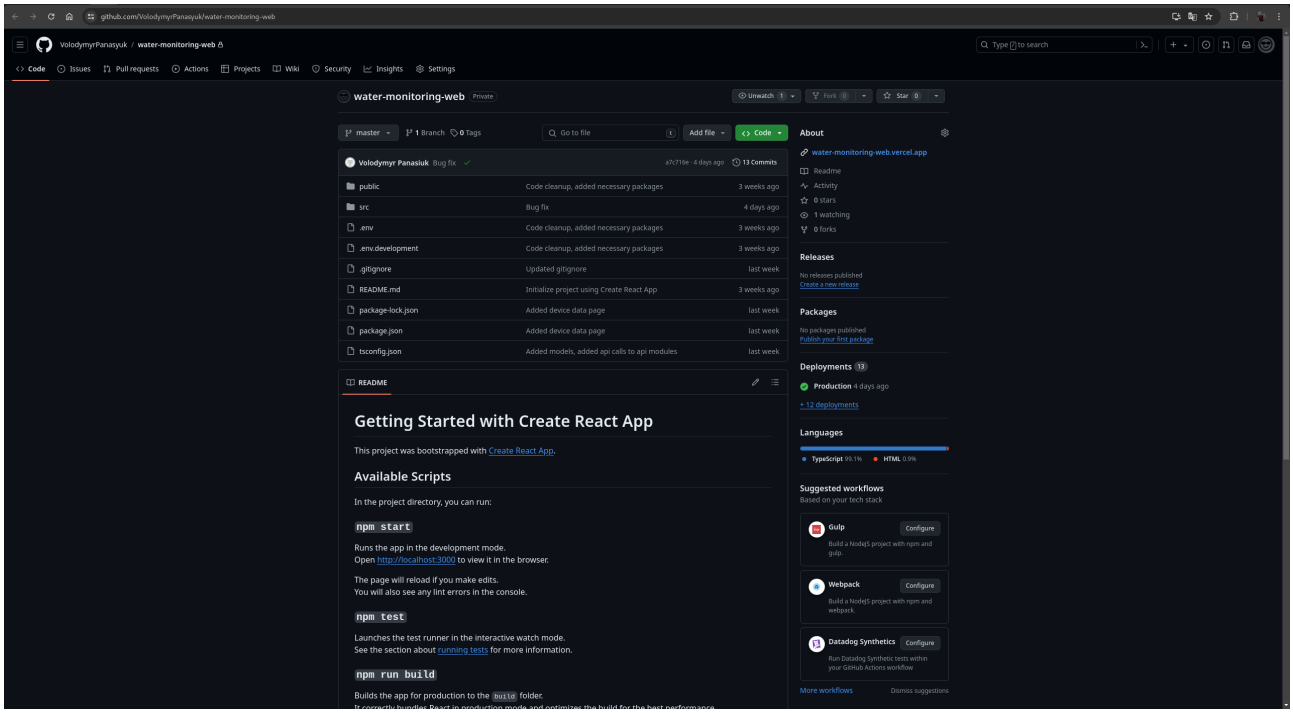



Рисунок 2.21 – Github репозиторій клієнтської частини застосунку

Для хостингу бекенду (API) було обрано платформу Railway [21]. Railway - це хмарна платформа, що надає інфраструктуру та інструменти для розгортання та управління веб-додатками. Вона підтримує різні мови програмування та фреймворки, включаючи .NET, що дозволяє легко розгорнути бекенд, розроблений на C# та .NET Web API. Railway також надає інструменти для моніторингу та масштабування додатків, що забезпечує їх надійну та ефективну роботу. Крім того, Railway інтегрується з GitHub, що дозволяє автоматично розгортати нові версії бекенду при внесенні змін до коду у репозиторії (рис. 2.22).

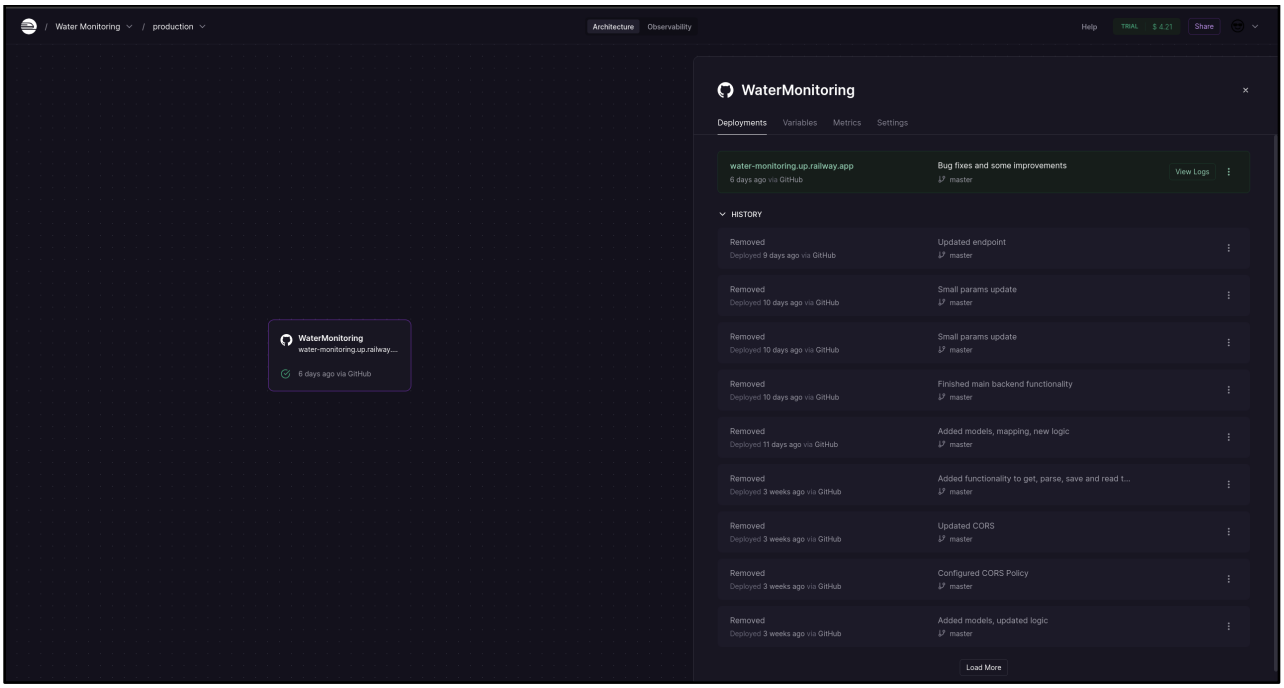



Рисунок 2.22 – Панель керування сервісу Railway

Для хостингу фронтенду (клієнтської частини) було обрано платформу Vercel [22]. Vercel - це хмарна платформа, спеціально розроблена для розгортання та хостингу статичних веб-сайтів та фронтенд-додатків. Вона підтримує різні фреймворки, включаючи React, що дозволяє легко розгорнути фронтенд, розроблений на React 17, TypeScript та Material UI. Vercel також надає інструменти для оптимізації продуктивності та кешування, що забезпечує швидке завантаження та роботу додатку. Vercel також інтегрується з GitHub, що дозволяє автоматично розгортати нові версії фронтенду при внесенні змін до коду у репозиторії (рис. 2.23).

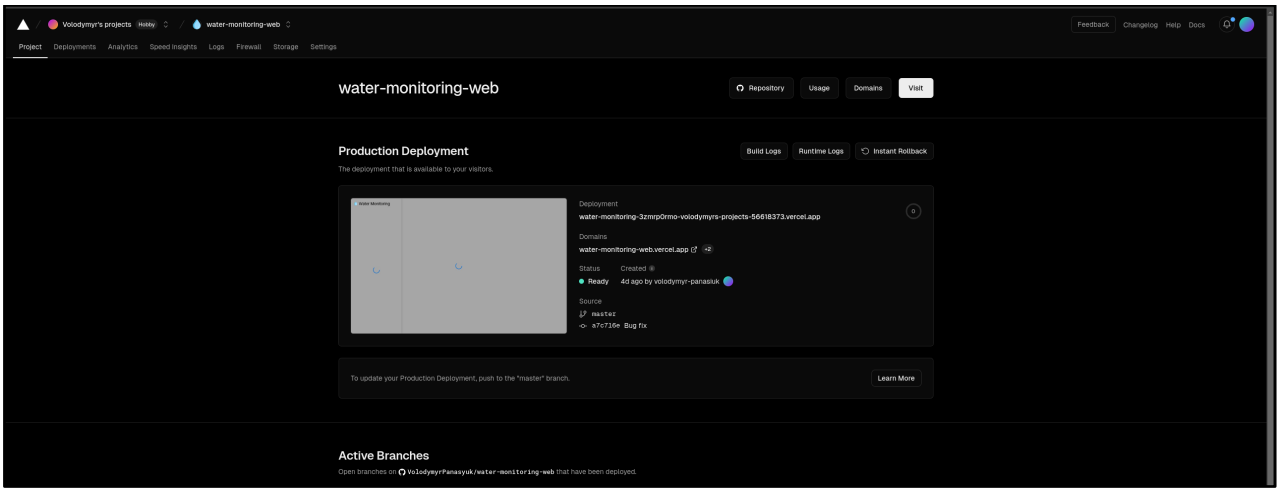



Рисунок 2.23 – Панель керування сервісу Vercel

Для хостингу бази даних MSSQL Server 2022 було обрано сервіс Somee [23]. Somee - це безкоштовний хмарний хостинг, що надає доступ до бази даних MSSQL Server. Він підтримує підключення з різних платформ та мов програмування, включаючи .NET та Entity Framework. Somee також надає інструменти для управління базою даних та резервного копіювання, що забезпечує безпеку та збереження даних (рис. 2.24).

