`Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Комп'ютерний моніторинг плавального басеину з використанням бездротової сенсорної мережі
(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-8
Сергєєв О.В.
(Прізвище, ініціали)

спеціальності *123 – Комп'ютерна інженерія*
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма *Комп'ютерна інженерія*
(повна назва освітньої програми)

Керівник проф. Кривуля Г. Ф

Допускається до захисту
Зав кафедри _____ Чумаченко С. В.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 Комп’ютерна інженерія
(код і повна назва)

Тип програми Освітньо-професійна

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри
(підпис)
«____»_____20____р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Сергєєву Олександру Вячеславовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Комп'ютерний моніторинг плавального басеину з використанням бездротової сенсорної мережі
затверджена наказом університету від 21 травня 2025 р. № 403 Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 11 червня 2025 р.
3. Вихідні дані до роботи
Веб-інтерфейс для візуалізації даних, серверною частиною на Node.js базою даних PostgreSQL, емулятором сенсорних даних, бездротовою мережею Wi-Fi

4. Перелік питань, що потрібно опрацювати в роботі
Аналіз та обґрунтування вибору компонентів системи, розробка апаратної та програмної частини, реалізація емулятору, реалізація веб інтерфейсу, тестування

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____
Схема передачі данних, інтерфейс, який бачить користувач_____
_____
_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Видача теми проекту, узгодження і затвердження теми	06.05.2025 – 09.05.2025	
2	Аналіз та вибір методу вирішення поставленої задачі поставленої задачі	10.05.2025 – 15.05.2025	
3	Ознайомлення з літературними джерелами аналіз та вибір методу вирішення поставленої задачі вибір системних засобів вирішення	16.05.2025 – 20.05.2025	
4	Проектування апаратної програмної частини	21.05.2025 – 25.05.2025	
5	Налагодження та тестування комп'ютерної системи	26.05.2025 – 31.05.2025	
6	Оформлення пояснювальної записки	01.06.2025 – 08.06.2025	
7	Перевірка виконаного проекту керівником, допуск до захисту	09.06.2025 – 11.06.2025	
8	Захист проекту	12.06.2025 – 24.06.2025	

Дата видачі завдання 06 травня 2025 р.

Здобувач  Сергєєв О.В.
(підпис)

Керівник роботи  проф. Кривуля Г.Ф.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи містить: 42 сторінок, 6 рисунків, 4 джерела за переліком посилань

СИСТЕМА МОНІТОРИНГУ, ESP32, pH, ХЛОР, ТЕМПЕРАТУРА, ВЕБ-ІНТЕРФЕЙС, БАЗА ДАНИХ, EMULATOR.

Метою кваліфікаційної роботи є розробка веб інтерфейсу для візуалізації данних моніторингу плавального басейну на основі бездротової сенсорної мережі з використанням мікроконтролера ESP32. У роботі розглянуто сучасні підходи до контролю якості водного середовища, проаналізовано переваги автоматизованих рішень та недоліки традиційних методів.

Описано архітектуру системи, яка включає сенсори температури, pH та хлору, серверну частину, веб-інтерфейс для відображення інформації в реальному часі та систему збереження даних у базі PostgreSQL. Детально розглянуто алгоритм обробки даних, емулятор сенсорів, що дозволяє тестувати систему без фізичних пристроїв, та обґрунтовано вибір інструментів розробки, зокрема Next.js, TypeScript і Prisma ORM.

Система забезпечує безперервний моніторинг, візуалізацію показників води та аналіз безпеки умов для плавання з урахуванням санітарних норм. Розроблене рішення є масштабованим, надійним і придатним для подальшого впровадження в реальні умови експлуатації.

ABSTRACT

The explanatory note of the qualification work contains: 42 pages, 6 figures, 4 sources listed in the references.

MONITORING SYSTEM, ESP32, pH, CHLORINE, TEMPERATURE, WEB INTERFACE, DATABASE, EMULATOR.

The purpose of this qualification work is to develop a web interface to develop a web interface for visualizing swimming pool monitoring data based on a wireless sensor network using the ESP32 microcontroller. The work examines modern approaches to water quality control, analyzes the advantages of automated solutions, and discusses the shortcomings of traditional methods.

The system architecture is described in detail, including temperature, pH, and chlorine sensors, a backend server, a web interface for real-time data visualization, and data storage using a PostgreSQL database. The data processing algorithm is presented, along with a sensor emulator that enables system testing without physical devices. The choice of development tools Next.js, TypeScript, and Prisma ORM is also justified.

The system provides continuous monitoring, visualization of water parameters, and an assessment of swimming safety based on sanitary norms. The developed solution is scalable, reliable, and suitable for deployment in real-world operating conditions.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ESP32 – мікроконтролер з вбудованими модулями Wi-Fi та Bluetooth, використовується для реалізації IoT-проєктів.

pH – показник кислотності або лужності водного середовища.

мг/л – міліграми на літр, одиниця вимірювання концентрації речовини у воді.

API – Application Programming Interface – інтерфейс програмування застосунків.

HTTP – HyperText Transfer Protocol – протокол прикладного рівня для передачі гіпертексту.

UI – User Interface – користувацький інтерфейс.

SSR – Server-Side Rendering – рендеринг сторінки на стороні сервера.

SSE – Server-Sent Events – односпрямований канал передачі даних від сервера до клієнта.

ORM – Object-Relational Mapping – технологія для зв’язку об’єктно-орієнтованих мов програмування з реляційними базами даних.

ВСТУП

Сучасний світ рухається до автоматизації процесів, сфера моніторингу не є винятком. Велику роль у цьому відіграють технології Інтернету речей (IoT) та бездротових сенсорних мереж (БСМ). БСМ пропонують гнучкі, масштабовані, фінансово та енергоефективні варіанти моніторингу та збору даних у реальному часі водних середовищ. Використання БСМ дозволяє безперервно відстежувати критично важливі показники, такі як температура, рівень pH, каламутність, концентрація хлору та багато інших показників. Підхід моніторингу з використанням БСМ підвищує рівень безпеки відвідувачів та оптимізує витрати за рахунок спрощення отримання показників.

Проблема розробки систем комп'ютерного моніторингу плавального басейну є актуальною через складність рішень представлених на ринку, зменшення витрат за рахунок вибору більш дешевих запчастин та кастомізація рішень під певні задачі.

Початок розробки та використання сенсорів у моніторингу водних середовищ почалось у 1906 році з розробки скляного електроду[1] Крамером Максом[2]. Вже у 1960-х технології оптичних сенсорів зробили великий крок з винаходом першого лазеру, це дозволило різко збільшити точність вимірювань та знизити витрати. В той же час люди почали переходити з лабораторних аналізів на автоматизовані системи збору та обробки даних. Дослідженням БСМ почали займатися з 1990-х, але використовувати мережі у моніторингу водоймищ почали тільки у 2000-х. Вони набули великої популярності через низьку вартість реалізації та легкість впровадження[2].

