

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА**

**Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»**

Кафедра інформаційних систем в економіці

**ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА
«Інформаційні управляючі системи і технології»**

галузь знань	12 Інформаційні технології
спеціальність	122 Комп'ютерні науки

Форма навчання: очна (денна)

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему: «Розроблення інформаційної системи для контролю якості питної води
централізованого водопостачання»

здобувача Кучера Владислава Васильовича _____
(ПІБ, підпис)

Науковий керівник: д.е.н. професор Устенко Станіслав Веніамінович
(науковий ступінь, учене звання, ПІБ)

(підпис)

**Робота допущена до захисту перед екзаменаційною
комісією з атестації здобувачів вищої освіти (ЕК)**

Завідувач кафедри: _____
(науковий ступінь, учене звання, ПІБ)

(підпис)

Київ 2025

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА**

**Навчально-науковий інститут «Інститут інформаційних технологій в економіці»
Кафедра інформаційних систем в економіці**

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА

«Інформаційні управляючі системи і технології»

галузь знань	12 Інформаційні технології
спеціальність	122 Комп'ютерні науки

ПОГОДЖЕНО:

Керівник проєктної групи (гарант)
освітньо-професійної програми

_____ Устенко С.В.
« ____ » _____ 202_ р.

ЗАТВЕРДЖУЮ:

Завідувач кафедри інформаційних
систем в економіці

_____ Тішков Б.О.
« ____ » _____ 202_ р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувача вищої освіти Кучера Владислава Васильовича

(прізвище, ім'я, по батькові)

Денної форми навчання

очної (денної), заочної, дистанційної

на підготовку кваліфікаційної магістерської роботи

на тему: «Розроблення інформаційної системи для контролю якості питної води централізованого водопостачання»

Тему затверджено наказом ректора Університету від «__» _____ 202_ р. № _____

Кваліфікаційна магістерська робота виконується на матеріалах

План кваліфікаційної магістерської роботи

Розділ I Дослідження предметної області та аналіз підходів до створення інформаційних управляючих систем для контролю якості питної води централізованого водопостачання

(назва розділу)

Розділ II Характеристика системи розроблення інформаційної системи для контролю якості питної води централізованого водопостачання та методи і моделі

(назва розділу)

Розділ III Розроблення проєктних рішень інформаційної системи та їх реалізація

(назва розділу)

Об'єкт дослідження: процес дослідження та розроблення системи для контролю якості питної води централізованого водопостачання

Предмет дослідження: сукупність методів проєктування та впровадження систем для контролю якості питної води централізованого водопостачання

Мета кваліфікаційної магістерської роботи: розроблення ІУС для контролю якості питної води систем централізованого водопостачання

Конкретні завдання, які здобувач повинен виконати для досягнення поставленої мети:

У розділі I Дослідити предметну область контролю якості питної води. Дати характеристику об'єкту дослідження. Провести порівняльний аналіз з існуючими рішеннями, визначити їх переваги та недоліки, зробити висновок щодо можливості їх покращення. Обґрунтувати вибір підходів до створення ІУС. Описати методи, що будуть закладені в системи.

У розділі II Сформувати концептуальну структуру ІУС, визначити основні структурні компоненти системи. Описати методи і моделі, що використовуються в системи, враховуючи функціональні та нефункціональні вимоги. Проаналізувати потенційну поведінку системи та надати характеристику алгоритмам, що будуть використовуватись при реалізації.

У розділі III Обґрунтувати вибір архітектурних рішень та підходів. Розробити базу даних системи, в якій будуть описані сутності та характеристики системи контролю якості питної води. Розробити проектні рішення технічного забезпечення та розробкою програмного забезпечення для системи. Визначити вимоги до тестових даних та провести тестування системи. Провести аналіз виконаних результатів та доцільність впровадження системи. Надати висновки.

**Завдання підготував
науковий керівник**

(підпис)

(ініціали, прізвище)

« ____ » _____ 202_ р.

**Завдання одержав
здобувач**

(підпис)

(ініціали, прізвище)

« ____ » _____ 202_ р.

Реферат

Кваліфікаційна магістерська робота містить 60 сторінок, 6 таблиць, 32 рисунки, список використаних джерел з 63 найменувань, додатки.

«Розроблення інформаційної системи для контролю якості питної води централізованого водопостачання»

Об'єктом дослідження кваліфікаційної магістерської роботи є процеси автоматизованого моніторингу та контролю якості питної води у централізованих системах водопостачання.

Предметом дослідження є сукупність теоретичних, методичних і практичних підходів до проєктування інформаційної системи автоматизованого контролю якості питної води з підтримкою сенсорного моніторингу, локальної обробки даних та централізованого керування.

Мета і завдання дослідження. Основною метою кваліфікаційної магістерської роботи є вдосконалення процесу забезпечення якості питної води шляхом розробки інформаційної управляючої системи, що забезпечує автоматизований моніторинг, аналіз і реагування на відхилення параметрів води від нормативних значень.

Відповідно до поставленої мети визначені такі *завдання*:

- дослідити предметну область, виконати аналіз існуючих систем контролю якості води та відповідної нормативної бази;
- обґрунтувати вибір підходів, технологій та архітектури для створення інформаційної системи;
- визначити структуру та функціональні можливості інформаційної управляючої системи моніторингу якості води; описати методи та моделі в системі моніторингу якості води;
- описати методи та моделі, що застосовуються в системі для забезпечення адаптивного реагування;
- спроектувати базу даних, що поєднує реляційний і нереляційний підходи до зберігання даних;
- спроектувати базу знань для реалізації гнучких багаторівневих правил прийняття рішень;
- розробити програмне забезпечення серверної, клієнтської та мікроконтролерної частин системи;
- обґрунтувати та описати технічне забезпечення для реалізації периферійних точок моніторингу;
- представити результати реалізації прототипу інформаційної системи, включно з тестуванням і аналізом ефективності.

Теоретична, методична та практична значущість отриманих результатів. Під час дослідження проаналізовано сучасні підходи до забезпечення якості води, вивчено нормативну базу України та ЄС, оцінено переваги й недоліки існуючих рішень у сфері автоматизованого контролю. Запропоновано концепцію інтегрованої інформаційної системи, що поєднує локальну обробку даних з централізованим керуванням на основі гнучких правил у форматі JSON. Практичні результати дослідження реалізовані у вигляді повноцінного програмно-апаратного прототипу, що забезпечує моніторинг і автономне реагування на критичні зміни параметрів води. Це сприяє підвищенню якості послуг у сфері водопостачання, зниженню ризиків для здоров'я населення та оптимізації управлінських рішень у галузі екологічного моніторингу.

Рік виконання кваліфікаційної магістерської роботи – 2025.

Рік захисту роботи – 2025.

Ключові слова: інформаційна система, якість питної води, автоматизований моніторинг, сенсорна мережа, ESP32, контроль параметрів, база знань, реагування в реальному часі, гібридна БД.

В і д г у к

про кваліфікаційну магістерську роботу

здобувача навчально-наукового інституту «Інститут інформаційних технологій в економіці»

освітньо-професійної програми «Інформаційні управляючі системи і технології»

Кучера Владислава Васильовича

на тему «Розроблення інформаційної системи для контролю якості питної води централізованого водопостачання»

1. Актуальність теми: дослідження визначається високим професійним рівнем вдосконалення процесу забезпечення якості питної води шляхом розробки інформаційної управляючої системи, що забезпечує автоматизований моніторинг, аналіз і реагування на відхилення параметрів води від нормативних значень.

2. Позитивні риси кваліфікаційної магістерської роботи: Автор роботи провів аналіз існуючих систем контролю якості води та відповідної нормативної бази обґрунтував вибір підходів, технологій та архітектури для створення інформаційної системи; спроектував базу даних; розробив програмне забезпечення серверної, клієнтської та мікроконтролерної частин системи; представив результати реалізації прототипу інформаційної системи, включно з тестуванням і аналізом ефективності.

3. Наявність і цінність самостійних розробок автора Розроблена ІУС може бути ефективно застосована для моніторингу якості питної води в умовах реального використання і її впровадження у комунальних підприємствах, екологічних службах, санітарних установах і громадах як засобу оперативного контролю, прогнозування та оптимізації процесів водопостачання.

4. Наявність недоліків: Рекомендація – потрібно впровадити дану систему на рівні масштабування в систему водопостачання України

6. Загальна оцінка кваліфікаційної магістерської роботи та її допущення до захисту перед ЕК: Результати дослідження по тематиці КМР написана стаття і тези, які опубліковані і стали основою впровадження практичних результатів дослідження. КМР виконана на високому теоретичному і практичному рівнях, відповідає усім вимогам до розроблення КМР ОПП ІУСТ і заслуговує на найвищу оцінку.

Науковий керівник д.е.н., професор С.В. Устенко

16.05.2025

Рецензія

на кваліфікаційну магістерську роботу
здобувача вищої освіти

Кучера Владислава Васильовича

Тема: «Розроблення інформаційної системи для контролю якості питної води
централізованого водопостачання»

Актуальність теми кваліфікаційної магістерської роботи і доцільність її розроблення. Обрана тема є надзвичайно актуальною, зважаючи на сучасні екологічні виклики, зношеність інфраструктури централізованого водопостачання та потребу в підвищенні якості контролю за питною водою. У роботі розглянуто важливі аспекти впровадження цифрових технологій у сфері екологічного моніторингу, що відповідає як державним, так і європейським вимогам. Запропоноване рішення має високу практичну доцільність та відповідає сучасним тенденціям розвитку інформаційних управляючих систем.