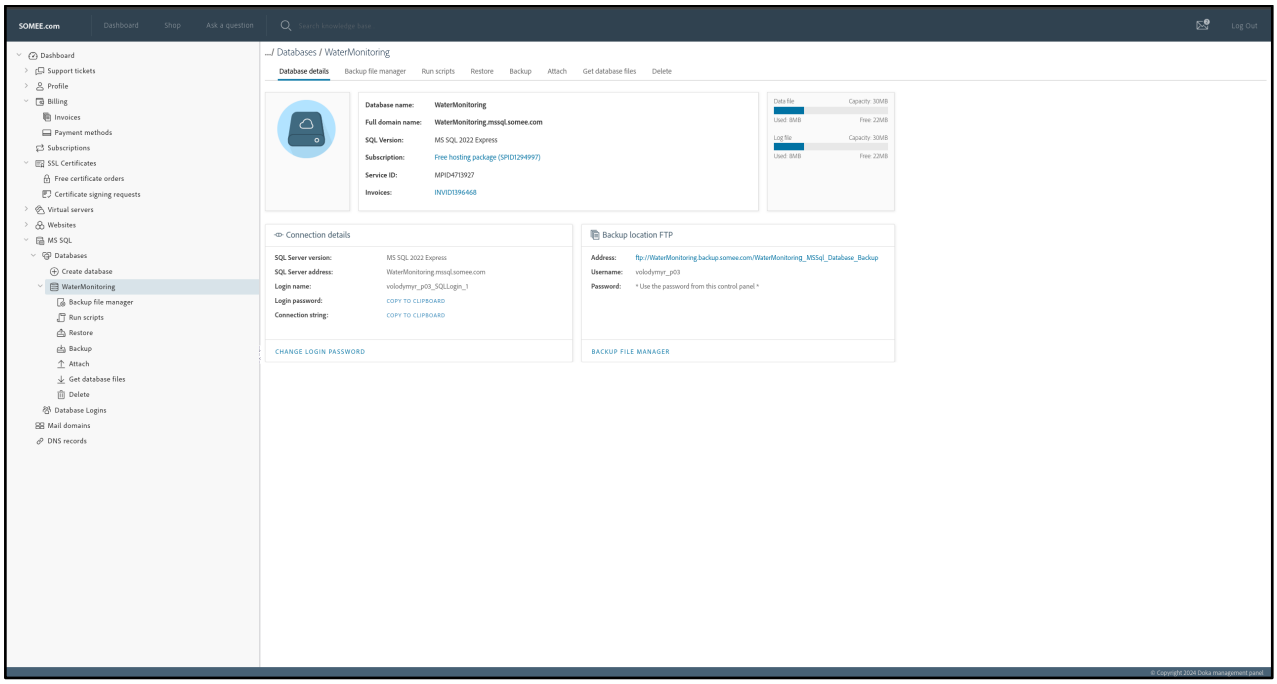



Рисунок 2.24 – Панель керування сервісу Somee

Процес CI/CD складається з наступних етапів:

1. Розробник вносить зміни до коду та відправляє їх у Git-репозиторій на GitHub;
2. GitHub Actions (або інший CI/CD сервіс) автоматично запускає процес збирання та тестування коду;
3. У разі успішного проходження тестів, код автоматично розгортається на серверах Railway (бекенд) та Vercel (фронтенд);
4. База даних на Somee оновлюється, якщо це необхідно;
5. Користувачі отримують доступ до оновленої версії додатку.

Такий підхід дозволяє забезпечити швидке та надійне розгортання нових версій додатку, мінімізувати ризик виникнення помилок та прискорити процес розробки. Автоматизація процесів CI/CD звільняє розробників від рутинних завдань та дозволяє їм зосередитися на розробці нових функцій та покращенні якості продукту.

# 3 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

## 3.1 Проведення експерементального дослідження і збору даних

Для перевірки роботи та оцінки ефективності розробленої IoT-системи, 22 травня 2024 року, разом з Гавриловим Г.А., Конотопом Б.П., та Проценком Д.П., було проведено експериментальне дослідження якості води у поверхневій водоймі (рис. 3.1), що знаходиться на території ВНТУ (рис. 3.2). Вимірювалися такі параметри, як температура, pH та каламутність. Дані збиралися за допомогою IoT-пристрою для вимірювання якості поверхневих вод (рис. 3.3), після чого дані передавалися через базову станцію Sigfox на Sigfox Cloud, після чого за допомогою Sigfox Api Callback передавалися в розроблену систему.