Також інтеграція веб технологій у комп'ютерний моніторинг з використанням БСМ стає здебільш актуальною та значимою частиною моніторинг програм, чим більше розповсюджується інтернет.

1. ТЕОРЕТИЧНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНОГО СТАНУ ПРОБЛЕМИ КОМП'ЮТЕРНОГО МОНІТОРИНГУ ПЛАВАЛЬНИХ БАСЕЙНІВ

Сучасний світ прагне до безпеки в різних сферах, плавальні басейни не є виключенням. Норми, рекомендації, правила – це все є складовою системи направленою на зменшення небезпеки у воді. Тому комп’ютерний моніторинг стає все більш актуальним.

Вода може бути небезпечною для здоров’я людини, тому якість води у басейні має першочергове значення. Вода повинна відповідати високим гігієнічним стандартам, тому що під час купання майже неможливо уникнути потрапляння води у рот.

В Україні є багато норм та документів які ці норми регламентують. Основним документом є Закон України «Про забезпечення санітарного та епідемічного благополуччя» від 24 лютого 1994 року. Плавальним басейнам потрібно дотримуватись цих норм, тому стає питання щодо методів моніторингу та аналізу води.

1.1Загальні відомості про бездротові сенсорні мережі (БСМ)

Бездротова сенсорна мережа (БСМ, англ. Wireless Sensor Network, WSN) – це розподілена сенсорна мережа, що складається з множини сенсорів та, можливо, виконавчих пристроїв (актуаторів), які об'єднані між собою за допомогою радіоканалів та здатні до самоорганізації .

Основні елементи, що включають у себе БСМ:

– сенсор(датчик). Пристрій який сприймає контрольований вплив та перетворює дані вимірювання в сигнал;

– сенсорний вузол. Пристрій що складається що найменше з одного сенсору, мікроконтролера, модулю зв’язку та джерела живлення. Вузол забезпечує збір даних, їх первинну обробку та може передавати дані далі;

– актуатор: Виконавчий пристрій, що реагує на сигнал для зміни стану керованого об'єкта;

Перевагами БСМ є здатність самовідновлюватись, само організовуватись, передача даних на значні відстані використовуючи малі потужності за рахунок ретрансляції, малий розмір, простота установки та обслуговування.

Рисунок 1.1 - Датчик рівня РН

1.2 Огляд існуючих підходів та систем моніторингу параметрів водного середовища

Ефективний контроль за якістю води у басейнах є ключовим для забезпечення здоров’я відвідувачів та дотримання санітарних норм. Проведемо огляд та аналіз різних підходів до моніторингу.

Традиційні методи контролю зазвичай базуються на лабораторному аналізі стану води. Ці методи передбачають збір зразків, транспортування до лабораторії та проведення фізико-хімічного аналізу для отримання результатів. Як правило, відбір проб проводиться раз на місяць.

Недоліки: низька оперативність. Процедура збору зразків та лабораторного аналізу зазвичай займає велику кількість часу. Також за час між аналізами стану води, її якість може значно погіршитись та вийти за норми без можливості своєчасного реагування;

З розвитком електроніки та сенсорних технологій з’явилася можливість здійснювати постійний моніторинг будь якої складності об’єктів та оперативного реагування на зміни у стані води. У системах з використанням БСМ зазвичай є об’єднанням сенсорів та обладнання що, що забезпечують моніторинг водних ресурсів шляхом автоматичного вимірювання фізико-хімічних показників.

Переваги автоматизованих систем:

- оперативність. Можливість отримання даних аналізу у реальному часі або з великою періодичністю;
- зручність та віддалений доступ. Сучасні системи дозволяють підключатися віддалено за допомогою веб-інтерфейсів та мобільних додатків;
- автоматизоване дозування хімікатів. Деякі системи дозволяють окрім моніторингу також дозувати кількість хімікатів у автоматичному режимі;
- безперервність моніторингу. Автоматизовані системи забезпечують постійний контроль на відміну від традиційних методів;

Недоліки:

- вартість. Початкові інвестиції в обладнання та його встановлення можуть бути значними;
- обмеженість вимірюваних параметрів: Не всі системи можуть вимірювати повний спектр можливих забруднювачів, особливо складні органічні сполуки або специфічні мікроорганізми, для визначення яких все ще може знадобитися лабораторний аналіз.
- обслуговування та калібрування: Датчики потребують регулярного обслуговування, очищення та калібрування для забезпечення точності показань;
- складність: Деякі системи можуть бути складними в налаштуванні та експлуатації, вимагаючи кваліфікованого персоналу;

1.3 Короткий огляд комерційних та дослідницьких рішень

компактна інтелектуальна тест-система, що здійснює моніторинг стану води за такими показниками, як TDS (електропровідність), pH, ORP (Cl^-) та температура. Система розраховує необхідну кількість хімії для басейну, враховуючи поточну та майбутню погоду, та дозволяє контролювати якість води в режимі реального часу через мобільний додаток. Ціна 17 000 грн[4];

Hayward AquaRite Plus та AquaRite Advanced: Це системи, які не тільки контролюють параметри води, але й автоматично дезінфікують її шляхом електролізу солі (перетворення солі на вільний хлор). Вони можуть регулювати рівень pH, ORP та вільного хлору, керувати додатковим обладнанням та підтримувати віддалене Wi-Fi управління. Такі станції забезпечують самоочищення електродів та самодіагностику.

Ціна 91 805 – 96 850 грн[5];

Станції контролю від AstralPool: Компанія пропонує автоматизовані системи для постійного моніторингу та регулювання ключових параметрів води в басейні, таких як рівень pH. Ціна 115 150 – 256 090 грн;

Оглядаючи вищезазначене можна стверджувати, що впровадження систем комп’ютерного моніторингу є актуальною, також зважаючи на недоліки готових рішень можемо зробити висновок об актуальності самостійної розробки таких систем.

2 ПРОЕКТУВАННЯ ВСІЄЇ СИСТЕМИ

2.1 Схема БСМ яку ми використовуємо:

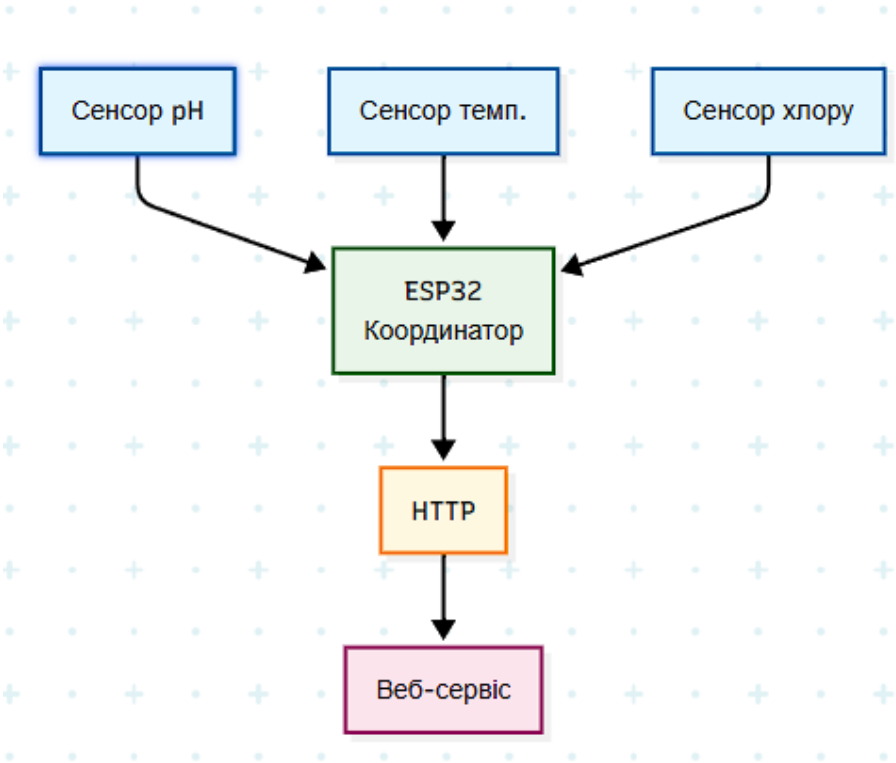

Рисунок -2.1

Опис ESP32 який виступає ядром цієї схеми:

ESP32 — це потужний і доступний мікроконтролер від компанії Espressif Systems, призначений для проєктів Інтернету речей (IoT) та вбудованих систем. Він має двоядерний процесор Xtensa LX6 з тактовою частотою до 240 МГц, 520 КБ оперативної пам’яті та підтримку зовнішньої пам’яті. Основна перевага ESP32 — вбудовані Wi-Fi та Bluetooth (Classic і BLE), що дозволяє легко створювати бездротові пристрої. Мікроконтролер підтримує різні інтерфейси: GPIO, SPI, I2C, UART, ADC, DAC, PWM і сенсорні входи.[3]

2.2Вимоги до веб-інтерфейсу

Щоб спроектувати веб-сервіс який би відповідав меті роботі нам треба висунути перелік критеріїв, яким би він повинен був відопвідати:

– наглядність. веб-інтерфейс повинен дозволяти людині з першого погляду зрозуміти поточний стан басейну. користувач має миттєво бачити, чи безпечно плавати, які параметри в нормі, а які потребують уваги. всі критичні показники (температура, pH, хлор) повинні бути представлені у вигляді зрозумілих карток з кольоровою індикацією статусу;

– зручність. інтерфейс має бути інтуїтивно зрозумілим для будь-якої людини, незалежно від технічної підготовки. навігація повинна бути простою — від головної сторінки до панелі керування, від поточних даних до історичних графіків. користувач не повинен замислюватися, де знайти потрібну інформацію або як інтерпретувати отримані дані;

– масштабованість. Система повина мати змогу легко масштабуватись незалежно від кількості датчиків та данних, які потрібно буде обробити.

– емульованість. у випадках, коли фізичне підключення до контролерів недоступне або недоцільне, система повинна підтримувати режим емулювання. це дозволяє тестувати функціональність інтерфейсу, логіку обробки даних та сценарії взаємодії без необхідності в реальному обладнанні. важливо, щоб емульовані дані були максимально наближені до реальних, з можливістю їх налаштування для перевірки крайових випадків.

2.3 Алгоритм передачі даних

Алгоритм :

– запитуємо дані з сенсорів. система ініціює регулярні звернення до підключених сенсорів для отримання актуальної інформації про стан середовища, зокрема температури, pH, рівня хлору тощо;

– отримуємо їх у контролері. контролер приймає ці дані, агрегує їх та підготовлює до подальшої обробки, з урахуванням можливих помилок чи нестабільності в сигналі;

– обробляємо їх. система нормалізує, фільтрує та інтерпретує отримані значення, визначаючи, які з них потребують особливої уваги, а які є в межах норми;

– передаємо через HTTP до нашого інтерфейсу. дані надсилаються на веб-інтерфейс через API у форматі, зручному для відображення та подальшої візуалізації;

– проводимо аналіз отриманих даних. застосовується логіка для виявлення відхилень, формування попереджень та рекомендацій для користувача на основі встановлених порогів та шаблонів поведінки;

– зберігаємо дані у базу даних. всі історичні значення, події та аномалії фіксуються у базі даних для подальшого аналізу, формування графіків і звітів;

– відображаємо дані у тому вигляді, який найзручніший для користувача. інтерфейс формує візуальні блоки, графіки, статуси та повідомлення так, щоб користувач міг швидко і легко зрозуміти ситуацію та прийняти рішення.

Повна схема пересування данних показано на риснуку 2.2

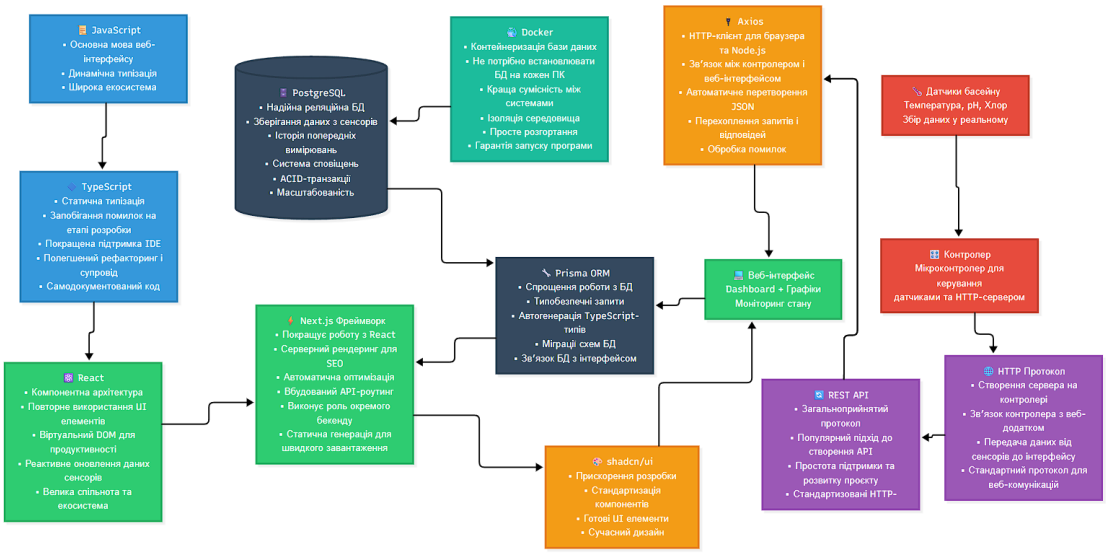

Рисунок 2.2 - Повна схема пересування данних

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЄКТУ

Так як в нас є схема, але фізичної частини її немає, важливим компонентом розробки веб-інтерфейсу – це емуляція схеми. Для можливості отримання даних та тестування різних кейсів, які можуть бути в житті.

3.1 Схема емулятора.