Якість проведеного дослідження. Робота характеризується високим рівнем наукової обґрунтованості, логічністю викладення матеріалу та системним підходом до розв'язання поставлених задач. Проведено ґрунтовний аналіз предметної області, нормативної бази та сучасних ІТ-рішень. У дослідженні використано сучасні методи проектування, системного аналізу, моделювання та програмування, що свідчить про високий рівень підготовки здобувача.

Позитивні риси кваліфікаційної магістерської роботи. Серед позитивних рис слід відзначити глибокий аналіз існуючих рішень, обґрунтування вибору власної архітектури системи та використання сучасних технологій. В роботі розроблено повноцінний прототип, який може бути використан для побудови систем екологічного моніторингу, автоматизованих станцій контролю якості води, а також як основа для розширених IoT-платформ у сфері смарт-інфраструктури. До позитивних рис можна також віднести наявність практичної реалізації із тестуванням в умовах, наближених до реальних.

Зауваження. Суттєвих зауважень до структури і змісту роботи немає. Водночас, доцільно було б ширше розкрити порівняльний аналіз економічної ефективності запропонованої системи та розглянути перспективи її масштабування в умовах різних моделей господарювання.

Практична значимість висновків і рекомендацій. Результати дослідження мають чітко виражену прикладну цінність і можуть бути використані в роботі підприємств водопостачання, місцевих органів влади та

екологічних служб. Розроблена система сприяє автоматизації моніторингу якості води, зменшенню впливу людського фактору та підвищенню ефективності управлінських рішень.

Старший науковий співробітник
Інституту кібернетики
імені В.М. Глушкова НАН України
канд. фіз.-мат. наук

Ніколаєвська О.А.

(підпис, ПІБ)

Підпис засвідчую: Заб. канцлер

(посада, підпис)



ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	3
ВСТУП	4
РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПІДХОДІВ ДО СТВОРЕННЯ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ КОНТРОЛЮ ЯКОСТІ ПИТНОЇ ВОДИ.....	10
1.1 Аналіз існуючих інформаційних управляючих систем.....	10
1.1.1 Дослідження предметної області	10
1.1.2 Дослідження інформаційних управляючих систем.....	15
1.2 Обґрунтування вибору підходів і технологій для створення інформаційної управляючої системи	19
РОЗДІЛ 2. ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ ТА МЕТОДИ І МОДЕЛІ.....	24
2.1 Структура і характеристика моделі.....	24
2.2. Методи та моделі в інформаційних управляючих системах і технологіях.....	36
РОЗДІЛ 3. РОЗРОБЛЕННЯ ПРОЄКТНИХ РІШЕНЬ ТА ЇХ РЕАЛІЗАЦІЯ..	39
3.1 Проєктування баз даних для інформаційної управляючої системи	39
3.1.1 Характеристика даних та вибір моделей зберігання	39
3.1.2. Проєктування інфологічних моделей реляційної та нереляційної БД	40
3.1.3 Обґрунтування вибору СКБД та фізичне проєктування.....	42
3.1.4 Контрольний приклад і обрахунок прогнозних обсягів	46
3.2 Проєктування бази знань інформаційної управляючої системи.....	49
3.3 Програмне забезпечення	52

3.3.1	Проектування програмного забезпечення	52
3.3.2	Вибір архітектури програмного забезпечення.....	56
3.3.3	Інструментальні засоби розробки для створення прикладного програмного забезпечення інформаційних управляючих систем.....	58
3.3.4	Тестування програмного забезпечення.....	60
3.4	Технічне забезпечення.....	62
3.4.1	Мета та технічні вимоги до розроблювального пристрою	62
3.4.2	Обґрунтування вибору технічного забезпечення	62
3.4.3	Опис розробленого технічного пристрою	63
3.4.4	Методика виконання дослідно-експлуатаційного випробування технічного пристрою.....	64
3.4.5	Настанова щодо використання технічного пристрою.....	64
3.5	Реалізація інформаційної системи.....	65
	Висновки	70
	Список використаних джерел	73
	Додатки.....	80

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ДСанПіН – Державні санітарні правила і норми

ГДК – Гранично допустима концентрація

ІУС – Інформаційна управляюча система

СКБД – Система керування базами даних

ACID – Atomicity, Consistency, Isolation, Durability (набір властивостей, що гарантують надійну роботу транзакцій бази даних)

BSON – Binary JSON (подання JSON-документів у двійковому вигляді)

HMI – Human-Machine Interface (людино-машинний інтерфейс)

IDE – Integrated Development Environment (інтегроване середовище розробки)

IDEF0 – ICam DEFinition for Function Modeling (модель функціонального моделювання)

IoT – Internet of Things (інтернет речей)

JSON – JavaScript Object Notation (формат обміну структурованими даними)

LoRa – Long Range (бездротовий мережевий протокол, який використовується для побудови мереж Інтернету речей на великі відстані)

ORM – Object-Relational Mapping (об'єктно-реляційне відображення)

REST – Representational State Transfer (архітектурний стиль вебсервісів)

SCADA – Supervisory Control and Data Acquisition (система диспетчерського контролю та збору даних)

SQL – Structured Query Language (структурована мова запитів)

SSR – Server Side Rendering (рендеринг на стороні сервера)

TTL – Time-To-Live (час життя об'єкта, наприклад, у БД або мережах)

UML – Unified Modeling Language (уніфікована мова моделювання)

ВСТУП

Система централізованого водопостачання є важливою складовою задоволення базових потреб людини. При цьому вода може бути як корисною для споживання, так і становити певну загрозу – зокрема, підвищувати ймовірність виникнення захворювань [1, 2]. В умовах зростаючих екологічних ризиків, старіння інфраструктури та ускладнення процесів управління, актуальність розробки автоматизованої інформаційно-управляючої системи для моніторингу якості води є беззаперечною. Така система дозволяє мінімізувати вплив людського фактора, забезпечити оперативне реагування на зміни параметрів якості води та підвищити ефективність управлінських рішень. Більше того, це сприятиме виконанню екологічних, санітарних і нормативно-правових вимог, зокрема вимог ДСТУ та директив ЄС щодо якості питної води.

Область контролю якості питної води не є новою. Вже проведено значну кількість досліджень щодо стану як централізованих, так і нецентралізованих систем водопостачання. Проте, з огляду на нові виклики – зокрема, кліматичні зміни, євроінтеграційний курс України та збройну агресію РФ – питання забезпечення якісної питної води набуває додаткової актуальності порівняно з дослідженнями минулих років.

Зокрема, у праці О. В. Гафурової, Т. С. Новака та С. М. Голуба [3] проаналізовано проблему доступу населення до інформації про якість питної води, а також ускладнення ситуації через війну, потрапляння нітратів у водні об'єкти та зміни клімату, які набувають тропічного характеру. Попри цінність цього дослідження, його автори зосереджуються насамперед на правових аспектах, не пропонуючи технічних або інформаційно-технологічних рішень. Саме тому в межах цієї роботи пропонується розробка інформаційної системи, яка б забезпечувала централізований моніторинг параметрів якості води з відкритим доступом для громадськості.

Аналогічно актуальним є дослідження О. В. Шестопалова та співавторів [4], у якому розглядаються сучасні методи аналізу якості води, включно з фізико-хімічними та мікробіологічними показниками. Автори акцентують увагу на проблемах забруднення водних ресурсів в умовах активної урбанізації, індустріального навантаження та війни. У дослідженні також подано порівняльний аналіз нормативів якості води в Україні та інших країнах, що дозволяє оцінити рівень наближення національних стандартів до європейських. Водночас у публікації відсутній акцент на інструменталізацію процесів контролю якості води.

Ще одним важливим напрямом є дослідження сучасного стану водних ресурсів у контексті дотримання санітарних норм. Так, у роботі Н. О. Кануннікової та співавторів [5] проведено аналіз якості води басейну Дніпра з порівнянням її характеристик із водами інших країн. Автори акцентують увагу на необхідності вдосконалення систем моніторингу якості води та управління ресурсами в умовах кліматичних змін та антропогенного тиску. Однак, дослідження має радше аналітичний характер і не висвітлює практичні підходи до цифрової трансформації процесів контролю.

Практичніші аспекти удосконалення водопостачання, зокрема в умовах міської інфраструктури, розкрито в роботі С. М. Епоєна та співавторів [6]. Автори пропонують методи підвищення ефективності роботи водопровідних мереж, проте не приділяють достатньої уваги питанням моніторингу якості води в режимі реального часу, що лишає нішу для впровадження інформаційних систем.

Не зважаючи на наявність нормативної бази та технічних регламентів, на сьогоднішній день в комунальній сфері надання послуг, зокрема систем централізованого водопостачання, відсутня технічна реалізація автоматизованих систем для контролю, та навіть дистанційного регулярного моніторингу показників, що ускладнює вчасне реагування на критичні відхилення. Тому, існує потреба в розробленні масштабованої інтегрованої інформаційної системи контролю якості води, що відповідатиме сучасним технічним і нормативним вимогам.

Результати дослідження можуть бути використані підприємствами водоканалу, органами місцевого самоврядування, екологічними службами, а також у наукових установах для аналізу динаміки показників якості води та відповідності технічного забезпечення стандартам.

Особистий внесок автора полягає у формалізації вимог до інформаційної системи, розробці концептуальної архітектури, а також у створенні прототипу модуля моніторингу з візуалізацією даних та можливістю автоматизованого дистанційного контролю. Адже попри наявність низки публікацій, присвячених аналізу якості питної води та правовим аспектам її контролю, практичні інструменти для реалізації автоматизованих, доступних та масштабованих систем моніторингу залишаються малодослідженими в науковій літературі.