Рисунок 3.1 – Проведення експерементального дослідження

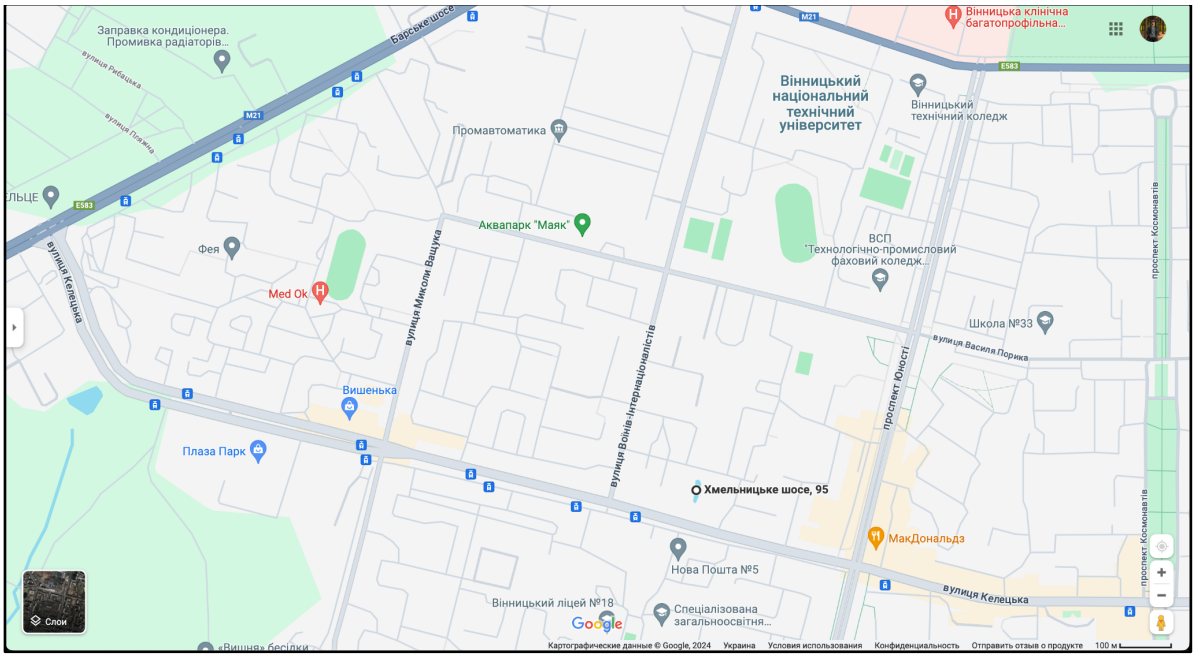


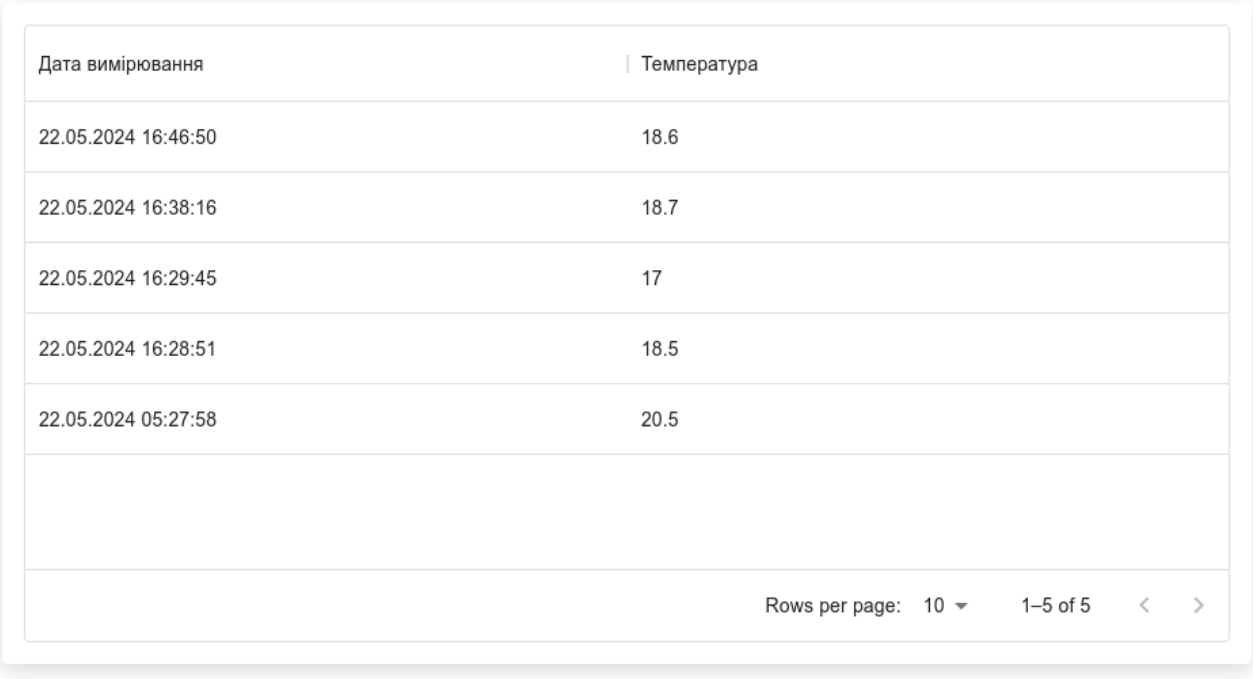

Рисунок 3.2 – Місце проведення експеременту

Рисунок 3.3 – IoT-пристрій для вимірювання якості поверхневих вод

## 3.2 Аналіз та візуалізація даних у мобільному додатку

Отримані від IoT-пристрою для моніторингу якості поверхневих вод, дані у форматі HEX-кодів пройшли конвертацію у відповідні одиниці вимірювання (°C для температури, pH для кислотності та NTU для каламутності). Після цього дані були очищені від можливих помилок та аномалій, а також агреговані за часовими інтервалами для спрощення подальшого аналізу та візуалізації.

Аналіз даних показав, що коливання температури води протягом експерименту були незначними, в межах 17-20,5 °C. Середнє значення температури становило 18.66 °C. Це свідчить про відносно стабільний термічний режим водойми протягом досліджуваного періоду (рис. 3.4).



| Дата вимірювання | Температура |
|---|---|
| 22.05.2024 16:46:50 | 18.6 |
| 22.05.2024 16:38:16 | 18.7 |
| 22.05.2024 16:29:45 | 17 |
| 22.05.2024 16:28:51 | 18.5 |
| 22.05.2024 05:27:58 | 20.5 |

Rows per page: 10 1–5 of 5

Рисунок 3.4 – Таблиця зміни температури води протягом вимірювання

Для наочного представлення динаміки зміни температури води протягом експерименту у мобільному додатку було побудовано графік (рис. 3.5).

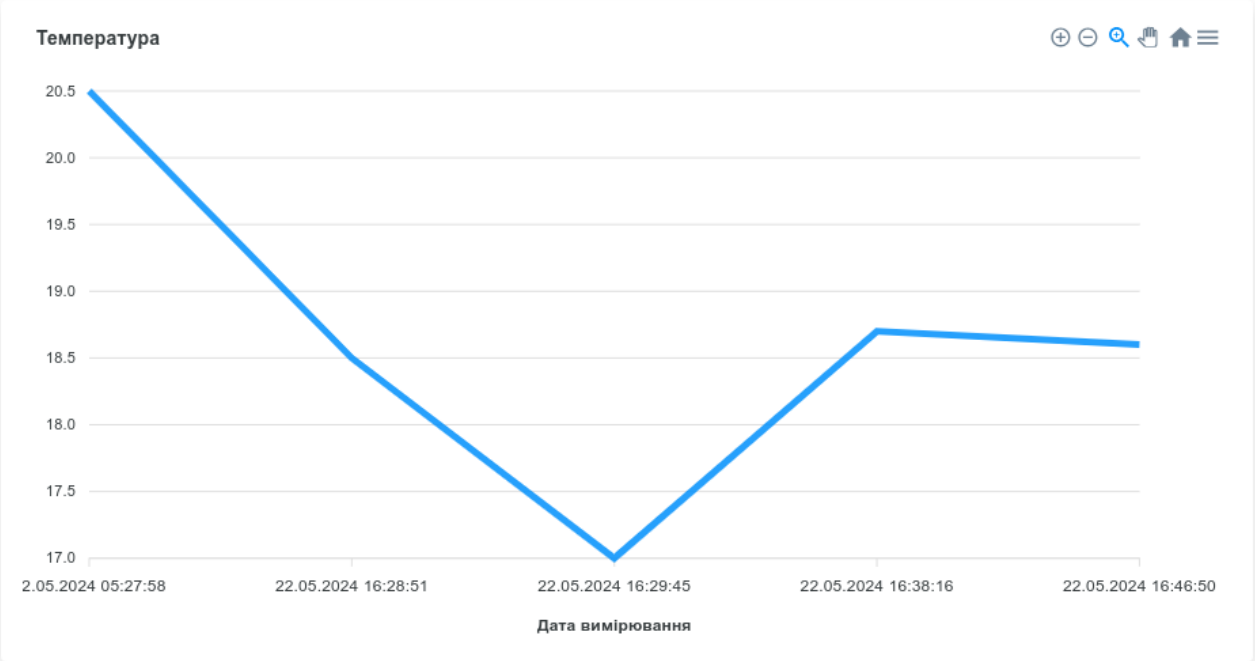



Рисунок 3.5 – Графік зміни температури води протягом вимірювання

Значення pH незначно коливались протягом експерименту, залишаючись в межах 7.8-8.2, з середнім значенням 8.05. Це свідчить про стабільний кислотно-лужний баланс води, що є важливим показником для підтримки життєдіяльності водних організмів. Отримані значення pH відповідають нормі (6.5 - 8.5) для більшості прісноводних водойм (рис. 3.6).

| Дата вимірювання | Ph |
|---|---|
| 22.05.2024 16:49:51 | 8.1 |
| 22.05.2024 16:48:19 | 7.8 |
| 22.05.2024 16:41:19 | 8.2 |
| 22.05.2024 16:39:48 | 8.1 |
| 22.05.2024 16:32:47 | 7.9 |
| 22.05.2024 16:31:16 | 8.2 |

Rows per page: 10 1–6 of 6

Рисунок 3.6 - Таблиця зміни pH води протягом вимірювання

Для наочного представлення динаміки зміни pH води протягом експерименту у мобільному додатку було побудовано графік (рис. 3.7).

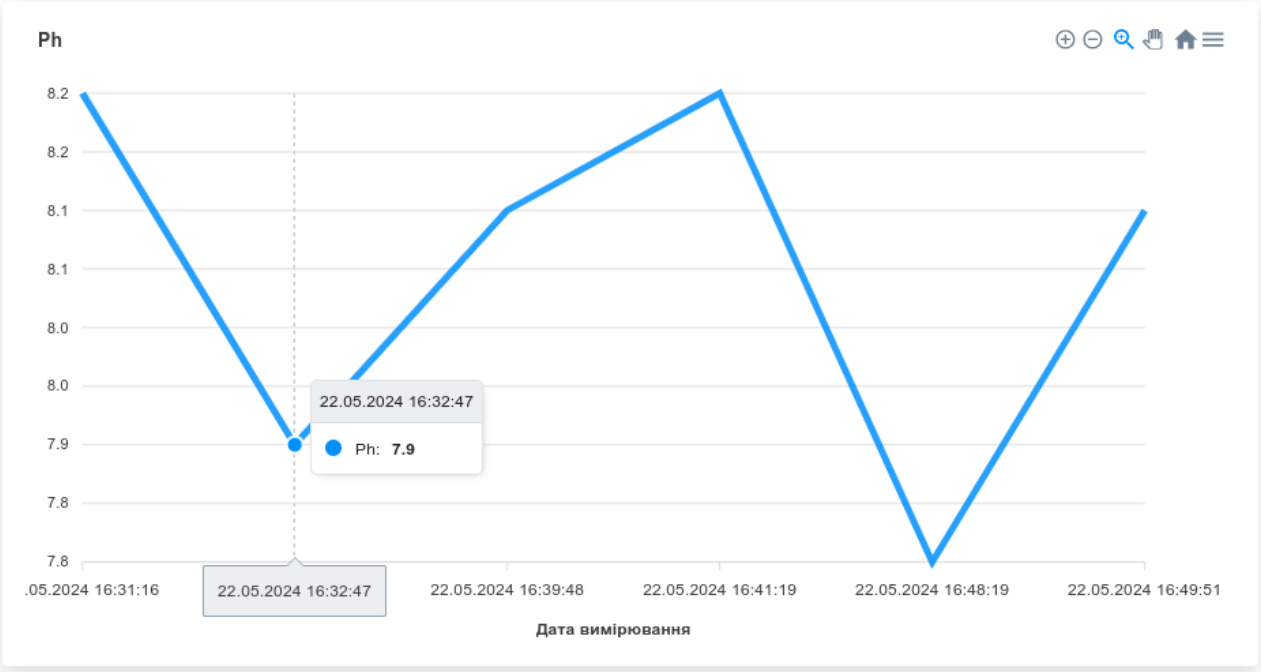



Рисунок 3.7 - Графік зміни pH води протягом вимірювання

Рівень каламутності води незначно коливався протягом вимірювання, залишаючись у межах 58-63 NTU. Середнє значення каламутності становить 60.6 NTU. Це свідчить про наявність певної кількості завислих частинок у воді, що може бути пов'язано з природними процесами (наприклад, стоком з берегів) або антропогенним впливом (рис. 3.8).

| Дата вимірювання | Каламутність |
|---|---|
| 22.05.2024 16:49:51 | 60 |
| 22.05.2024 16:48:19 | 63 |
| 22.05.2024 16:41:19 | 62 |
| 22.05.2024 16:39:48 | 59 |
| 22.05.2024 16:32:47 | 58 |
| 22.05.2024 16:31:16 | 61 |

Rows per page: 10 1–6 of 6

Рисунок 3.8 - Таблиця зміни рівня каламутності води протягом вимірювання

Для наочного представлення динаміки зміни каламутності води протягом експерименту у мобільному додатку було побудовано графік (рис. 3.9).