Емулятор скдадається з двух частин:

- класс-емулятор. Відповідає за емуляцію всіх данних
- http сервер реалізований на fastify, відповідає за отримання данних від класу-емулятору та зв’язок з веб інтерфейсом.

3.2 Про клас SensorEmulator

Клас SensorEmulator реалізує складну логіку генерації реалістичних даних з урахуванням фізичних обмежень. Він підтримує два основні режими: нормальні флуктуації (плавні зміни в межах ±0.1 одиниці) та наближення до цільових значень (поступовий рух зі швидкістю 0.05 одиниці за крок). Система автоматично обмежує значення в межах поточного діапазону роботи.

Покриваємі кейси: Емулятор моделює звичайну роботу датчиків температури (25-30°C), pH (7.2-8.0) та хлору (0.1-0.5 мг/л), тимчасові аварійні ситуації (додавання льоду, хімічні збої), постійні зміни режиму (кипіння води), та автоматичне відновлення після аварій. Підтримується ручний скидання до нормального режиму.

Тестування та валідація: Система забезпечує відтворюваність сценаріїв для тестування алгоритмів обробки даних, дозволяє проводити стресове тестування екстремальних ситуацій, валідує логіку роботи систем моніторингу при різних типах змін даних, та слугує для навчання персоналу поведінці датчиків у різних умовах.

Вхідні та вихідні значення: Клас приймає параметри ініціалізації (мінімум, максимум, початкове значення, швидкість нормальних змін, швидкість наближення до цілі) та методи керування (встановлення цільового значення, тригери винятків, скидання). Видає поточні значення датчиків з часовими мітками, статус наближення до цілі, та інформацію про поточні обмеження діапазону.

Далі дані отриманні від класу емуляції потрапляють до сервісу, де ці дані збираються та відправляються до нашого веб інтерфейсу через htpp сервер, як відповідь на запит по endpoint.

Лістинг 3.1- код Методу generateNextValue:

```
 public generateNextValue(): number {
if (this.isApproachingTarget && this.targetValue !== null) {
this._approachTargetValue();
} else {
this._generateNormalFluctuation();
}
this.currentValue = Math.max(this.currentMin, Math.min(this.currentMax,
this.currentValue));
return this.currentValue;
}
```

Лістинг 3.2 - код класу EmulatorService

```
 import { SensorEmulator } from "../emulator/emulator";
export class EmultatorService {
  private readonly temperatureSensor = new SensorEmulator(25, 30, 26.5, 0.1);
    private readonly phSensor = new SensorEmulator(7.2, 8.0, 7.5, 0.005);
    private readonly chlorineSensor = new SensorEmulator(0.1, 0.5, 0.3,
0.005);
```

```
    async fetchSensorData() {
        try {
          const sensorData = {
            currentData: {
              temperature: this.temperatureSensor.generateNextValue(),
              pH: this.phSensor.generateNextValue(),
              chlorineLevel: this.chlorineSensor.generateNextValue()},
            timestamp: new Date().toISOString()
          };



          return sensorData;
        } catch (error: unknown) {
          const errorMessage = error instanceof Error
            ? error.message
            : 'Unknown error occurred';

          console.error(`Error fetching sensor data: ${errorMessage}`);
          throw new Error(`Failed to fetch sensor data: ${errorMessage}`);
        }
      }
}
```

Лістинг 3.3 – Код серверу

```
import { EmultatorService } from './dataEmutlationService/Emultator.service'
import cors from '@fastify/cors'
const server = fastify()

server.register(cors, {
  origin: 'http://localhost:3000',
  methods: ['GET', 'POST', 'PUT', 'DELETE'],
  allowedHeaders: ['Content-Type', 'Authorization'],
  credentials: true,
})


const emulator = new EmultatorService();

server.get('/fetchCurrentData', async (request, reply) => {
```

```
 const data = emulator.fetchSensorData()
  return data;
})

server.listen({ port: 8080 }, (err, address) => {
  if (err) {
    console.error(err)
    process.exit(1)
  }
  console.log(`Server listening at ${address}`)
})
```

3.3 Вибор архітектурного підходу при розробці веб інтерфейсу

Було розглянуто 2 підходи.

Підхід з розділеними React + Node.js.

Традиційна архітектура передбачає створення окремого React-додатку для фронтенду та Express/Fastify сервера на Node.js для бекенду. React-додаток взаємодіє з API через fetch/axios запити, отримуючи дані від мікроконтролерів через REST або WebSocket з'єднання на бекенді.

Такий підхід вимагає налаштування CORS політик, управління станом між клієнтом і сервером, синхронізації типів даних. Розгортаннявключає два окремі процеси, налаштування reverse proxy (nginx), управління портами та змінними оточення для кожного сервісу. При розробці необхідно запускати dev-сервери для фронтенду та бекенду паралельно.

Backend for Frontend в Next.js.Next.js дозволяє створити єдиний додаток, де API Routes служать прошарком між мікроконтролерами та React-компонентами. Дані від пристроїв обробляються в `/api` маршрутах, а візуалізація відбувається в React-компонентах того ж додатку.

3.4 Обґрунтування вибору 2го варіанту.

Єдиний простір розробки. Це означає, що вся логіка обробки даних мікроконтролерів знаходиться поруч з компонентами візуалізації. При зміні формату даних пристрою можна одночасно оновити парсинг в API Route та типи в компонентах без перемикання між проектами.

Реактивність у реальному часі. Вона реалізується простіше завдяки Server-Sent Events або WebSocket в API Routes. Стан підключень до мікроконтролерів може бути глобальним для додатку, уникаючи дублювання з'єднань.

Типобезпечність end-to-end. TypeScript покриває шлях від отримання даних пристрою до відображення в графіках. Одна схема типів описує дані на всіх рівнях додатку.

Спрощене розгортання особливо критично для IoT-проектів. Docker-контейнери або Vercel-deploy замінює оркестрацію multiple services. Конфігурація мережевих портів мінімальна - тільки один вхідний порт для HTTP та вихідні для мікроконтролерів.

Оптимізація продуктивності через SSR. SSR дозволяє попередньо завантажувати історичні дані сенсорів, що важливо для аналітичних дашбордів. ISR може кешувати агреговану статистику пристроїв.

Відлагодження та моніторинг централізовані. Логи обробки даних пристроїв та рендерингу компонентів знаходяться в одному місці. Next.js DevTools показують повну картину від API до UI.

Для IoT-аналітики важливіше швидко ітерувати над алгоритмами обробки даних сенсорів, ніж готуватися до мікросервісної архітектури.

3.5 Вибір засобів розробки для веб-додатку системи моніторингу басейну

Для розробки веб-додатку системи моніторингу басейну було розглянуто наступні засоби розробки програмного забезпечення:

- Next.js
- React
- TypeScript
- Prisma ORM
- PostgreSQL
- React Query (TanStack Query)

3.5.1 Next.js

Next.js – це React-фреймворк, розроблений компанією Vercel, що дозволяє створювати повноцінні веб-додатки з серверним рендерингом (SSR), статичною генерацією (SSG) та API-маршрутами. Next.js надає оптимізовану структуру проєкту, автоматичне розділення коду, оптимізацію зображень та вбудовану підтримку TypeScript.