Метою дослідження є розробка і впровадження ІУС для контролю якості питної води, яка забезпечує автоматизований збір, обробку, аналіз даних у режимі реального часу та реалізацію алгоритмів оперативного автономного реагування на виявлення відхилень від нормативних показників.

До завдань дослідження віднесено:

1. Провести аналіз предметної області та дослідити сучасні підходи до моніторингу і управління якістю питної води з урахуванням нормативно-правової бази України та ЄС.
2. Оцінити технічні й програмні можливості існуючих ІУС у сфері контролю якості води та виявити їхні недоліки і обмеження.
3. Розробити концептуальну модель інтегрованої ІУС, яка забезпечуватиме автоматизований збір та аналіз даних в режимі реального часу, а також автономне реагування на відхилення параметрів якості води згідно з попередньо встановленими правилами.
4. Спроекувати архітектуру та функціональну структуру системи, передбачивши інтеграцію сенсорних пристроїв на базі мікроконтролерів, серверного компонента для керування правилами та зберігання даних, а також веб-інтерфейсу для адміністраторів та кінцевих споживачів.

5. Реалізувати алгоритм автоматичного керування параметрами якості води на рівні окремих контрольних точок, з можливістю гнучкого централізованого налаштування.
6. Розробити прототип системи, провести його тестування в умовах, наближених до реальних, та оцінити ефективність функціонування й відповідність встановленим вимогам.

Об'єктом дослідження є процес контролю якості питної води в централізованих системах водопостачання, що включає автоматизований моніторинг та автономні процедури реагування на відхилення ГДК води.

Предметом дослідження виступають ІУС автоматизованого моніторингу та контролю якості питної води, включно з їх структурою, алгоритмами обробки даних, способами реалізації автономного керування процедурами очищення та методами забезпечення надійності і стабільності роботи в умовах масштабованості.

Для досягнення поставленої мети використано такі методи наукового дослідження:

- Методи аналізу та синтезу – для дослідження сучасного стану предметної області, виявлення переваг і недоліків існуючих систем контролю якості води.
- Методи системного аналізу і моделювання – для побудови концептуальної та функціональної моделей запропонованої ІУС, визначення взаємодії між компонентами системи.
- Методи математичного та статистичного аналізу – для обробки та інтерпретації отриманих даних про якість води, а також для визначення ефективності алгоритмів керування.
- Об'єктно-орієнтоване проектування – для розробки структури системи із застосуванням UML-діаграм, що дозволяє наочно представити архітектуру, основні компоненти та їхні взаємозв'язки.

- Інженерно-технічне проектування – для обґрунтування вибору технічних компонентів, зокрема сенсорних пристроїв, мікроконтролерів, способів їх підключення та методів передачі даних.
- Експериментальний метод – для перевірки функціональності розробленого прототипу системи, оцінки його ефективності та стабільності роботи в умовах, що максимально наближені до реальних.

Інформаційною базою дослідження слугують наукові публікації вітчизняних та зарубіжних авторів, нормативні документи (зокрема, ДСТУ, ДСанПіН та Директиви ЄС), офіційні дані МОЗ, Держводагентства України, а також звіти екологічних організацій і практичні приклади реалізації подібних систем у світі.

Наукова новизна роботи полягає в поєднанні методів інтелектуального аналізу даних, сенсорного моніторингу та автоматизованого управління у рамках єдиної системи, адаптованої до сучасних викликів, зокрема в умовах воєнного стану та зміни клімату.

Загальна структура магістерської роботи складається з трьох основних розділів:

У першому розділі здійснено аналіз предметної області – розглянуто сучасний стан систем водопостачання, нормативно-правову базу України та ЄС, проаналізовано існуючі підходи до моніторингу якості питної води, виявлено ключові проблеми поточних рішень, а також обґрунтовано доцільність створення автоматизованої системи контролю на базі сенсорної мережі.

Другий розділ присвячено проектуванню інформаційної управляючої системи. У ньому описано концептуальну архітектуру, визначено структурні компоненти системи, розроблено функціональні моделі, схеми бази знань і бази даних, наведено обґрунтування вибору мікроконтролера ESP32 та гібридної СКБД (PostgreSQL + MongoDB), а також описано метод адаптивного реагування на основі логічних правил.

У третьому розділі наведено результати реалізації прототипу системи: описано програмну реалізацію серверної, клієнтської та мікроконтролерної частин, представлено архітектуру програмного забезпечення, результати моделювання, тестування, дослідної експлуатації та рекомендації з практичного використання. Також оцінено продуктивність, обсяг даних і ефективність системи в умовах, наближених до реального середовища.

Розроблена інформаційна система моніторингу якості питної води орієнтована на підвищення рівня технологічної оснащеності підприємств водопостачання, забезпечення відповідності якості води національним і міжнародним стандартам, а також зниження ризиків для здоров'я населення. Ефективність запропонованих рішень підтверджується підвищенням швидкості реагування на відхилення, зменшенням потреби у ручній обробці даних, забезпеченням прозорості та обґрунтованості управлінських рішень у сфері екологічного контролю.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПІДХОДІВ ДО СТВОРЕННЯ ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ КОНТРОЛЮ ЯКОСТІ ПИТНОЇ ВОДИ

1.1 Аналіз існуючих інформаційних управляючих систем

1.1.1 Дослідження предметної області

Предметна область дослідження охоплює процеси, пов'язані з моніторингом та контролем якості питної води в централізованих системах водопостачання. В умовах сучасної урбанізації, збільшення навантаження на обласні центри, зміни клімату та військово-політичної обстановки, забезпечення якості води є критично важливим пунктом для громадського життя.

Моніторинг в даній сфері включає постійний або періодичний збір параметрів води для дотримання гігієнічних вимог до питної води, призначеної до споживання людиною. Контроль, в свою чергу, означає здатність системи не лише зафіксувати порушення, а й впливати на процеси очищення, подачі та розподілу води в автоматичному або напівавтоматичному режимі.

Згідно українського законодавства, вода питна, призначена для споживання людиною (питна вода), – вода, склад якої за органолептичними, фізико-хімічними, мікробіологічними, паразитологічними та радіаційними показниками відповідає вимогам державних стандартів та санітарного законодавства (з водопроводу - водопровідна, фасована, з бюветів, пунктів розливу, шахтних колодязів та каптажів джерел), призначена для забезпечення фізіологічних, санітарно-гігієнічних, побутових та господарських потреб населення, а також для виробництва продукції, що потребує використання питної води [7]. Тобто вода, що призначена для споживання людиною, повинна відповідати певним гігієнічним вимогам для того, щоб вона могла вважатись придатною для вживання. А враховуючи, що системи централізованого постачання обслуговують значну частину населення країни, то низька якість води окрім екологічної небезпеки, може також спричинити значні

витрати на охорону здоров'я та навіть мати вплив на продуктивність громадян. Перелік типових показників якості питної води представлено на рисунку 1.1.



Рисунок 1.1 – «Класифікація показників якості питної води»

Джерело: використане інтернет-джерело [60]

Для кожного з них відповідною структурою встановлюються допустимі межі. Проби води відбираються на різних етапах процесу, зокрема проби беруться на виході з джерела, після кожного етапу очищення та з кінцевих точок. Далі, ці проби доставляють до лабораторії в спеціальних ємностях. В залежності від типу дослідження, термін та умови зберігання можуть бути різними – від 2-6 годин для мікробіологічних досліджень до днів для хімічних. Після цього проводяться відповідні хімічні реакції, необхідні для виведення точних показників. Показники порівнюють із встановленими нормативами. Якщо виявлено перевищення ГДК, вживаються заходи, такі як повторна обробка води або сповіщення відповідних органів [8].

До основних об'єктів управління процесом постачання та моніторингу якості питної води належать:

- Водозабірні станції. Вони можуть бути розташовані біля поверхневих (річки, озера) або підземних джерел. Основні ризики – забруднення через промислові викиди, сільськогосподарські стоки чи природні фактори.
- Очисні споруди. Процес очищення води може включати такі етапи:
 - Механічне очищення (видалення великих часток, таких як пісок та мул [9]);
 - Хімічне очищення (зціджування, хлорування [10]);
 - Біологічне очищення (застосування мікроорганізмів для розкладання органічних забруднювачів [11]).
- Системи транспортування та розподілу води. Сюди входять трубопроводи, насосні станції та резервуари [12].

Контроль за дотриманням встановлених вимог на місцевому рівні виконується органами місцевого самоврядування, а на державному рівні – органами державного санітарно-епідеміологічного нагляду, зокрема це Державна служба України з питань безпечності харчових продуктів та захисту споживачів (Держпродспоживслужба [13]). Вона також виконує функцію моніторинг якості питної води, перевірку на дотримання нормативів на місцевому рівні, а також реагування на випадки виявлення відхилень. Задля цього існують Центри контролю та профілактики хвороб (ЦКПХ) МОЗ, що здійснюють епідеміологічний моніторинг та аналізують зв'язок між якістю води та поширенням захворювань [14]. Європейські стандарти якості питної води, засновані на рекомендаціях ВООЗ, значно суворіші за українські. У країнах ЄС якість води регулюється Директивою 98/83/ЄС [15]. Основні показники, як-от жорсткість води, рівень амонію чи заліза, мають жорсткіші обмеження в ЄС, ніж в Україні. Україна поступово імплементує ці європейські норми, проте цей процес ускладнений через значні витрати на модернізацію систем водопостачання та очищення води. Це зумовлено необхідністю оновлення технологій та забезпечення відповідності жорсткішим

нормативам, зокрема станом обладнання та деякими відносно полегшеними вимогами в законодавстві [16].