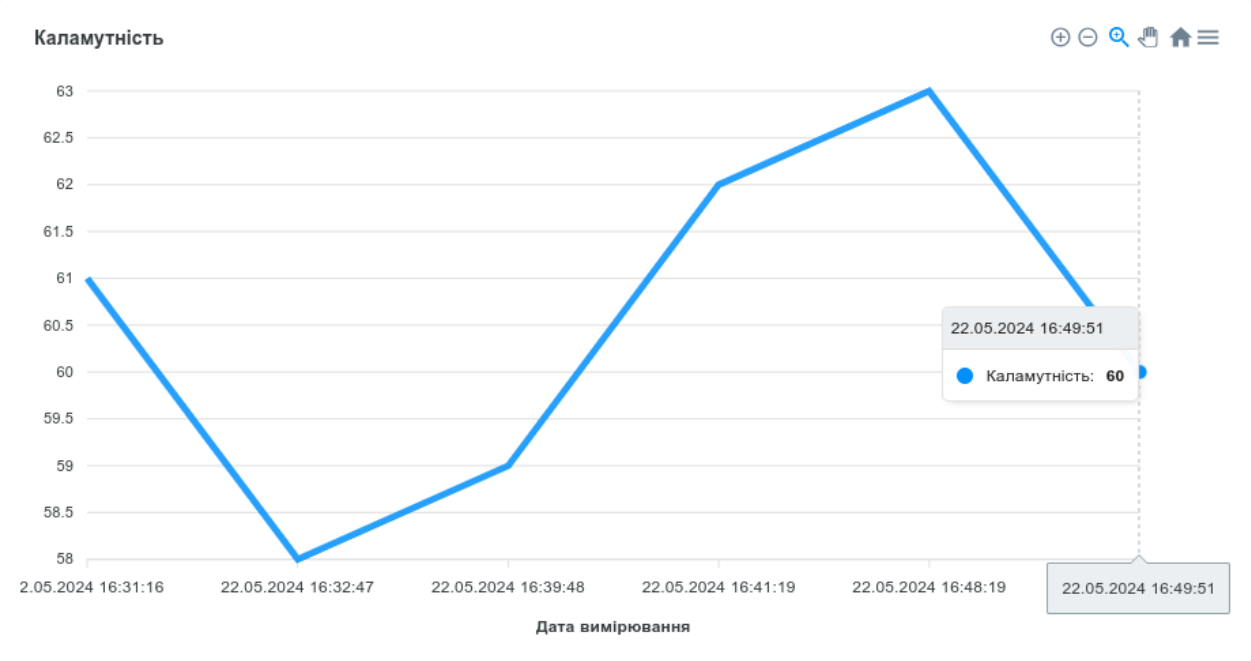



Рисунок 3.9 - Графік зміни рівня каламутності води протягом вимірювання

Також в рамках тестування розробленої IoT-системи було додано ще один пристрій, а саме ультразвуковий датчик для вимірювання відстані, проведено

його конфігурування в розробленій системі, а також протестовано працездатність. Цей пристрій може використовуватися для моніторингу рівня води. Було отримано значення відстані в см., а також значення живлення в мВ (рис 3.10-3.13).

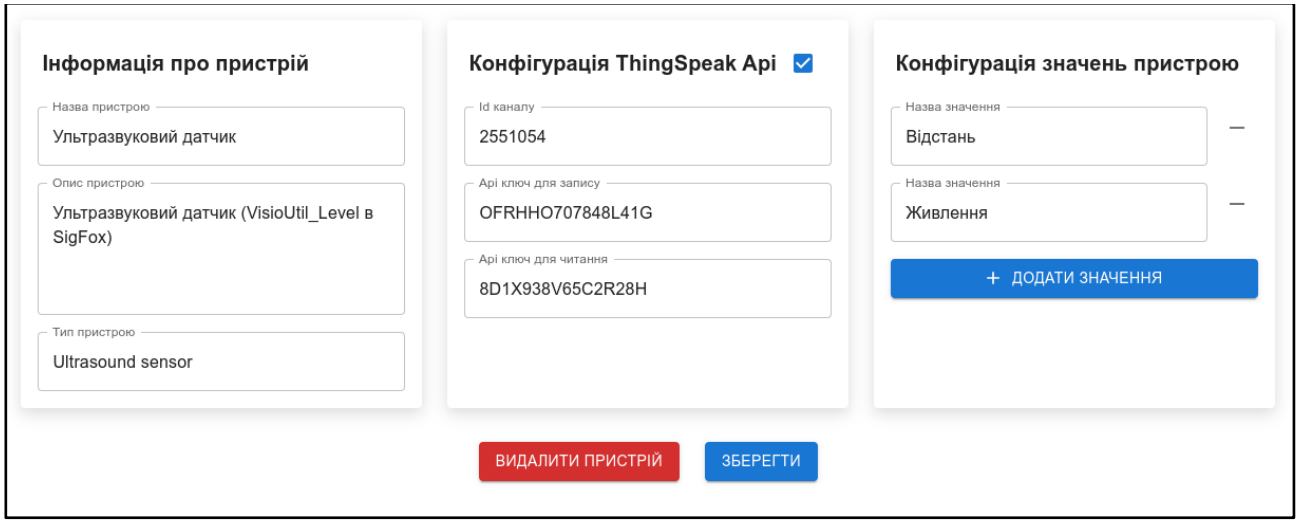



Рисунок 3.10 – Конфігурування ультразвукового датчику

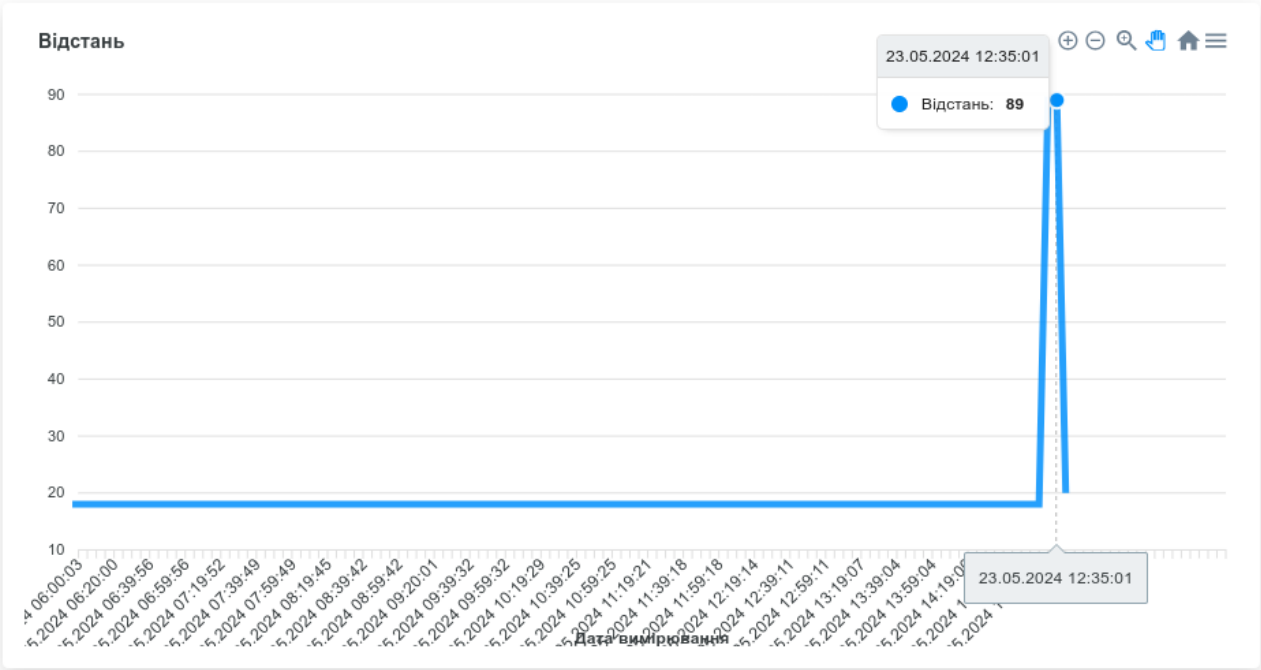



Рисунок 3.11 – Графік зміни відстані протягом вимірювання


Рисунок 3.12 – Графік напруги батереї живлення в мВ протягом вимірювання

## 3.3 Оцінка ефективності розробленої системи для аналізу та візуалізації даних

Оцінка ефективності розробленої IoT-системи моніторингу та мобільного додатку для аналізу та візуалізації даних є важливим етапом дослідження, який дозволяє визначити, наскільки успішно система виконує свої функції та відповідає поставленим вимогам. Для оцінки ефективності було проведено комплексний аналіз, що включав кількісні та якісні методи дослідження.