Перевагами Next.js є швидка розробка завдяки готовим рішенням, автоматична оптимізація продуктивності, SEO-дружність через серверний рендеринг, простота розгортання та велика екосистема. Проте, Next.js може бути надмірним для простих додатків та має певну криву навчання для новачків.

3.5.2 React

React – бібліотека JavaScript для створення користувацьких інтерфейсів, розроблена компанією Meta. React використовує компонентний підхід, що дозволяє створювати перевикористовувані UI-компоненти та ефективно керувати станом додатку. Перевагами React є велика спільнота, багато готових бібліотек, гнучкість у розробці, віртуальний DOM для оптимізації продуктивності та підтримка TypeScript. Недоліками є необхідність вивчення додаткових бібліотек для повноцінного додатку та потенційна складність для новачків.

3.5.3 TypeScript

TypeScript – це надмножина JavaScript, що додає статичну типізацію та об'єктно-орієнтоване програмування. TypeScript допомагає виявляти помилки на етапі компіляції та покращує читабельність коду.

Перевагами TypeScript є раннє виявлення помилок, краща підтримка IDE, покращена читабельність коду та безпека типів. Недоліками є додаткова складність налаштування та необхідність вивчення системи типів.

3.5.4 Prisma ORM

Prisma – це сучасний ORM (Object-Relational Mapping) для Node.js та TypeScript, що забезпечує типобезпечний доступ до бази даних. Prisma автоматично генерує TypeScript-типи на основі схеми бази даних.

Перевагами Prisma є типобезпека, автоматична генерація типів, інтуїтивний API, міграції бази даних та підтримка різних баз даних.

Недоліками є додаткова складність налаштування та потенційні проблеми з продуктивністю при складних запитах.

3.5.5 Вибір та архітектура бази данних

Було розглянуто 3 бази данних:

PostgreSQL. - це потужна об'єктно-реляційна система управління базами даних з відкритим кодом. PostgreSQL підтримує ACID-транзакції, складні запити та має розширені можливості, також підтримує як SQL, так і NoSQL функції.

Переваги:

– відмінна підтримка часових рядів та аналітичних запитів - ідеально для історії вимірювань з датчиків

– розширені типи даних (JSON, масиви, геопросторові типи) - корисно для зберігання складних даних датчиків

– надійність та ACID-сумісність для критично важливих даних моніторингу

– чудова інтеграція з Prisma ORM

– можливість масштабування при розвитку проєкту

– вбудовані функції для агрегації та аналізу даних

Недоліки:

– трохи складніше налаштування порівняно з простішими БД;

– більше споживання ресурсів для невеликих проектів.

SQLite - легка вбудована реляційна база даних, яка зберігається в одному файлі.

Переваги:

– нульове налаштування - ідеально для початку маленького проєкту
– дуже швидка для читання даних
– не потребує окремого сервера
– добра підтримка Prisma.

Недоліки:
– обмеження по одночасних записах - проблематично для множинних датчиків
– складне масштабування при розвитку проєкту
– відсутність деяких аналітичних функцій для побудови графіків.

MySQL- популярна реляційна база даних з відкритим кодом.

Переваги:
– широка підтримка спільноти
– добра продуктивність для веб-додатків
– підтримка Prisma ORM

Недоліки:
– менш розвинені аналітичні можливості порівняно з PostgreSQL
– обмежена підтримка JSON та складних типів даних
– менше функцій для роботи з часовими рядами

Критерії по яким була вибрана саме PostgreSQL:

– відповідність завданням. PostgreSQL створена з урахуванням потреб аналітики та роботи з великими обсягами даних. У межах проєкту необхідно зберігати історію вимірювань датчиків, здійснювати побудову графіків, виявляти тенденції та проводити подальший аналіз даних - усе це напряму відповідає сильним сторонам PostgreSQL.

– гнучкість у роботі з даними. PostgreSQL підтримує зберігання як строго структурованих даних (наприклад, температури, рівня pH, концентрації хлору), так і неструктурованих метаданих у форматі JSON. Це забезпечує зручність та універсальність у зберіганні даних з різною природою без втрати цілісності.

– потужні інструменти для аналітики. Убудовані засоби PostgreSQL, зокрема функції для роботи з часовими рядами, агрегації та віконні функції, надають широкі можливості для аналізу динаміки показників басейну у часі, що є ключовим для забезпечення ефективного моніторингу.

– масштабованість. Хоча проєкт на початковому етапі має невеликий обсяг, обрана СКБД готова до подальшого зростання - як у плані кількості підключених сенсорів, так і в обсягах історичних даних. PostgreSQL здатна ефективно обробляти великі масиви інформації без суттєвого зниження продуктивності.

– інтеграція з Prisma. PostgreSQL має найповнішу підтримку у Prisma ORM — як щодо типів даних, так і функціональних можливостей. Це дозволяє зручно реалізовувати доступ до бази, формувати складні запити.

– надійність і збереження даних. Для системи моніторингу, де критичною є цілісність та безпека даних вимірювань, важливим фактором є підтримка ACID-властивостей, які гарантує PostgreSQL. Це забезпечує впевненість у надійності збереження навіть при збої чи несподіваних обставинах.

Таким чином, обрання PostgreSQL як основної СКБД для мого проєкту є обґрунтованим рішенням, яке поєднує поточні потреби у збиранні та аналізі даних з перспективою подальшого розвитку системи.[4].

3.5.6 React Query (TanStack Query)

React Query – це бібліотека для управління станом серверних даних у React-додатках. Вона забезпечує кешування, синхронізацію та оновлення

даних між сервером та клієнтом. Перевагами React Query є автоматичне кешування, синхронізація даних, оптимістичні оновлення та простота використання. Недоліками є додаткова складність для простих додатків та необхідність вивчення концепцій.

3.5.7 Обгрунтування вибору засобів розробки

Для розробки системи моніторингу басейну було вирішено використовувати сучасний стек технологій: Next.js, React, TypeScript, Prisma, PostgreSQL та React Query.

По-перше, Next.js забезпечує швидку розробку та оптимізовану продуктивність, що критично для системи моніторингу в реальному часі.

По-друге, TypeScript забезпечує типобезпеку та зменшує кількість помилок під час розробки, що важливо для системи, що працює з критичними даними про стан басейну.

По-третє, Prisma забезпечує типобезпечний доступ до бази даних та спрощує роботу з даними сенсорів.