Загалом, згідно національної доповіді про якість питної води та стану питного водопостачання та водовідведення в Україні у 2013 році від Міністерства розвитку громад та територій України, за період з 2016 року відбувається тенденція до збільшення доступу до централізованих систем водопостачання у містах, селищах та селах. Проте, внаслідок початку повномасштабної агресії рф в Україні та руйнувань системи, спостерігається певний регрес протягом останніх двох років [17, 18]. Крім того, в рамках інтерв'ю Українським новинам, Заступниця МОЗ Марина Слободніченко звернула увагу на те, що в Україні, згідно європейським стандартам, немає «питної» води, а є лише «технічна», тобто вода для виробничого або обробного процесу [19].

Для виконання завдань контролю якості води, водоканали та інші місцеві підприємства виконують регулярні перевірки різних параметрів води, включаючи рівень хімічних речовин, мікробіологічну чистоту, наявність токсичних сполук та інші фактори, що впливають на якість питної води. Аналіз результатів надає необхідну інформацію для прийняття відповідних заходів у разі виявлення будь-яких проблем або відхилень в якості води. Загалом, існують способи і проводити контроль якості питної води і в домашніх умовах, наприклад органолептичні властивості (запах, колір, смак, прозорість) [20, 21]. І хоча результати таких перевірок є доволі обмеженими та неточними, саме проведення перевірок в кінцевій точці допоможе включити в зону уваги також стан мережі водопостачання. Зношеність інфраструктури є проблемою, що може призвести до вторинного забруднення вже очищеної води. Адже моніторинг стану якості води відбувається лише на етапах збирання та очищення води – далі відбувається транспортування розподіленою системою, де централізований аналіз фактично неможливий. Додатково, можливе зношення очищувальних споруд, що послаблює очищувальні можливості [22].

Крім законодавчого та інфраструктурного фактору, на середовище прийняття рішень впливає також соціальний фактор – рівень споживання води, що

може коливатись в залежності від зовнішніх умов, та обізнаність населення щодо якості води. Не можна й оминати зовнішні екологічні чинники: забруднення джерел води через неправильну утилізацію, використання промисловістю та неправильне ведення сільського господарства. Так, згідно обсягів скидів забруднених (забруднених без очистки та недостатньо очищених) стічних вод у водні об'єкти від Державної служби статистики України, серед основних забруднювачів води виокремлюють ЖКГ (77%), потім промисловість (17%), та сільське господарство (0,5%). Проте існують чинники, що вказують на можливість зниження впливу сільського господарства внаслідок недосконалості системи моніторингу забруднення води [23].

Сучасні виклики в сфері водопостачання вимагають більш детального та швидкого процесу прийняття рішень. Ручне оброблення інформації є трудомістким та часовитратним процесом, що також має властивість бути схильним до помилок. Традиційні методи контролю якості води залежать від періодичного забору проб та лабораторного аналізу, що є трудомістким і не дозволяє оперативно реагувати на зміни. Значна частина забруднень може залишатися непоміченою між регулярними перевітками, що призводить до запізненого вжиття заходів. Крім того, сучасні системи досить часто є слабоавтоматизованими, і як наслідок зменшення рівня контролю на проміжних етапах. В результаті отримується збільшення факторів впливу на якість води [24]. З огляду на зазначені фактори, доцільним є створення інформаційно-управляючої системи з такими ключовими характеристиками:

- Автоматизований збір даних від сенсорів;
- Обробка та аналіз інформації (вбудована логіка, граничні значення, тренди);
- Прийняття рішень (алгоритми з адаптивним порогом, шаблони реагування);
- Керування виконавчими механізмами (електроклапани, хімдозатори тощо);
- Людино-машинний інтерфейс для перегляду, налаштування та контролю вручну;
- Журнали подій для аудиту, аналізу ефективності роботи і пошуку залежностей;

- Можливість автономної роботи.

Цей підхід дозволяє перейти від реактивного до превентивного управління якістю води, що особливо актуально в умовах кліматичних викликів, інфраструктурного зносу та війни [25].

1.1.2 Дослідження інформаційних управляючих систем

Системи контролю якості питної води є доволі комплексними інформаційними системами, тому безпосередні аналоги не є розповсюдженим явищем. Проте, існує ряд платформ, що дозволяють виконувати деякі функції: інтегруватися з сенсорами, забезпечувати віддалений контроль та виконувати автоматичне оповіщення. До таких систем відносять:

- **SCADA-системи [26].** Вони використовуються для збору даних з пристроїв та моделювання інформаційної панелі, включно з обробкою даних. Такі системи підтримують масштабування, інтеграцію з різними джерелами даних та можливість автоматизації процесу контролю якості. Важливою перевагою є підтримка роботи в реальному часі та гнучкість впровадження для різних сценаріїв використання (наприклад, підключення до бази даних, або робота на зовнішньому сервері). До недоліків можна віднести високу вартість впровадження такої системи, та складність налаштування специфічних вимог у випадку роботи з малими підприємствами. Прикладом такої системи є Fuxa [27]. FUXA виступає як надійне веб-програмне забезпечення, що дозволяє швидко створювати та розгортати масштабовані системи SCADA, HMI, Dashboard або IoT. Приклад використання зображено на рисунку 1.2.

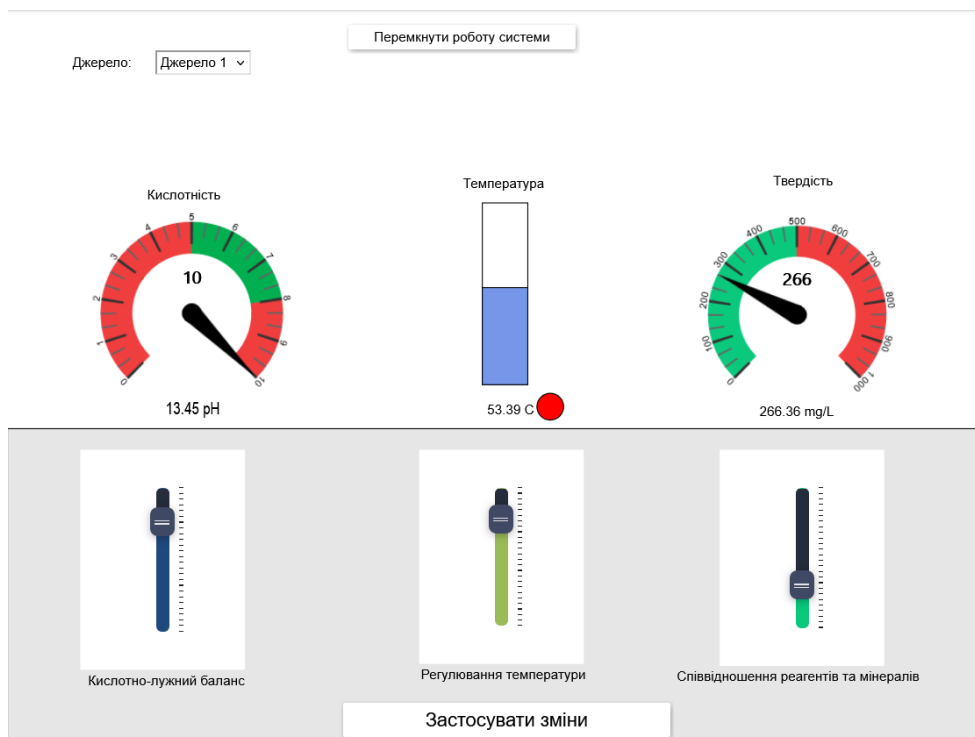


Рисунок 1.2. – «Приклад використання Fuxa для моделювання системи контролю якості води»

Джерело: розроблено автором

- **IoT-платформи для моніторингу води [28].** Такі системи використовуються безпосередньо з IoT-сенсорами, та дозволяють збирати дані про аналіз якості води. Наявна підтримка автоматичного сповіщення в разі відхилення від норм. До переваг можна віднести хмарне зберігання даних, можливість доступу до системи з мобільних платформ та швидкість роботи. Серед недоліків залежність від інтернет-з'єднання, можливі проблеми з безпекою даних, а також обмежений функціонал реагування. Крім того, як правило, такі платформи є закритими та платними. Прикладом є Aquasuite [29], Аyyeka [30], Aquawatch [31], платформа Synox [32], Disrupt-X [33](див. рисунок 1.3) тощо.

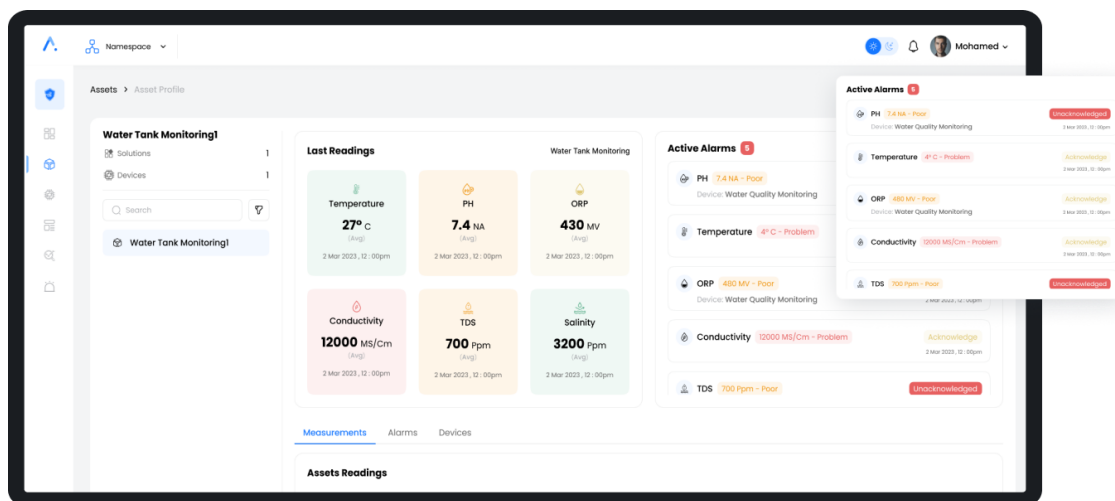


Рисунок 1.3 – «Інтерфейс Disrupt-X»

Джерело: використане інтернет-джерело [34]

- **Системи моніторингу води.** Це комплексні вузькоспеціалізовані середовища для збирання та надсилення показників з поодиноких джерел даних. Вони являють собою в першу чергу набір сенсорів з можливістю подальшої інтеграції до інформаційної системи. І все ж, такі системи спрямовані на оптимізацію використання ресурсу, а не контролю якості питної води. Прикладом є ENVIRA [34], Energyly [35], CONNECTED SENSORS [36] (див. рисунок 1.4).