Результати експериментального дослідження, проведеного 22 травня 2024 року, підтвердили ефективність розробленої системи моніторингу якості води у поверхневих водоймах. Зібрані дані щодо температури, pH та каламутності води свідчать про успішну роботу IoT-пристроїв та коректність передачі інформації.

Зафіксовані значення температури (17-20,5 °C) та pH (7.8-8.2) знаходяться в межах типових для поверхневих вод і не викликають занепокоєння [24]. Стабільність цих показників протягом експерименту свідчить про відсутність різких змін умов середовища, що могли б негативно вплинути на водні організми.

Рівень каламутності води (58-63 NTU) вказує на наявність завислих частинок у воді. Це може бути пов'язано з природними процесами або антропогенним впливом. Для більш детальної оцінки необхідно провести додаткові дослідження та врахувати інші параметри якості води, а також зібрати дані за триваліший період.

Для об'єктивної оцінки ефективності розробленої системи було застосовано ряд кількісних методів, які дозволяють виміряти конкретні параметри роботи системи та порівняти їх з еталонними значеннями або вимогами.

Швидкість передачі даних: Вимірювання часу, необхідного для передачі даних з сенсорів на сервер та їх обробки. Для цього використовувалися інструменти моніторингу мережевого трафіку та логи сервера. Результати вимірювань показали, що час передачі та обробки даних є мінімальним, що забезпечує оперативність отримання інформації про стан водойм та можливість швидкого реагування на зміни.

Час відгуку додатку: Вимірювання часу, необхідного для завантаження сторінок додатку та виконання операцій з даними. Для цього використовувалися інструменти веб-аналітики та спеціалізоване програмне забезпечення для тестування продуктивності веб-додатків. Результати тестування показали, що мобільний додаток демонструє високу швидкість роботи та миттєвий відгук на дії користувача, що забезпечує комфортну та ефективну взаємодію з системою.

Обсяг збережених даних: Оцінка обсягу даних, що зберігаються в базі даних, та ефективності їх стиснення. Для цього використовувалися інструменти моніторингу бази даних та аналіз розміру файлів бази даних. Результати аналізу показали, що завдяки використанню ефективних алгоритмів стиснення даних, обсяг збережених даних є мінімальним, що дозволяє зберігати велику історію вимірювань без значного збільшення вимог до дискового простору.

Проведене дослідження підтвердило ефективність розробленої IoT-системи моніторингу та мобільного додатку для аналізу та візуалізації даних.

# ВИСНОВКИ

У ході виконання бакалаврської дипломної роботи було досягнуто поставленої мети – покращено процес моніторингу та аналізу стану природних водойм завдяки розробці масштабованого мобільного додатку, який дозволив автоматизувати збір даних про якість води в режимі реального часу та забезпечив їх оперативну візуалізацію і аналіз.

В результаті аналізу існуючих рішень для моніторингу стану природних водойм було виявлено їх переваги та недоліки. Традиційні методи моніторингу, хоч і є точними, але вимагають значних витрат часу та ресурсів. IoT-системи дозволяють автоматизувати процес збору даних та отримувати їх у режимі реального часу, що значно підвищує ефективність моніторингу.

Розроблено концепцію мобільного додатку для аналізу та візуалізації даних спостережень. Обрано оптимальні мову програмування, середовище розробки та дизайн додатку. Розроблений мобільний додаток надає користувачам зручний доступ до даних моніторингу, дозволяючи переглядати їх у вигляді таблиць та графіків, аналізувати дані за різними критеріями, налаштовувати сповіщення та експортувати дані у різні формати.

Проведені експериментальні дослідження підтвердили ефективність розробленої системи. Мобільний додаток забезпечує швидкий та зручний доступ до даних моніторингу, а також потужні інструменти для їх аналізу та візуалізації.

Розроблена IoT-система моніторингу та мобільний додаток можуть бути використані для оперативного контролю стану природних водойм, своєчасного виявлення проблем та прийняття ефективних рішень щодо їх усунення. Система може бути корисною для екологічних організацій, державних установ, наукових досліджень та інших зацікавлених сторін.

В подальшому планується розширення функціональних можливостей системи, таких як прогнозування змін стану водойм, інтеграція з іншими джерелами даних (метеорологічні дані, дані дистанційного зондування Землі), а також розробка додаткових інструментів для аналізу та візуалізації даних.

Робота виконувалась на замовлення науково-дослідної лабораторії автоматизованих систем управління в енергетиці та транспорті (НДЛ АСУЕТ) кафедри САІТ Вінницького національного технічного університету.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гончаренко Д. В., Мокін В. Б., Проценко Д. П., Переваги технологій Інтернету речей Sigfox для створення локальної системи моніторингу атмосферного повітря 2023. [Електронний ресурс] Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2023/paper/view/17211/14669

2. Хільчевський В.К., Гребінь В.В. Основи гідрохімії. К.: Ніка-Центр, 2012. 312 с.

3. Водний кодекс України // Відомості Верховної Ради України (ВВР), 1995, № 24, ст.189

4. Закон України «Про охорону навколишнього природного середовища» // Відомості Верховної Ради України (ВВР), 1991, № 41, ст.546

5. Постанова Кабінету Міністрів України від 19 вересня 1997 р. № 1076 «Про затвердження Положення про державний моніторинг навколишнього природного середовища»

6. Кравченко К.С. Сучасні методи та засоби моніторингу водних об'єктів // Вісник Національного університету водного господарства та природокористування. - 2020. - № 1 (89). - С. 85-92.

7. Костюченко О.В. Інтернет речей в екологічному моніторингу // Інформаційні технології та комп'ютерна інженерія. - 2019. - № 2 (58). - С. 118-123.

8. Гавриленко О.А., Кузьменко В.І. Автоматизована система моніторингу якості води на основі Інтернету речей // Вісник Житомирського державного технологічного університету. Серія: Технічні науки. - 2021. - № 1 (95). - С. 113-120.

9. Міщенко О.В. Мобільні додатки як інструмент екологічної освіти та просвітництва // Інформаційні технології і засоби навчання. - 2021. - № 4 (76). - С. 113-128.

10. Марченко О.В. Розробка мобільних додатків для платформ iOS та Android: порівняльний аналіз // Наукові записки Тернопільського національного

педагогічного університету імені Володимира Гнатюка. Серія: Комп'ютерні технології. - 2020. - № 1 (4). - С. 56-63.

11. Ткаченко В.В. Кросплатформна розробка мобільних додатків: огляд фреймворків та інструментів // Вісник Національного університету "Львівська політехніка". Серія: Інформаційні системи та мережі. - 2023. - № 954. - С. 124-132.

12. Кузьменко В.І. Прогресивні веб-додатки: новий етап розвитку веб-технологій // Інформаційні технології та комп'ютерна інженерія. - 2022. - № 1 (63). - С. 78-85.

13. Коваленко О.В. Гібридні мобільні додатки: переваги та недоліки // Вісник Національного технічного університету України "Київський політехнічний інститут". Серія: Інформатика, управління та обчислювальна техніка. - 2019. - № 1 (64). - С. 89-96.

14. Albahari, J., & Albahari, B. (2022). C# 10 in a Nutshell: The Definitive Reference. O'Reilly Media. (p. 1-5, 15-20, 250-260)

15. Lerman, J. (2021). Programming Entity Framework: Building Data Centric Applications with the Microsoft ADO.NET Entity Framework. O'Reilly Media. (p. 50-75, 150-175, 250-275)

16. Ben-Gan, I. (2022). Microsoft SQL Server 2022 Revealed: Including Big Data Clusters and Machine Learning. Apress. (p. 80-100, 180-200, 300-320)

17. Bank, C. (2022). React: Up & Running: Building Web Applications. O'Reilly Media. (p. 1-10, 40-50, 80-90)

18. Microsoft. (2022). What's new in SQL Server 2022. [Електронний ресурс] Режим доступу: https://docs.microsoft.com/en-us/sql/sql-server/what-s-new-in-sql-server-2022?view=sql-server-ver16

19. Chacon, S., & Straub, B. (2014). Pro Git (2nd ed.). Apress. (p. 1-30, 50-70, 100-120)

20. GitHub. (2023). GitHub Docs. [Електронний ресурс] Режим доступу: https://docs.github.com/

21. Railway. (2023). Railway Documentation. [Електронний ресурс] Режим доступу: https://docs.railway.app/

22. Vercel. (2023). Vercel Documentation. [Електронний ресурс] Режим доступу: https://vercel.com/docs

23. Somee. (2023). Somee Documentation. [Електронний ресурс] Режим доступу: https://somee.com/DOKA/Help/Article/5/Introduction

24. Відповідність якості води водним об’єктам, в яких водяться риби // Наказ Міністерства охорони навколишнього природного середовища України 07.07.2017 № 282

Додаток А

(обов’язковий)

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
“____”_____ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську дипломну роботу

ІОТ-СИСТЕМА МОНІТОРИНГУ ПОКАЗНИКІВ СТАНУ ПРИРОДНИХ ВОДОЙМ. МОБІЛЬНИЙ ДОДАТОК АНАЛІЗУ ТА ВІЗУАЛІЗАЦІЇ ДАНИХ СПОСТЕРЕЖЕНЬ

08-34.БДР.010.00.000 ТЗ

Керівник: к.т.н., доц.
_____ Дмитро ПРОЦЕНКО
«__» _____ 2024 р.
Розробив студент гр. 2ІСТ-20б
_____ Володимир ПАНАСЮК
«__» _____ 2024 р.