3.6 Налаштування проєкту

Для створення проєкту було використано Next.js з TypeScript. Структура проєкту включає наступні директорії:

- `/src/app` – основні сторінки додатку
- `/src/components` – перевикористовувані компоненти
- `/src/serverActions` – серверні дії
- `/src/lib` – конфігурація бази даних
- `/src/generated` – згенеровані типи Prisma

Конфігурація проєкту зберігається у файлах:

- `package.json` – залежності та скрипти
- `next.config.js` – конфігурація Next.js

- `prisma/schema.prisma` – схема бази даних
- `.env` – змінні середовища

3.7 Розробка програмного коду

Код для системи моніторингу басейну написаний мовою TypeScript з використанням React та Next.js. Код складається з модулів, які відповідають за певний функціонал системи.

3.7.1 Розробка модуля отримання даних сенсорів

Перший модуль відповідає за отримання даних з сенсорів басейну через API. Модуль використовує axios для HTTP-запитів та забезпечує типобезпечне отримання даних.

Основною функцією модуля є `getPoolData()`. Вона виконує HTTP GET-запит до API басейну та повертає дані про температуру, pH та рівень хлору. Функція обробляє помилки та забезпечує надійне отримання даних.

Лістинг 3.4 – Код функції getPoolData()

```
import poolApi from "../instanse";
import { PoolData } from "@/app/types/pool";

async function getPoolData(): Promise<PoolData> {
    try {
      const response = await poolApi.get('/fetchCurrentData');
      return response.data;
    } catch (error) {
      console.error('Error fetching pool data:', error);
      throw error;
    }
  }
  export default getPoolData;
```

Функція `fetchAndNormalize()` об'єднує отримання даних, збереження в базу даних та нормалізацію даних для відображення.

Лістинг 3.5 – Код функції fetchAndNormalize()

```
import { saveDataToBd } from "@/serverActions/saveDataToBd"
import getPoolData from "../instanse/getPoolData/getPoolData"
import { normalizeData } from "./normalizeData"

export const fetchAndNormalize = async ()=> {
    const poolData = await getPoolData()
    console.log(poolData)
    await saveDataToBd(poolData)
    return normalizeData(poolData)
}
```

3.7.2 Розробка модуля аналізу безпеки плавання

Другий модуль відповідає за аналіз даних сенсорів та визначення безпеки плавання в басейні. Модуль використовує хуки React для управління станом та логікою аналізу.

Основною функцією модуля є хук `useSwimmingSafety()`. Він аналізує дані температури, pH та рівня хлору та визначає загальний стан безпеки басейну.

Лістинг 3.6 – Код хука useSwimmingSafety()

```
import { TransformedPoolData } from "@/app/types/pool"
import { NUMBER_OF_PARAMETERS } from "@/app/utils/consts/consts";
import { overall, rules } from "@/app/utils/rules/rules";
import { useEffect, useState } from "react";

export const useSwimmingSafety = (data: TransformedPoolData) => {
  const { currentData } = data;
```

```
        const [overallState, setOverall] = useState<typeof overall[keyof
typeof overall]>(overall.safe);

        const tempStatus = currentData.temperature.status;
        const phStatus = currentData.pH.status;
        const chlorineStatus = currentData.chlorine.status;

        useEffect(() => {
          const calculateOverallSafety = () => {
            const statuses = [tempStatus, phStatus, chlorineStatus];
            const safeCount = statuses.filter(status => status ===
"safe").length;

            if (safeCount === NUMBER_OF_PARAMETERS) {
              return overall.safe;
            }
            if (safeCount === NUMBER_OF_PARAMETERS) {
              return overall.acceptable;
            }
            return overall.warning;
          };

          setOverall(calculateOverallSafety());
        }, [tempStatus, phStatus, chlorineStatus]);

        return {
          overallState,
          tempAnalysis: rules.temperature[tempStatus],
          phAnalysis: rules.ph[phStatus],
          chlorineAnalysis: rules.chlorine[chlorineStatus]
        }
      }
```

Компонент `SwimmingSafety` відображає результати аналізу безпеки плавання з використанням карток та індикаторів стану.

Лістинг 3.7 – Код компонента SwimmingSafety

```
"use client"
```

```
      import { Card, CardContent, CardHeader, CardTitle } from
"@/components/ui/card"
      import { Badge } from "@/components/ui/badge"
      import { Alert, AlertDescription } from "@/components/ui/alert"
      import { CheckCircle, AlertTriangle, XCircle, Info } from "lucide-react"

      import { useSwimmingSafety } from "./hooks/useSwimmingSafety"
      import { TransformedPoolData } from "@/app/types/pool"

      interface SwimmingSafetyProps {
        data: TransformedPoolData;
      }

      export default function SwimmingSafety({ data }: SwimmingSafetyProps) {
        const { tempAnalysis, phAnalysis, chlorineAnalysis, overallState } =
useSwimmingSafety(data);

        const getIcon = (iconName: string) => {
          switch (iconName) {
            case "CheckCircle": return CheckCircle;
            case "AlertTriangle": return AlertTriangle;
            case "XCircle": return XCircle;
            default: return Info;
          }
        };

        const TempIcon = getIcon(tempAnalysis.icon);
        const PHIcon = getIcon(phAnalysis.icon);
        const ChlorineIcon = getIcon(chlorineAnalysis.icon);

        return (
          <div className="space-y-6">
            <Card className="bg-gradient-to-br from-slate-50 to-gray-50">
              <CardHeader>
                <CardTitle className="flex items-center gap-2">
                  <Info className="h-5 w-5 text-blue-600" />
                  Чи безпечно плавати в цьому басейні?
                </CardTitle>
              </CardHeader>
              <CardContent className="space-y-4">
                <Alert variant={overallState.alertType}>
```

```
                <AlertDescription className="flex items-center justify-
between">
                  <div>
                    <div className="font-semibold text-lg mb-
1">{overallState.message}</div>
                    <div className="text-
sm">{overallState.recommendation}</div>
                  </div>
                  <Badge variant={overallState.badgeVariant} className="ml-
4">
                    {overallState.status === "safe" ? "Безпечно" :
overallState.status === "acceptable" ? "Прийнятно" : "Увага"}
                  </Badge>
                </AlertDescription>
              </Alert>

              <div className="grid grid-cols-1 md:grid-cols-3 gap-4 mt-6">
                <div className="p-4 rounded-lg bg-white border">
                  <div className="flex items-start gap-3">
                    <TempIcon className={`h-5 w-5 ${tempAnalysis.color} mt-
0.5`} />
                    <div>
                      <h4 className="font-medium text-gray-900 mb-
1">Температура</h4>
                      <p className="text-sm text-gray-
600">{tempAnalysis.message}</p>
                    </div>
                  </div>
                </div>

                <div className="p-4 rounded-lg bg-white border">
                  <div className="flex items-start gap-3">
                    <PHIcon className={`h-5 w-5 ${phAnalysis.color} mt-0.5`}
/>
                    <div>
                      <h4 className="font-medium text-gray-900 mb-1">Рівень
pH</h4>
                      <p className="text-sm text-gray-
600">{phAnalysis.message}</p>
                    </div>
                  </div>
                </div>
```

```
                <div className="p-4 rounded-lg bg-white border">
                  <div className="flex items-start gap-3">
                    <ChlorineIcon className={`h-5 w-5
${chlorineAnalysis.color} mt-0.5`} />
                    <div>
                      <h4 className="font-medium text-gray-900 mb-1">Рівень
хлору</h4>
                      <p className="text-sm text-gray-
600">{chlorineAnalysis.message}</p>
                    </div>
                  </div>
                </div>

                <div className="mt-6 p-4 bg-blue-50 rounded-lg border border-
blue-200">
                  <h4 className="font-medium text-blue-900 mb-2">Українські
нормативи для басейнів:</h4>
                  <div className="text-sm text-blue-800 space-y-1">
                    <div>• Температура: 24-28°C (приватні басейни для
дорослих)</div>
                    <div>• pH: 7.2-7.6</div>
                    <div>• Хлор: 0.3-0.5 мг/л</div>
                  </div>
                </div>
              </CardContent>
            </Card>
          </div>
        );
      }
```