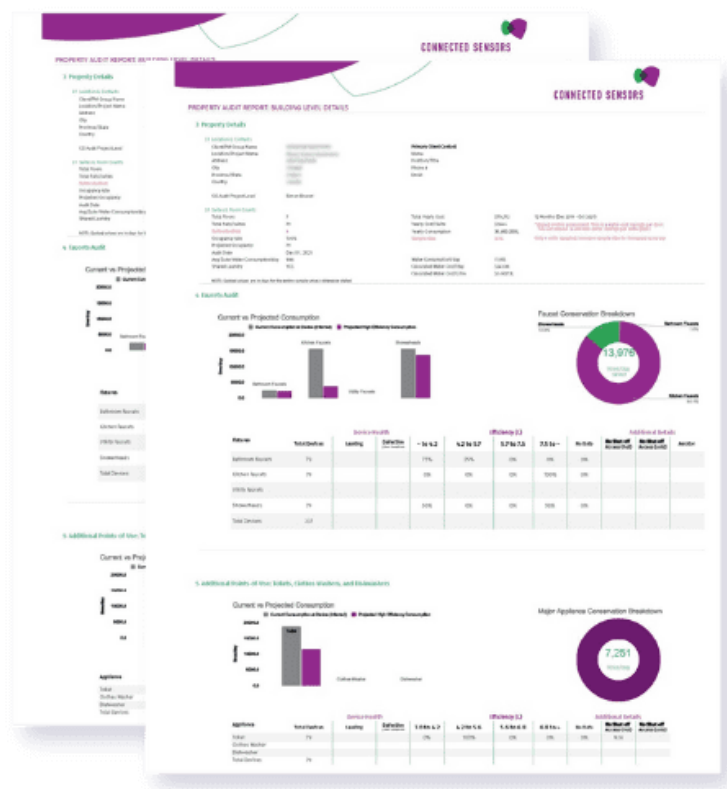


Рисунок 1.4 – «Інтерфейс Professional Commercial Water Audit від Energyly»

Джерело: використане інтернет-джерело [36]

У зв'язку зі стрімким розвитком інформаційних технологій, наразі існує кілька перспективних методів та технологій в розробці, що дозволять значно підвищити ефективність прийняття рішень. Деякі з них вже частково використовуються в найбільш актуальних системах. Зокрема, це використання алгоритмів ШІ та технології машинного навчання задля виявлення взаємозв'язку між показниками якості питної води та можливими зовнішніми факторами [37]. Це дозволить потенційно значно підвищити точність прогнозів, проте збільшить складність розробки та потребує великого об'єму історичних даних. Хмарні платформи, як альтернатива використання власних серверних потужностей, можуть забезпечити доступ до даних з будь-якого місця та пристрою, і дозволяють масштабувати обчислювальні потужності залежно від потреб системи [38]. Використання мікроконтролерів на базі Arduino [39] або ESP32 [40] для збору даних з датчиків кислотності, каламутності та температури допоможе створити автономну інфраструктуру. Такі модулі можуть працювати як автономні системи і

передавати дані в ІУС через Wi-Fi або LoRa. Як аналог, можна використовувати оптичні сенсори та спектроскопію для моніторингу води для підвищення точності вимірювання та можливості виявляти більш широке коло ГДК, проте це значно підвищує вартість розробки і створює певну залежність від зовнішніх пристроїв.

1.2 Обґрунтування вибору підходів і технологій для створення інформаційної управляючої системи

Для створення ІУС контролю якості питної води необхідно обрати найбільш відповідні підходи, технології та інструментальні засоби, що дозволять досягти найефективнішої роботи системи та максимальної відповідності поставленим вимогам. Зважаючи на особливості предметної області та актуальні виклики, було розглянуто декілька потенційних технологічних та архітектурних рішень. Вибір оптимального рішення здійснювався на основі низки ключових критеріїв. До них належать:

- Вартість розробки, впровадження та подальшого розширення системи;
- Складність технічної реалізації;
- Надійність та автономність роботи у випадку проблем зі зв'язком чи електроживленням;
- Гнучкість налаштування правил контролю якості води;
- Можливість масштабування системи до додаткових місць, приладів та складніших правил;
- Простота використання адміністраторами та кінцевими споживачами питної води;
- Відповідність нормативним та санітарним нормам України та ЄС.

Таким чином, було розглянуто три основні альтернативні варіанти побудови ІУС(див. таблицю 1.1):

- Використання SCADA-системи (наприклад, Fuxa);
- Використання готових IoT-платформ (наприклад, Disrupt-X);

- Власна розробка системи на базі мікроконтролерів ESP32 чи Arduino та використання власного сервера.

Таблиця 1.1 – «Порівняльна характеристика альтернативних варіантів побудови ІУС»

Критерії вибору	SCADA-системи	ІоТ-платформи	Власна розробка
Вартість реалізації	Висока	Середня/висока(в залежності від платформи)	Низька
Складність реалізації	Висока	Середня	Середня
Надійність та автономність роботи	Середня	Низька	Висока
Гнучкість налаштування правил	Середня	Низька	Висока
Можливість масштабування	Висока	Висока	Висока
Простота використання	Середня	Висока	Висока
Відповідність нормативам	Висока	Середня	Висока

Джерело: розроблено автором

На основі проведеного аналізу зроблено висновок, що розробка власної системи на базі мікроконтролерів ESP32 з інтегрованим сервером є найбільш доцільним з наступних причин:

- Менша вартість реалізації, оскільки використовуються доступні мікроконтролери і сенсори, і наразі існує вибір з різних цінових категорій технічної частини;
- Висока автономність і надійність через використання децентралізованої інфраструктури та мінімального навантаження на пристрої. Таким чином,

система в екстрених умовах матиме змогу працювати від власного енергоживлення та в умовах відсутності зв'язку з сервером;

- Гнучкість налаштування правил, що реалізовується за допомогою центрального сервера, де адміністратори можуть дистанційно керувати алгоритмами процедур впливу на якісний стан води в реальному часі;
- Відповідність законодавчим нормам завдяки можливості гнучкого налаштування параметрів якості води та способів впливу;
- Можливість масштабування, оскільки підключення нових контрольних точок та сенсорів буде максимально спрощеним і не вимагатиме подальшої модифікації архітектури системи.

Для реалізації системи було обрано мікроконтролер ESP32. Вибір саме цього мікроконтролера ґрунтується на низці суттєвих переваг порівняно з альтернативними рішеннями (наприклад, Arduino Uno, Arduino Mega або Raspberry Pi) [41], ESP32:

- Містить вбудовані модулі Wi-Fi та Bluetooth, що дозволяє легко і без додаткових компонентів забезпечувати зв'язок із сервером. Це значно спрощує архітектуру та знижує загальну вартість і складність системи.
- Має спеціальні режими зниженого енергоспоживання (Deep Sleep Mode), що дозволяє точкам контролю працювати автономно тривалий час навіть за умов перебоїв електроживлення, забезпечуючи безперервність моніторингу та управління.
- Оснащений двоядерним процесором із частотою до 240 МГц та великим об'ємом оперативної пам'яті (520 кБ), що забезпечує швидку обробку даних з численних датчиків у реальному часі та можливість виконувати складні алгоритми прямо на пристрої.
- Широко використовується в IoT-проектах завдяки доступності великої кількості бібліотек, документації та спільноти розробників, що значно спрощує розробку, налагодження та масштабування системи.

- Є доступним і недорогим компонентом, що дає змогу ефективно масштабувати систему на десятки чи навіть сотні точок моніторингу без суттєвого збільшення загальної вартості проєкту.

Концептуальна модель запропонованого рішення заснована на децентралізованій архітектурі типу «клієнт-сервер», що складається з таких основних компонентів:

- **Точки контролю** (мікроконтролери ESP32), оснащені сенсорами для вимірювання фізико-хімічних показників (температура, рН, каламутність тощо). Ці пристрої забезпечують автоматичний моніторинг параметрів якості води та можуть локально запускати процедури очищення чи коригування стану води за заданими алгоритмами, незалежно від сервера.
- **Центральний сервер** виконує завдання зберігання, аналізу та візуалізації даних, що надходять від точок контролю. На сервері також здійснюється конфігурація правил, за якими точки контролю реагують на зміни показників якості води.
- **Адміністративний веб-інтерфейс**, який дозволяє адміністратору системи налаштовувати параметри роботи точок контролю, переглядати історію показників, отримувати повідомлення про критичні ситуації та додавати нові точки моніторингу.
- **Користувацький інтерфейс** (окрема веб-сторінка), доступний кінцевим споживачам для перегляду актуальної інформації про стан якості питної води в різних районах на інтерактивній карті.