Вінниця 2024

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2024р., та індивідуальне завдання на БДР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2024р.

2. Джерела розробки:

1) Д.В. Гончаренко, В.Б. Мокін. Д.П. Проценко, Г.В. Горячев, І.В. Варчук Іот-система вимірювання параметрів стану вод на базі sigfox. 2024 ВНТУ URL: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/41813/20077.pdf

3. Мета і призначення роботи.

Метою даної роботи є розробка IoT-системи моніторингу показників стану природних водойм, яка дозволить автоматизувати процес збору даних про якість води та забезпечити їх оперативну візуалізацію та аналіз.

4. Вихідні дані для проведення робіт:

1) Хмарні сервіси IoT: Sigfox Cloud, ThingSpeak.

2) Дані зібрані з пристрою для вимірювання якості поверхневих вод.

5. Методи дослідження:

Використання платформи Sigfox Backend, Thing Speak, Somee, Railway, Vercel та Github.

6. Етапи роботи і терміни їх виконання:

a) Аналіз предметної області _____ – _____

b) Аналіз інформаційних систем для моніторингу стану природних водойм. Концепція мобільного додатку аналізу та візуалізації даних спостережень. _____ – _____

c) Вибір мови програмування та середовища розробки, та дизайну додатка. _____ – _____

d) Обробка результатів експерименту. _____ – _____

e) Оформлення матеріалів до захисту БДР _____ – _____

7. Очікувані результати та порядок реалізації розробка IoT-системи моніторингу показників стану природних водойм.

8. Вимоги до розробленої документації

Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей: 124 «Системний аналіз», 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології»)».

9. Порядок приймання роботи

Публічний захист «___» _____ 2024 р.

Початок розробки «___» _____ 2024 р.

Граничні терміни виконання БДР «___» _____ 2024 р.

Розробив студент групи 2ІСТ-20б _____ Володимир ПАНАСЮК

Додаток Б
(обов'язковий)

Протокол перевірки бакалаврської дипломної роботи на наявність текстових запозичень

Назва роботи: «IoT-система моніторингу показників стану природних водойм. Мобільний додаток аналізу та візуалізації даних спостережень»
Тип роботи: бакалаврська дипломна робота
Підрозділ: кафедра САІТ

**Показники звіту подібності Unicheck**

Оригінальність 94% Схожість 6%

Аналіз звіту подібності (відмітити потрібне)

• Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
o Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
o 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 _____ Сергій ЖУКОВ
(підпис)

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

Автор роботи _____ Володимир ПАНАСЮК
(підпис)

Керівник роботи _____ Дмитро ПРОЦЕНКО
(підпис)

Додаток В

(довідниковий)

Фрагмент лістингу програми

```
public static List<DataRecord> MapHexDeviceFieldToDataRecords(Guid deviceId, string deviceType, string value, DateTime createdAt, List<DeviceValuesConfiguration> deviceValuesConfigurations,
    List<ValueType> valueTypes)
{
    var results = new List<DataRecord>();

    if (string.IsNullOrEmpty(value))
    {
        return results;
    }

    try
    {
        switch (deviceType)
        {
            case Constants.UltrasoundSensorDeviceType:
                results = MapUltrasoundSensorDeviceFieldToDataRecords(deviceId, value, createdAt, deviceValuesConfigurations, valueTypes);
                break;
            case Constants.WaterQualitySensorDeviceType:
                results = MapWaterQualitySensorToDataRecords(deviceId, value, createdAt, deviceValuesConfigurations, valueTypes);
                break;
            default:
                var intValue = Convert.ToInt32(value, 16);
                var dataRecordId = Guid.NewGuid();
```

```
                results.Add(new DataRecord()
                {
                    Id = dataRecordId,
                    CreatedAt = DateTime.Now,
                    DeviceId = deviceId,
                    MeasurementDate = createdAt,
                    RecordValues = new List<RecordValue>()
                    {
                        new RecordValue()
                        {
                            Id = Guid.Empty,
                            DataRecordId = dataRecordId,
                            Value = intValue.ToString(),
                            ValueTypeId = valueTypes.First(x => x.Type == "int").Id,
                            ValueNameId = deviceValuesConfigurations.First().Id
                        }
                    }
                });
                break;
            }

            return results;
        }
        catch (Exception e)
        {
            return results;
        }
    }
```

```
        private static List<DataRecord> MapUltrasoundSensorDeviceFieldToDataRecords(Guid deviceId, string value, DateTime createdAt, List<DeviceValuesConfiguration> deviceValuesConfigurations,
            List<ValueType> valueTypes)
        {
            var results = new List<DataRecord>();


            var powerOffset = 4;
            var minutesOffset = -5;
            var power = TryParseIntFromHex(value.Substring(value.Length - powerOffset));


            var dataRecordId = Guid.NewGuid();
            results.Add(new DataRecord()
            {
                Id = dataRecordId,
                CreatedAt = DateTime.Now,
                DeviceId = deviceId,
                MeasurementDate = createdAt,
                RecordValues = new List<RecordValue>()
                {
                    new RecordValue()
                    {
                        Id = Guid.Empty,
                        DataRecordId = dataRecordId,
                        Value = power.ToString(),
                        ValueTypeId = valueTypes.First(x => x.Type == "int").Id,
                        ValueNameId = deviceValuesConfigurations.FirstOrDefault(x => x.ValueName == "Живлення")?.Id ?? Guid.Empty
                    }
                }
```

```
            });

            if (value.Contains(Constants.UltrasoundSensorInitialValue1) ||
value.Contains(Constants.UltrasoundSensorInitialValue2))
            {
                return results;
            }
            else
            {
                var chunkSize = 3;
                var valueWithouPower = value.Substring(0, value.Length -
powerOffset);
                var recordCouner = 6;

                for (var i = 0; i < valueWithouPower.Length; i += chunkSize)
                {
                    var hexValue = valueWithouPower.Substring(i, chunkSize);
                    var parsedValue = TryParseIntFromHex(hexValue);
                    var measurementDate = createdAt.AddMinutes(recordCouner *
minutesOffset);

                    dataRecordId = Guid.NewGuid();
                    results.Add(new DataRecord()
                    {
                        Id = dataRecordId,
                        CreatedAt = DateTime.Now,
                        DeviceId = deviceId,
                        MeasurementDate = measurementDate,
                        RecordValues = new List<RecordValue>()
                        {
```

```
                    new RecordValue()
                    {
                        Id = Guid.Empty,
                        DataRecordId = dataRecordId,
                        Value = parsedValue.ToString(),
                        ValueTypeId = valueTypes.First(x => x.Type == "int").Id,
                        ValueNameId = deviceValuesConfigurations.FirstOrDefault(x => x.ValueName == "Відстань")?.Id ?? Guid.Empty
                    }
                }
            });

            recordCouner--;
        }
    }

    return results;
}

private static List<DataRecord> MapWaterQualitySensorToDataRecords(Guid deviceId, string value, DateTime createdAt, List<DeviceValuesConfiguration> deviceValuesConfigurations,
    List<ValueType> valueTypes)
{
    var dataRecordId = Guid.NewGuid();
    var results = new List<DataRecord>()
    {
        new DataRecord()
        {
            Id = dataRecordId,
```

```
                CreatedAt = DateTime.Now,
                DeviceId = deviceId,
                MeasurementDate = createdAt,
                RecordValues = new List<RecordValue>()
            }
        };

        var intValue = TryParseIntFromHex(value);
        if (intValue is > 0 and <= 400)
        {
            var temperature = (double)intValue / 10;
            results[0].RecordValues?.Add(new RecordValue()
            {
                Id = Guid.NewGuid(),
                DataRecordId = dataRecordId,
                Value = temperature.ToString(),
                ValueTypeId = valueTypes.First(x => x.Type == "double").Id,
                ValueNameId = deviceValuesConfigurations.FirstOrDefault(x =>
x.ValueName == "Температура")?.Id ?? Guid.Empty
            });
        }
        else if (intValue is > 400 and <= 600)
        {
            var ph = (double)(intValue - 400) / 10;
            results[0].RecordValues?.Add(new RecordValue()
            {
                Id = Guid.NewGuid(),
                DataRecordId = dataRecordId,
                Value = ph.ToString(),
                ValueTypeId = valueTypes.First(x => x.Type == "double").Id,
```

```
                ValueNameId = deviceValuesConfigurations.FirstOrDefault(x => x.ValueName == "Ph")?.Id ?? Guid.Empty
            });
        }
        else if (intValue is > 600 and <= 700)
        {
            var turb = intValue - 600;
            results[0].RecordValues?.Add(new RecordValue()
            {
                Id = Guid.NewGuid(),
                DataRecordId = dataRecordId,
                Value = turb.ToString(),
                ValueTypeId = valueTypes.First(x => x.Type == "int").Id,
                ValueNameId = deviceValuesConfigurations.FirstOrDefault(x => x.ValueName == "Каламутність")?.Id ?? Guid.Empty
            });
        }

        return results;
    }

    private static int TryParseIntFromHex(string value)
    {
        try
        {
            return Convert.ToInt32(value, 16);
        }
        catch (Exception e)
        {
            return 0;
```

```
        }
    }

public class ThingSpeakHttpClient : IThingSpeakHttpClient
    {
        private readonly HttpClient _httpClient = new();

        public async Task<int> WriteChanelFeedAsync(WriteChanelFeedRequestDto model)
        {
            var url = ThingSpeakUrlHelper.BuildWriteChanelFeedUrl(model);
            var response = await _httpClient.GetAsync(url);

            response.EnsureSuccessStatusCode();

            var content = await response.Content.ReadAsStringAsync();
            return int.Parse(content);
        }

        public async Task<string> ReadChanelFeedAsync(ReadChanelFeedRequestDto model)
        {
            var url = ThingSpeakUrlHelper.BuildReadChannelFeedUrl(model);
            var response = await _httpClient.GetAsync(url);

            response.EnsureSuccessStatusCode();

            return await response.Content.ReadAsStringAsync();
        }
```

```
        public async Task<string> ReadChanelFieldAsync(ReadChanelFieldRequestDto model)
        {
            var url = ThingSpeakUrlHelper.BuildReadChannelFieldUrl(model);
            var response = await _httpClient.GetAsync(url);

            response.EnsureSuccessStatusCode();

            return await response.Content.ReadAsStringAsync();
        }
    }
```