3.8 Розробка основної сторінки

Останнім модулем є головна сторінка додатку та панель керування, які об'єднують всі інші модулі та забезпечують користувацький інтерфейс системи.

Головна сторінка `page.tsx` забезпечує вхідну точку до додатку з описом функціональності та переходом до панелі керування.

Лістинг 3.8 – Код головної сторінки

```
import { Button } from "@/components/ui/button"
import Link from "next/link"

export default function Home() {
  return (
    <div className="min-h-screen bg-gradient-to-br from-blue-50 to-cyan-
50 flex items-center justify-center p-4">
      <div className="max-w-2xl mx-auto text-center space-y-8">
        <div className="space-y-6">
          <h1 className="text-4xl md:text-6xl font-bold text-gray-900
leading-tight">
            Pool Control
            <span className="block text-blue-600">& Monitoring</span>
          </h1>

          <p className="text-xl md:text-2xl text-gray-600 max-w-xl mx-
auto leading-relaxed">
            Привіт, це інтерфейс для відслітковування показників якості
води в басейні
          </p>
        </div>

        <div className="pt-4">
          <Link href="/dashboard">
            <Button
              size="lg"
              className="text-lg px-8 py-6 bg-blue-600 hover:bg-blue-700
text-white shadow-lg hover:shadow-xl transition-all duration-200 transform
hover:scale-105"
            >
              Перейти до дашборду
            </Button>
          </Link>
        </div>
      </div>
    )
```

```
}
```

Панель керування `dashboard/page.tsx` забезпечує основний інтерфейс для моніторингу стану басейну з використанням React Query для кешування даних.

Лістинг 3.9 – Код панелі керування

```
import { dehydrate, HydrationBoundary, QueryClient } from
"@tanstack/react-query";
import DashboardContent from "./_components/DashBoardContent";
import { fetchAndNormalize } from "../utils/helpers/fetchAndNormalize";

export default async function DashboardPage() {
  const queryClient = new QueryClient()

  await queryClient.prefetchQuery({
    queryKey: ['poolData'],
    queryFn: fetchAndNormalize,
  })

  return (
    <div className="min-h-screen bg-gradient-to-br from-blue-50 to-cyan-
50">
      <div className="container mx-auto px-4 py-8">
        <div className="mb-8">
          <h1 className="text-3xl font-bold text-gray-900 mb-2">
            Панель керування басейном
          </h1>
          <p className="text-gray-600">
            Моніторинг стану води та безпеки плавання в реальному часі
          </p>
        </div>

        <HydrationBoundary state={dehydrate(queryClient)}>
          <DashboardContent />
        </HydrationBoundary>
      </div>
    </div>
  );
```

```
}
```

Також є сторінка графіків. Графіки відображаються для наших 3ьох показників. Та мають можливість переглянути різні часові діапазони.

Нижче на рисунках наведені вигляд веб-інтерфейсу.