Точки контролю періодично здійснюють обмін даними з сервером для актуалізації правил і передачі історичних даних. У разі критичних відхилень від заданих нормативних меж, система автоматично інформує відповідального адміністратора для швидкого реагування на ситуацію.

Отже, запропонована концептуальна модель дозволяє максимально ефективно виконати поставлені завдання, знизити витрати на впровадження,

підвищити рівень надійності та забезпечити відповідність сучасним нормативним вимогам.

РОЗДІЛ 2. ХАРАКТЕРИСТИКА ІНФОРМАЦІЙНИХ УПРАВЛЯЮЧИХ СИСТЕМ ТА МЕТОДИ І МОДЕЛІ

2.1 Структура і характеристика моделі

Контекстна діаграма функціональної моделі процесу «Контроль якості питної води» зображена на рисунку 2.1. Вона описує головну функцію інформаційної системи – забезпечення контролю якості питної води у децентралізованій архітектурі.

До вхідних даних дані про фізико-хімічні параметри води, які зчитуються в реальному часі та показники алгоритмів обробки води(наприклад, рівень активації фільтрації);

На виході отримується опціональне сповіщення адміністраторові (операторові) про відхилення показника від ГДК; Відображення стану якості води та алгоритмів обробки у графічному вигляді для адміністраторів, а також окремий інтерфейс для кінцевих користувачів; Можливі прямі команди адміністратора(оператора) щодо задіяння алгоритмів обробки води.

Даними процесами відбувається керування за допомогою встановлення правил реагування на події(набір умов та логік, за якими система приймає рішення), параметрів збору(деякі точкові налаштування конкретних сенсорів), політика оновлень(параметри, за якими локальна машина виконує запит до серверу. Важливо зазначити, що в даному випадку реалізовується механізм першого запиту з локального пристрою, оскільки це є більш енергоефективним, ніж альтернатива постійного очікування запиту з серверу), та ролей доступу для визначання рівня прав доступу до інтерфейсу.

Для цього використовуються такі механізми, як мікроконтролери з сенсорами, де відбуваються локальні обчислення; центральний сервер з базами даних, де відбувається просунута обробка; адміністративний та користувацький інтерфейс.

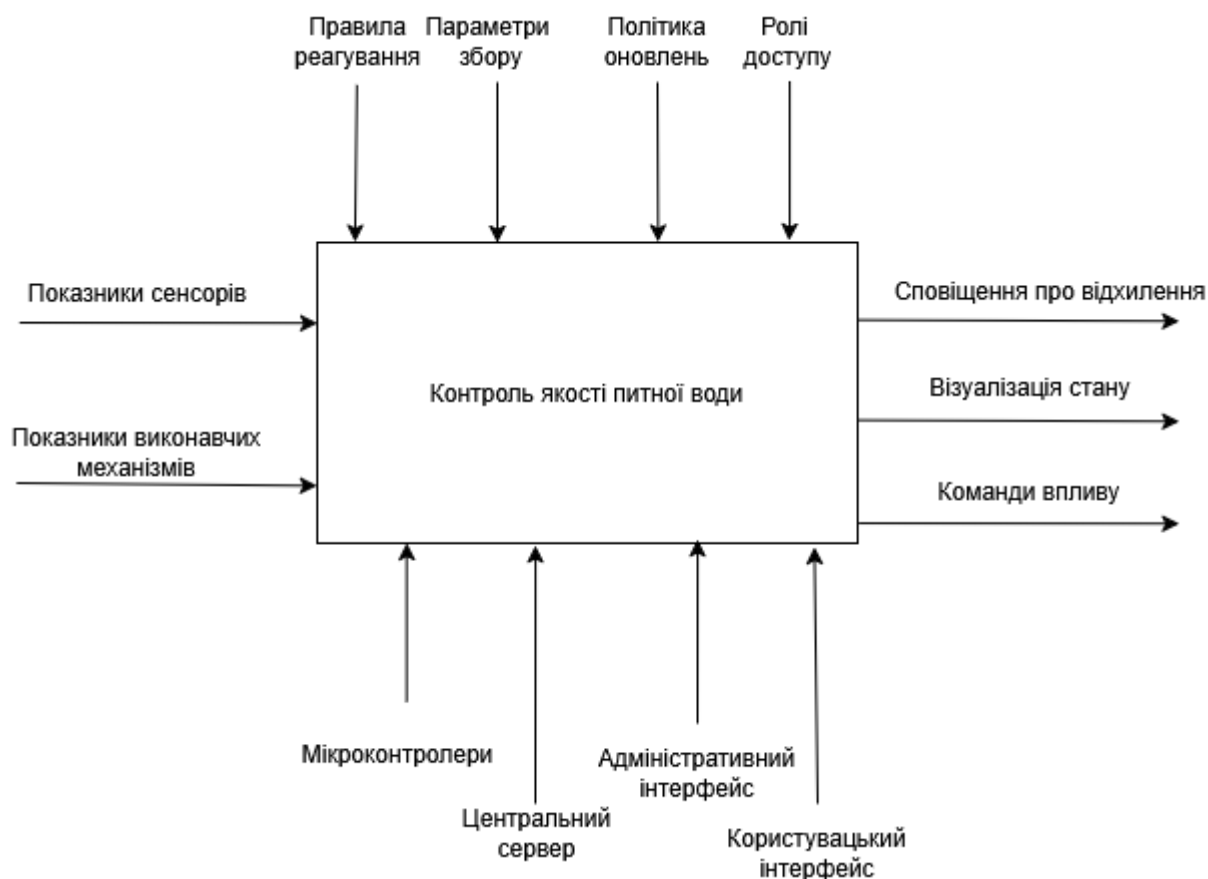


Рисунок 2.1 – «Функціональна модель IDEF0. Контекстна діаграма»

Джерело: розроблено автором

Оскільки контекстна діаграма показує найвищий рівень абстракції, то вона окреслює основну функцію системи. Але для розуміння реалізації моделі потрібно виконати декомпозицію – розбиття загальної функції на деталізовані підфункції. Декомпозиція контекстної діаграми зображена на рисунку 2.2.

Тут процес контролю якості питної води починається з точок контролю, обладнаних мікроконтролерами та підключеними до них сенсорами. Ці пристрої регулярно проводять збір фізико-хімічних показників води. Даний процес контролюється частотою збору та калібрувальними коефіцієнтами, що дозволяють компенсувати можливі похибки вимірювання. Отримані дані проходять первинну обробку та структурування на рівні мікроконтролера, що дозволить зменшити обсяг переданих даних.

Далі, ці дані передаються до локального блоку прийняття рішень. На основі збережених локальних правил реагування мікроконтролер оцінює, чи

потрібно застосувати будь-який алгоритм впливу (наприклад, увімкнути фільтрацію). При цьому враховується не лише поточний стан води, а й стан виконавчих механізмів.

Згідно з встановленою політикою оновлень, що визначає частоту та умови зв'язку, мікроконтролер надсилає накопичену інформацію на центральний сервер. У відповідь від сервера надходять оновлення конфігурації – нові правила реагування, зміна частоти збору чи калібрувальних коефіцієнтів сенсора.

На сервері отримані дані проходять глибший аналіз. Система потенційно використовувати дані для навчання або визначень тенденцій чи критичних ситуацій, що вимагають ручного втручання.

Далі, в залежності від ролі доступу, відбувається доступ до інтерфейсу. Якщо це адміністратор, то надається розгорнута візуалізація стану води, можливість оновлення правил чи втручання в роботу. Якщо адміністратор змінює параметри — ці нові правила та конфігурації надсилаються назад на мікроконтролери, де застосовуються у подальшій локальній роботі. Таким чином, система працює циклово та адаптивно, забезпечуючи автономність, реактивність і централізоване управління в одному процесі. Якщо це користувач, то надається спрощений інтерфейс, через який можна дізнатись про стан води в конкретному районі.