Додаток Г

(обов’язковий)

**ІЛЮСТРАТИВНА ЧАСТИНА**

ІОТ-СИСТЕМА МОНІТОРИНГУ ПОКАЗНИКІВ СТАНУ ПРИРОДНИХ ВОДОЙМ. МОБІЛЬНИЙ ДОДАТОК АНАЛІЗУ ТА ВІЗУАЛІЗАЦІЇ ДАНИХ СПОСТЕРЕЖЕНЬ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2024 р.

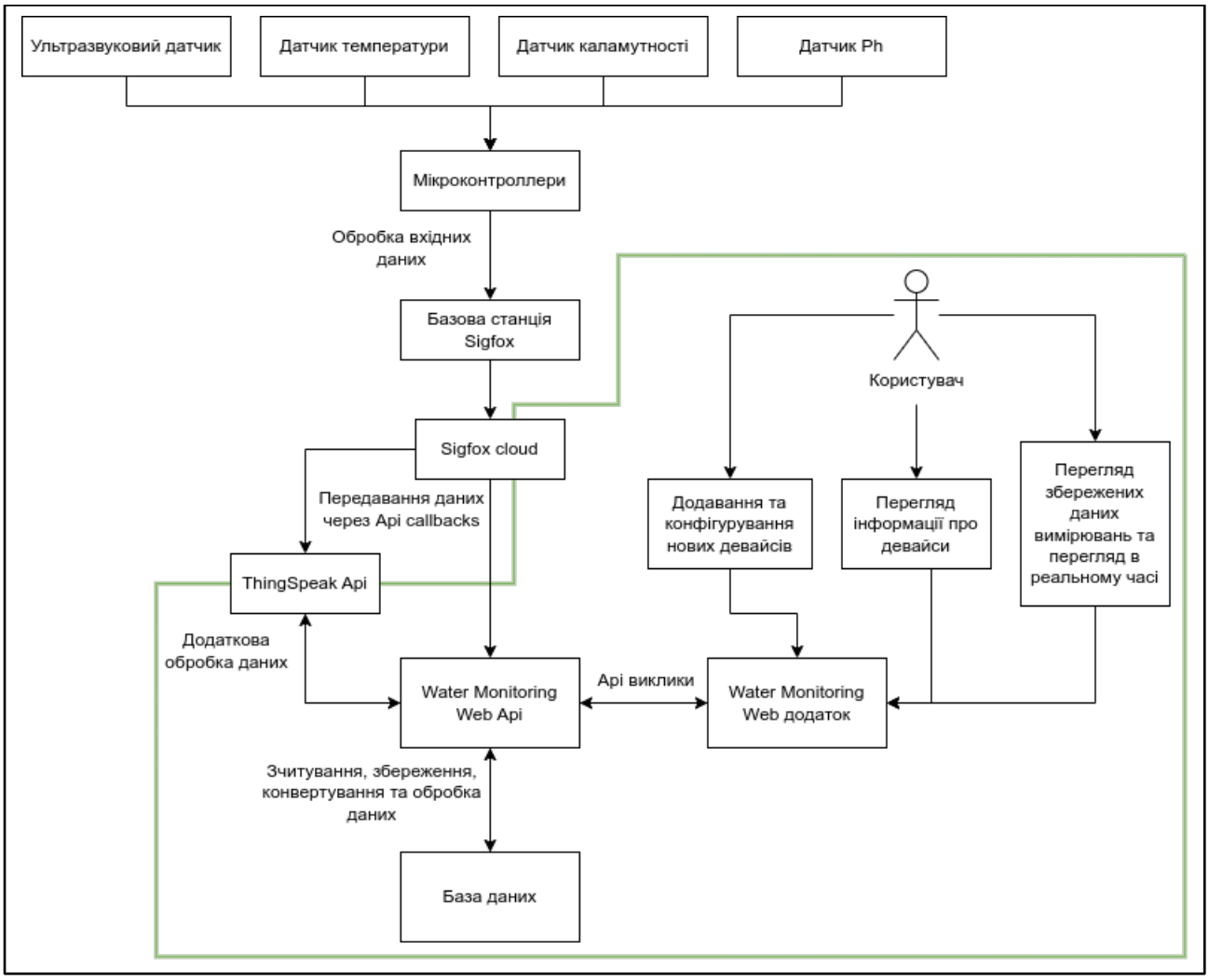



Рисунок Г.1 – Архітектура IoT системи моніторингу стану природних водойм