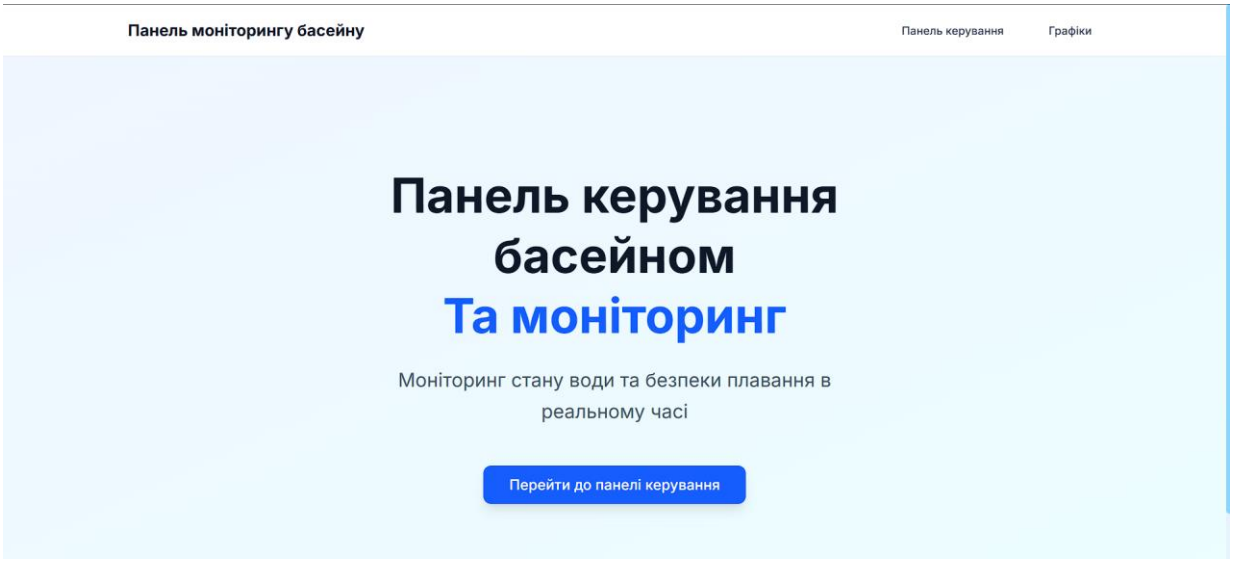

Рисунок 3.1 – головна сторінка

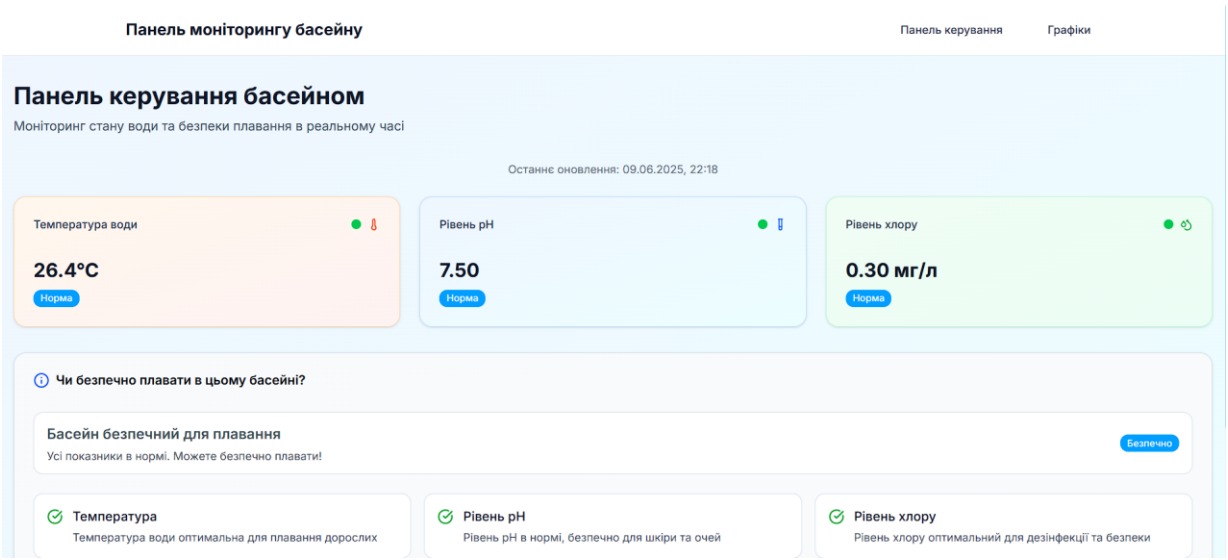

Рисунок 3.2 – Панель керування

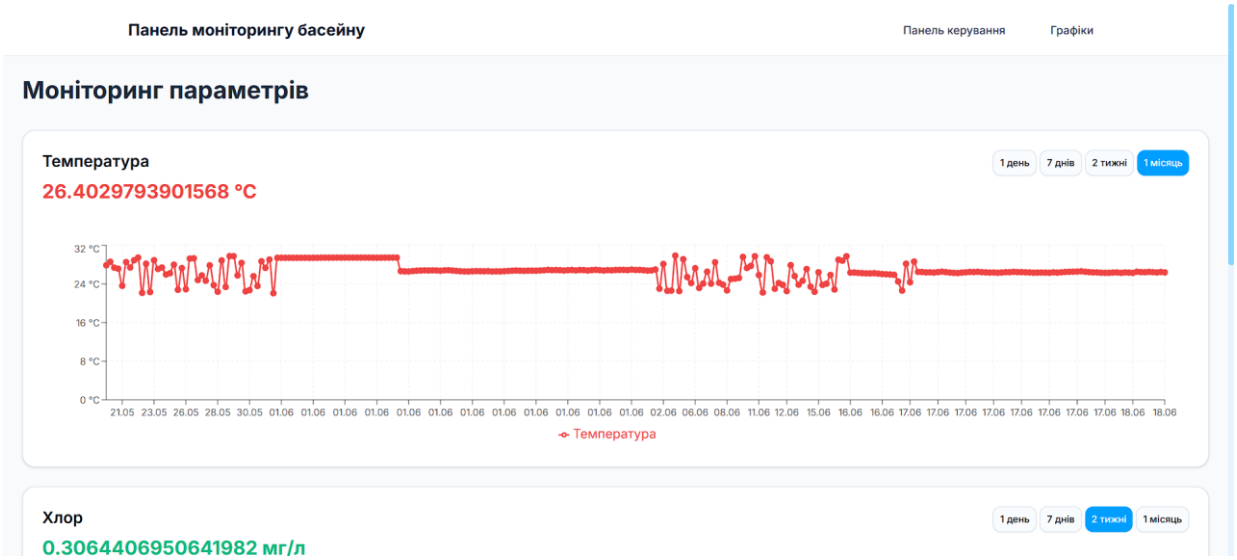

Рисунок 3.3 – Сторінка графіків

Система моніторингу басейну забезпечує безперервний контроль стану води та безпеки плавання через сучасний веб-інтерфейс з використанням типобезпечного коду та оптимізованої архітектури.

ВИСНОВКИ

У кваліфікаційній роботі було розроблено веб інтерфейс для відображення результатів моніторингу плавального басейну на базі бездротової сенсорної мережі з використанням мікроконтролера ESP32. Враховуючи актуальність завдання щодо контролю за якістю води в басейнах, у роботі запропоновано гнучке, масштабоване та економічно доцільне рішення, здатне здійснювати постійний збір і аналіз даних у режимі реального часу.

У процесі розробки проведено аналіз предметної області, сучасних вимог до систем моніторингу водних середовищ та недоліків існуючих комерційних рішень.

Розроблено архітектуру системи з урахуванням вимог до точності, оперативності, масштабованості та зручності користування.

Реалізовано програмну емуляцію сенсорів з моделюванням реалістичних сценаріїв поведінки датчиків температури, рівня pH та хлору.

Створено веб-інтерфейс з використанням сучасного технологічного стеку (Next.js, React, TypeScript, Prisma, PostgreSQL), що забезпечує зручний доступ до аналітики стану басейну.

Обґрунтовано вибір PostgreSQL як основної СКБД завдяки її потужним аналітичним функціям, підтримці часових рядів, надійності та гнучкості у роботі з різнорідними даними.

Застосовано архітектурний підхід «Backend for Frontend» у межах одного проєкту для забезпечення цілісності, продуктивності та типобезпеки.

Таким чином, було реалізовано повноцінну систему моніторингу, яка дозволяє не лише спостерігати за станом водного середовища, а й забезпечити високий рівень автоматизації, точності та інформативності, що є надзвичайно важливим для забезпечення безпеки користувачів басейну.

Отримані результати свідчать про доцільність та ефективність впровадження бездротових сенсорних мереж у сфері контролю якості води.

.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. . A Short History of Sensor Developments for Water Monitoring / Sensileau // Benten Water. – [Електронний ресурс]. – Режим доступу: https://sensileau.benten-water.com/news/a-short-history-of-sensor-developments-for-water-monitoring.html (дата звернення: 18.06.2025). – Заголовок з екрана

2. Tapparello C., Boukerche A. A Reliable Data Delivery Protocol for Heterogeneous Wireless Sensor Networks // ACM Transactions on Sensor Networks. – 2017. – Vol. 13, No. 3. – Article 16. [Електронний ресурс] – Режим доступу:www/URL:https://www.hajim.rochester.edu/ece/sites/tapparello/papers/Tapparello_ACMTOSN2017.pdf – Загол. з екрана.

3. Мікроконтролер ESP32 / IT Master — електроніка та програмування. — [Електронний ресурс]. — Режим доступу: https://itmaster.biz.ua/directory/microcontrollers/esp32.html (дата звернення: 18.06.2025). — Заголовок з екрана.

4. SQLite vs MySQL vs PostgreSQL: A Comparison of Relational Database Management Systems / O.S. Tezer, Mark Drake // DigitalOcean Community Tutorials. – [Електронний ресурс]. – Режим доступу: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems (дата оновлення: 10.03.2022; дата звернення: 18.06.2025). – Заголовок з екрана.