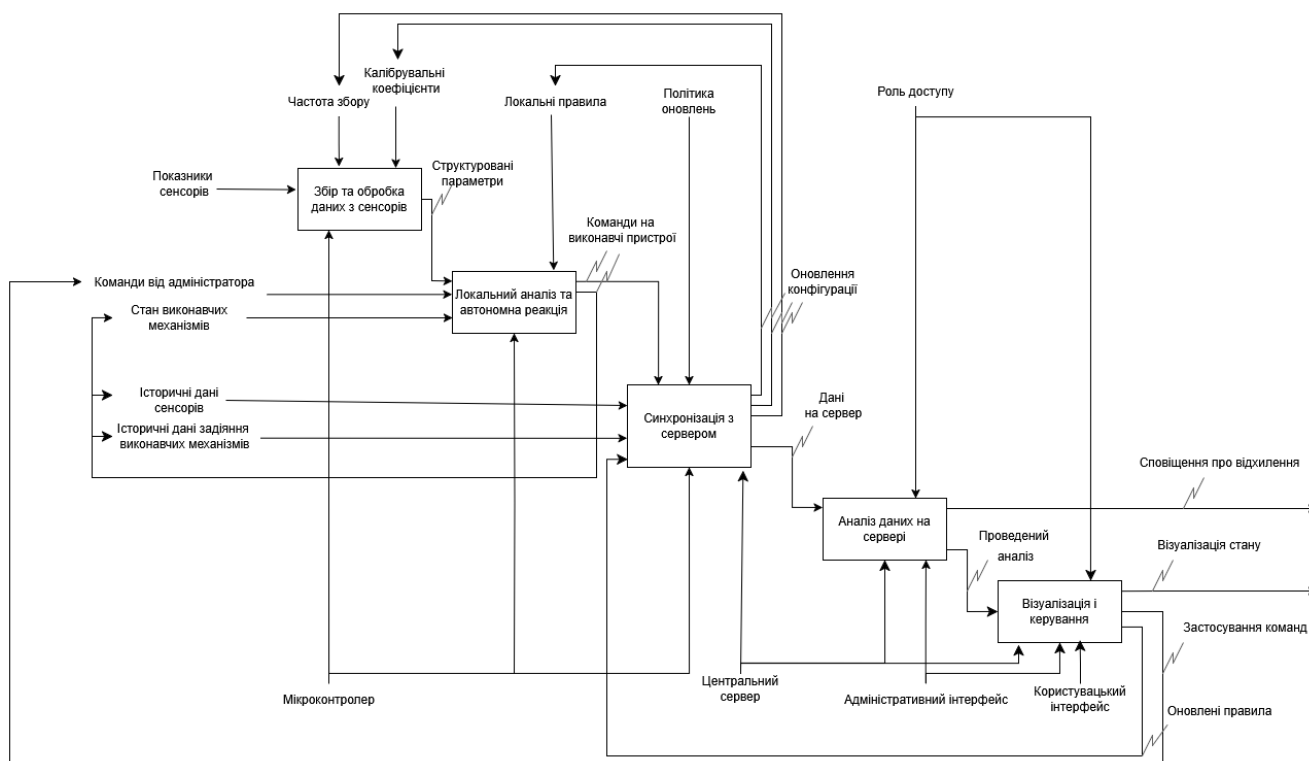


Рисунок 2.2 – «Функціональна модель IDEF0. Декомпозиція контекстної діаграми»

Джерело: розроблено автором

Більш детальна декомпозиція функції «Збір та обробка даних з сенсорів» зображена на рисунку 2.3. Тут спочатку відбувається підключення сенсорів та перевірка доступності, завантаження конфігурації. Далі, на основі вказаної в конфігурації частоти збору, відбувається очікування вказаного періоду. Після цього відбувається збір показників з фізичних сенсорів. Далі значення коригуються з урахуванням вказаної похибки. В кінці дані перетворюються у формат, придатний для подальшої обробки. Все це відбувається за допомогою механізму мікроконтролера.

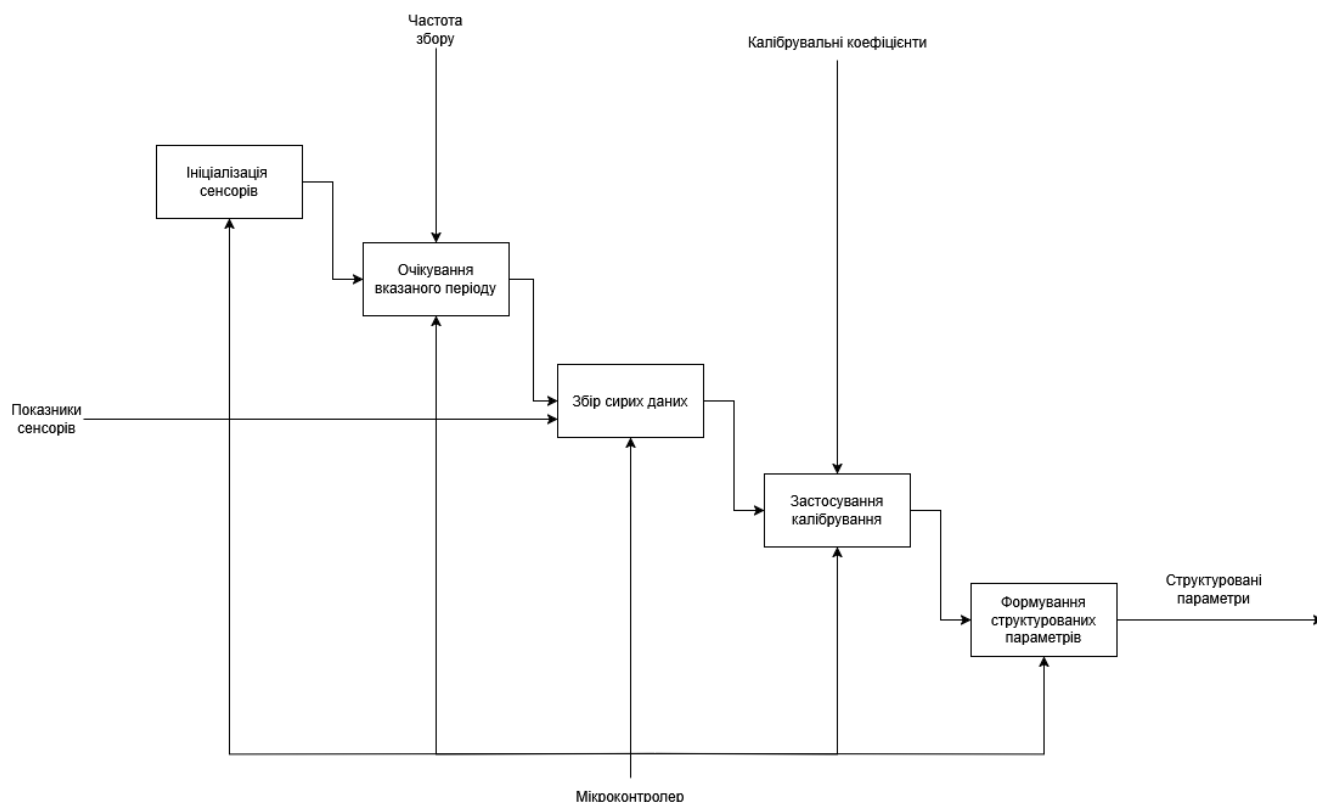


Рисунок 2.3 – «Функціональна модель IDEF0. Декомпозиція функції «Збір та обробка даних з сенсорів»»

Джерело: розроблено автором

Більш детальна декомпозиція функції «Збір та обробка даних з сенсорів» зображена на рисунку 2.4. Тут спочатку відбувається перевірка локального кешу. Далі проводиться аналіз стану води шляхом оцінки показників за правилами. Після цього відбувається перевірка чи доступні виконавчі пристрої для активації. В подальшому процес вибору необхідної дії; надсилення команд на ці пристрої. В кінці запис дій і сенсорних значень до локальної пам'яті.

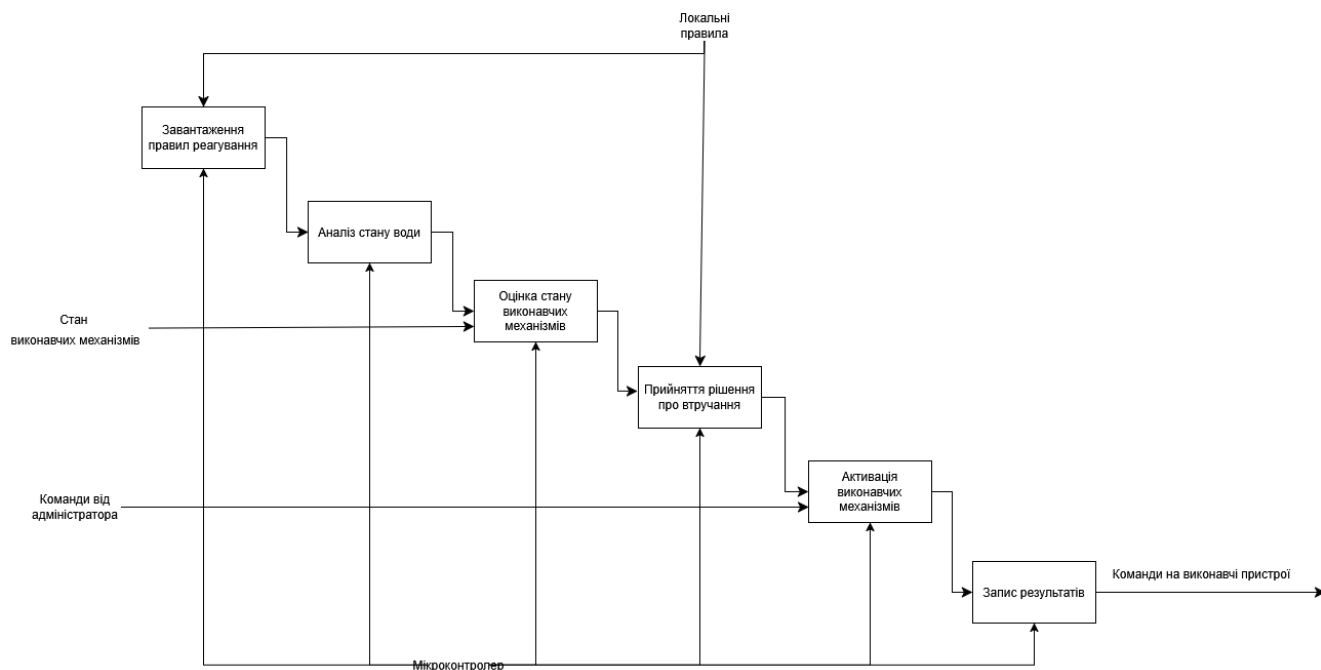


Рисунок 2.4 – «Функціональна модель IDEF0. Декомпозиція функції “Локальний аналіз та автономна реакція”»

Джерело: розроблено автором

Більш детальна декомпозиція функції «Синхронізація з сервером» зображена на рисунку 2.5. Тут спочатку відбувається збір історії сенсорів, задіяння механізмів, запитів, інших станів. Далі перевіряється чи настав час синхронізації і чи виконуються інші умови. Потім відбувається безпосередній запит на сервер та очікування відповіді. В разі отримання нової конфігурації, вона застосовується локально.