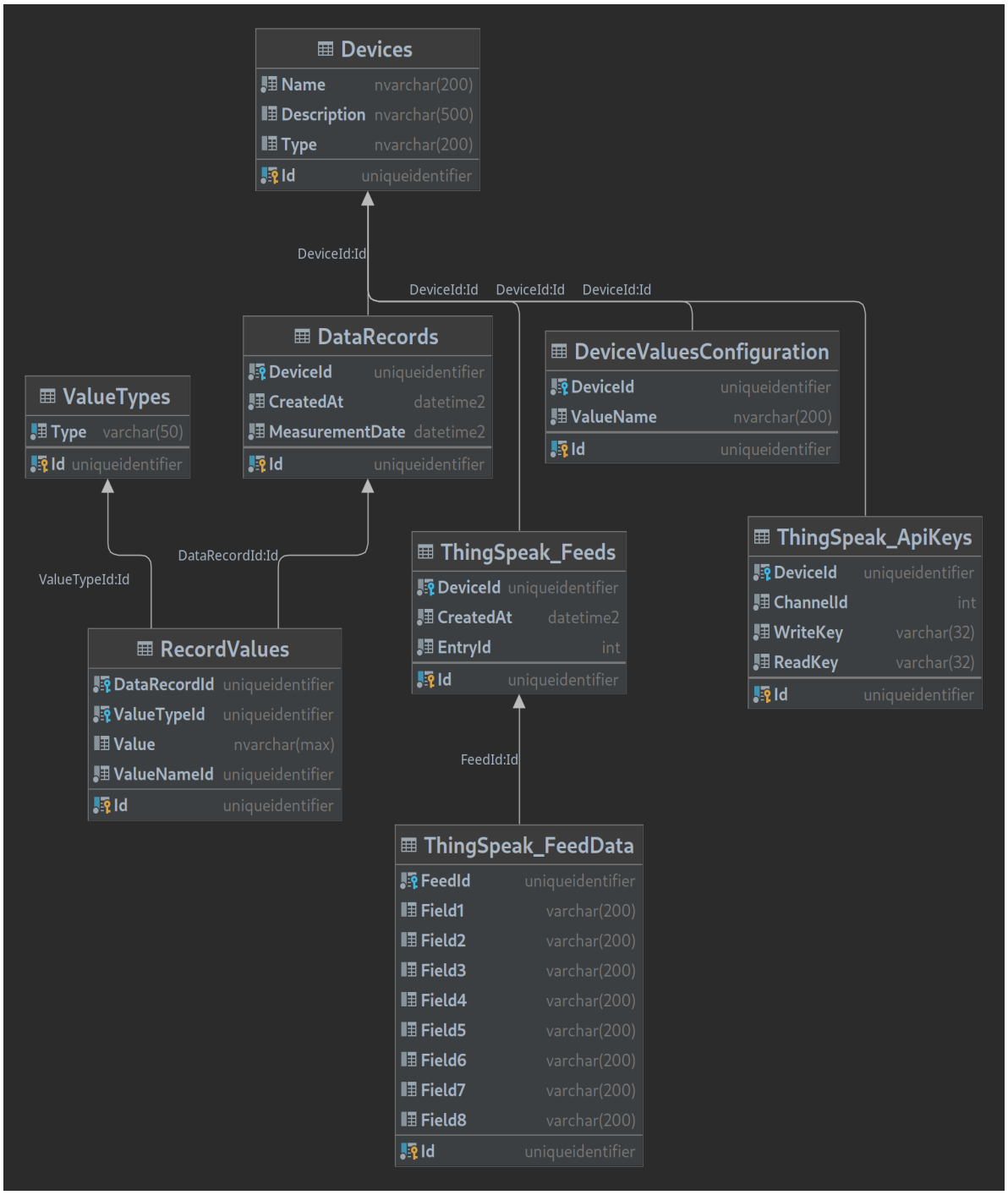



Рисунок Г.2 – Модель даних IoT системи моніторингу стану природних водойм

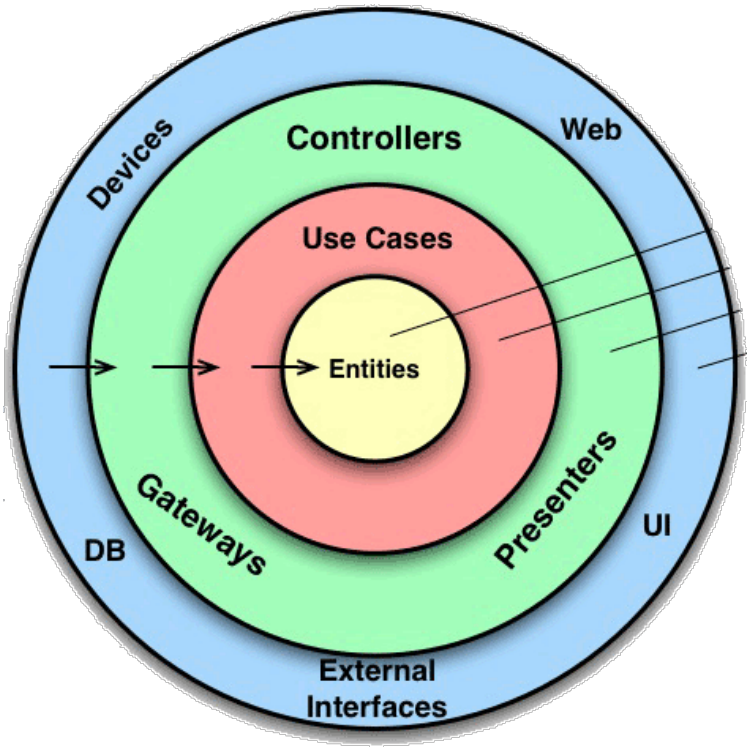



Рисунок Г.3 – Архітектура ASP.NET Core Web APIs

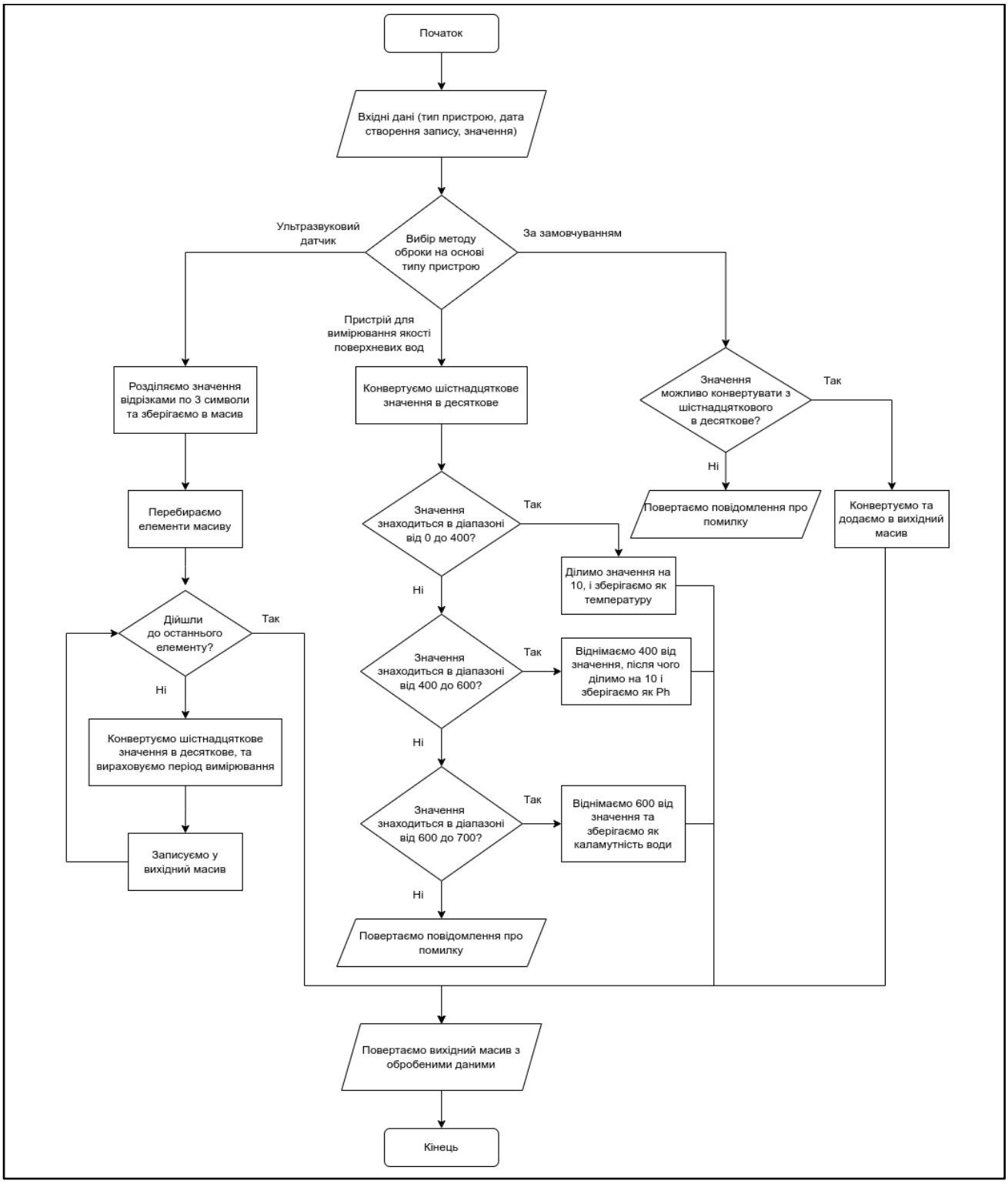



Рисунок Г.4 – Алгоритм конвертування даних

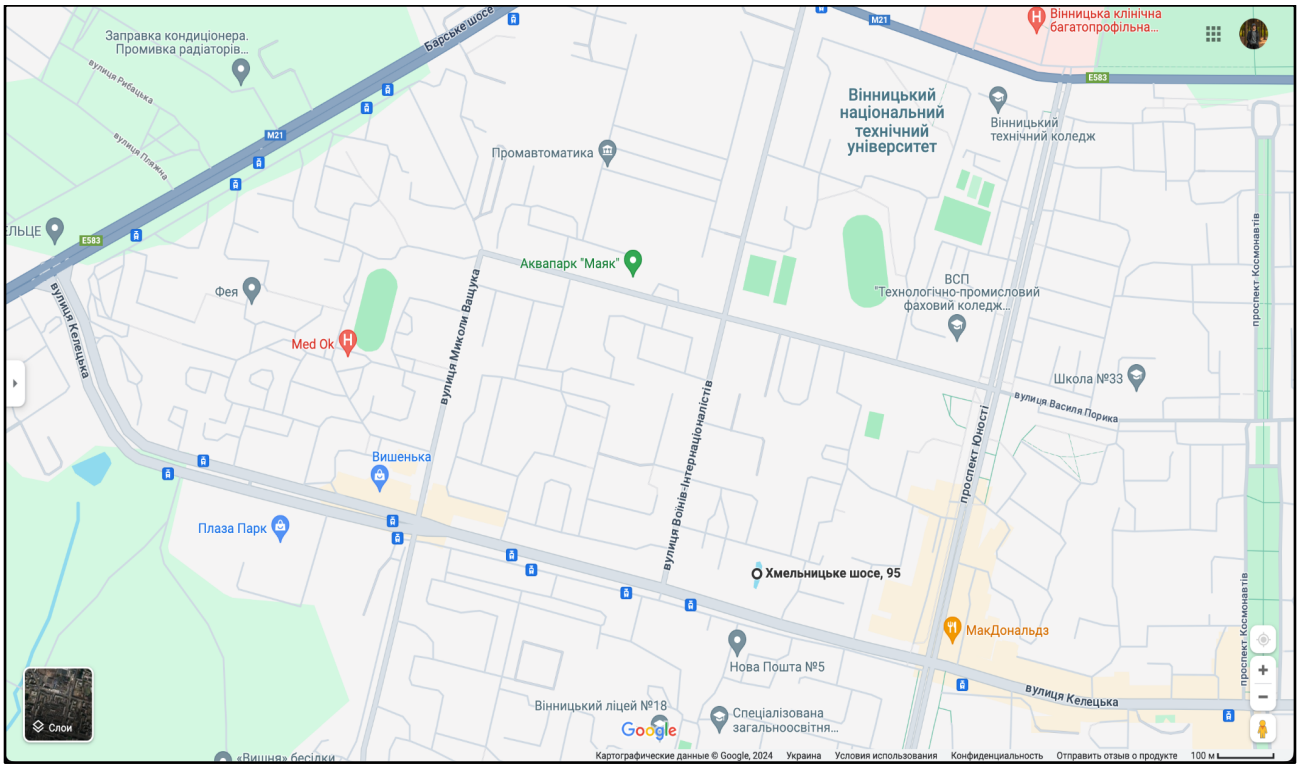



Рисунок Г.5 – Місце проведення експеременту