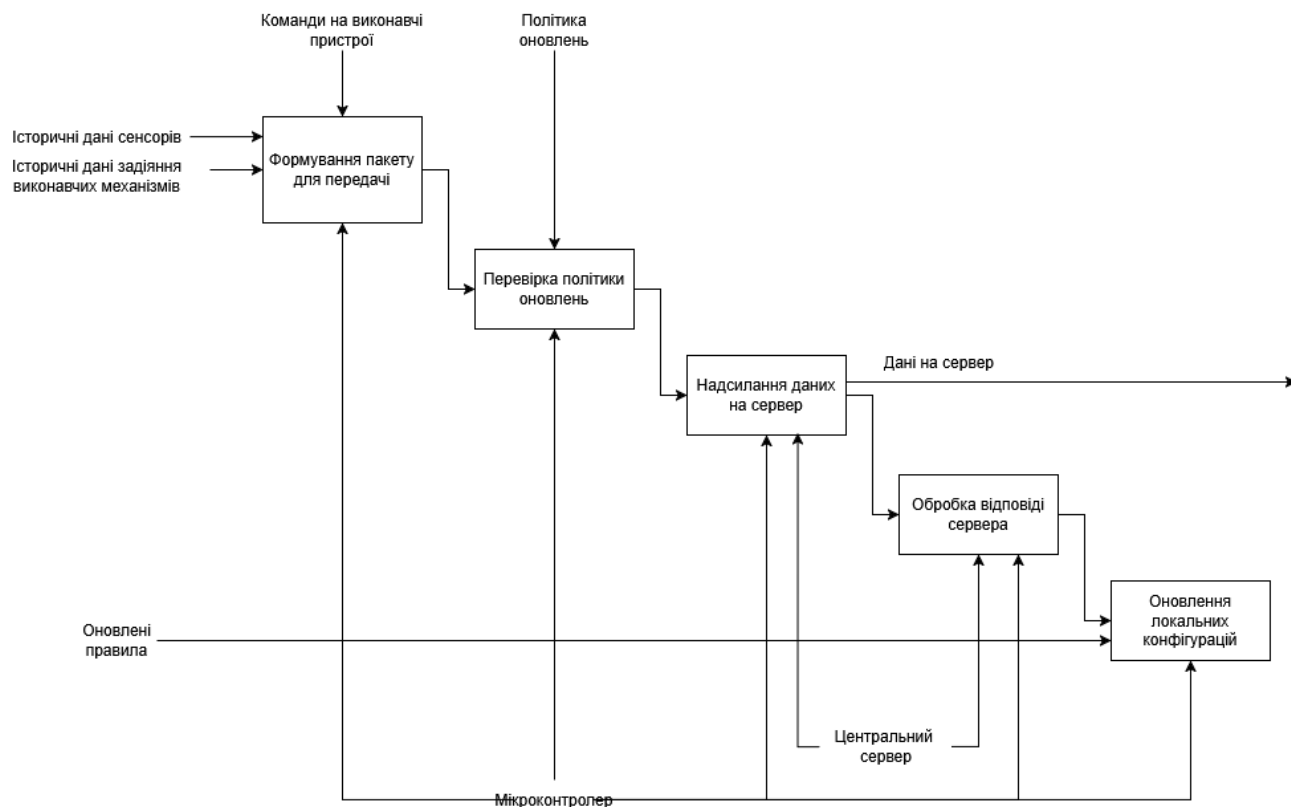


Рисунок 2.5 – «Функціональна модель IDEF0. Декомпозиція функції
«Синхронізація з сервером»»

Джерело: розроблено автором

Більш детальна декомпозиція функції «Аналіз даних на сервері» зображена на рисунку 2.6. Тут спочатку відбувається отримання, валідація даних та їх збереження у базу даних. Далі відбувається порівняння з даних з нормативами, побудова агрегацій, підсумків, узагальнень. У випадку критичних ситуацій відбувається тригер сповіщення.

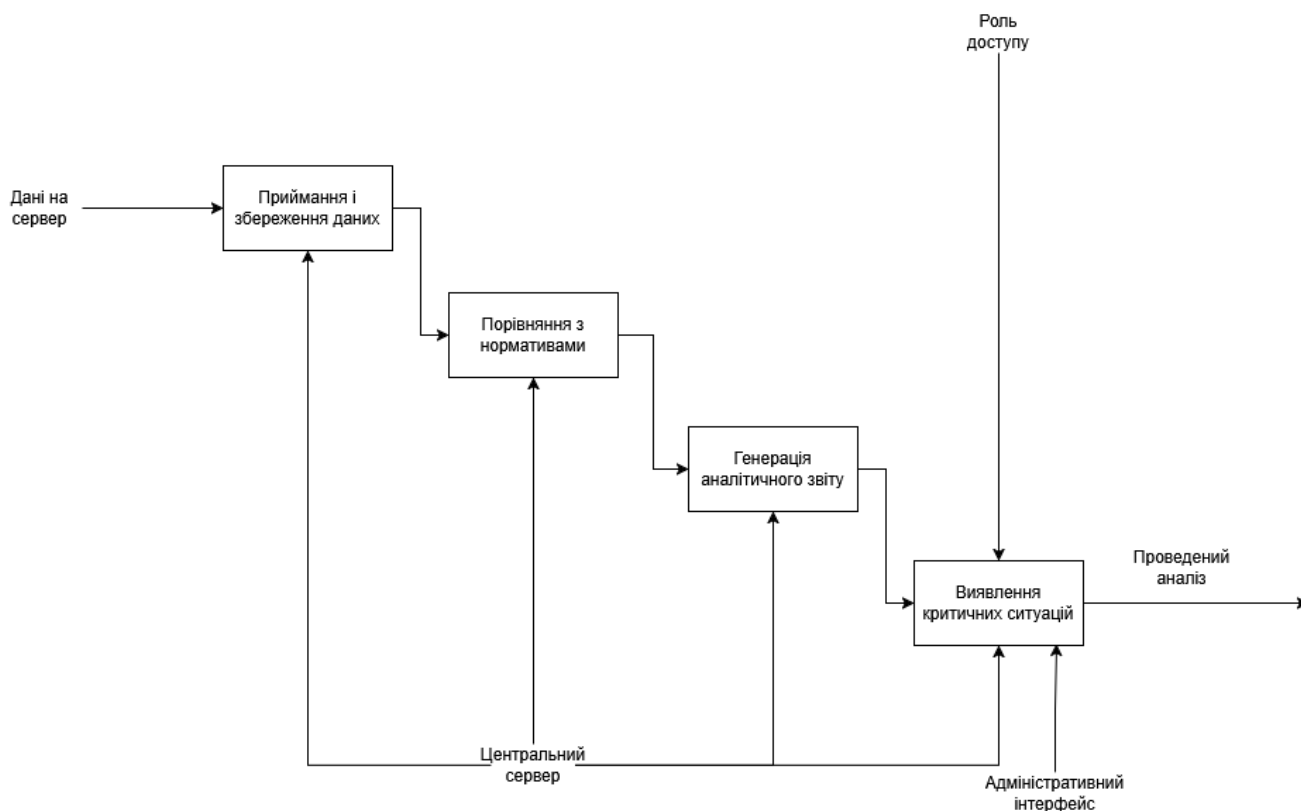


Рисунок 2.6 – «Функціональна модель IDEF0. Декомпозиція функції «Аналіз даних на сервері»»

Джерело: розроблено автором

Більш детальна декомпозиція функції «Візуалізація і керування» зображена на рисунку 2.7. Тут спочатку формується інтерфейс адміністратора або користувача, в залежності від ролі доступу. Далі, інтерфейс адміністратора має можливість ручного втручання в стан виконавчих механізмів та редагування правил для кожного сенсору. В кінці відбувається передача оновлень на сервер, які після запиту мікроконтролера будуть завантажені локально для виконання.

Command описує інструкцію, яка має бути застосована до певного пристрою (targetDeviceId) з певною інтенсивністю в заданий момент часу. Device — об'єкт, до якого застосовується команда. Містить метод applyCommand() для виконання отриманої інструкції.

Controller відповідає за локальну обробку даних: підключає сенсори (connectedSensors) і пристрої (connectedDevices), а також локальні правила (localRules). Також здійснює збір даних (collectData()), створення процедури реакцій (react()), та синхронізацію з сервером (syncWithServer()). Кожен контролер пов'язаний із логічною точкою контролю — Checkpoint.

Checkpoint Представляє географічно визначену точку контролю, з ідентифікатором, координатами, описом та прив'язкою до відповідного контролера.

Server виступає як центральний вузол обробки. Він зберігає зібрані дані (storedData), правила (rules) і користувачів (users). Метод storeSensorData() відповідає за фіксацію нових показників. analyzeData() виконує аналіз зібраної інформації з метою виявлення порушень, результатом чого є список об'єктів Alert. updateRules() та updateConfiguration() дозволяє змінювати налаштування для певного контролера чи сенсора. generateVisualization() формує графічне уявлення даних для конкретного користувача.

Сповіщення відбувається за допомогою Alert, що виникає в результаті аналізу даних, та містить часову мітку, повідомлення і пов'язані дані (SensorData). Метод notifyAdmin() відповідає за надсилання сповіщення адміністратору.

Для візуалізації використовується Visualization. Представляє графік, діаграму або інше уявлення результатів аналізу для певного користувача.

User містить роль (звичайний користувач чи адміністратор) і можливість переглядати дані (viewData()). AdminUser наслідує User та має додаткові права: редагувати правила (editRule()) і надсилати команди на пристрої (sendCommand()).

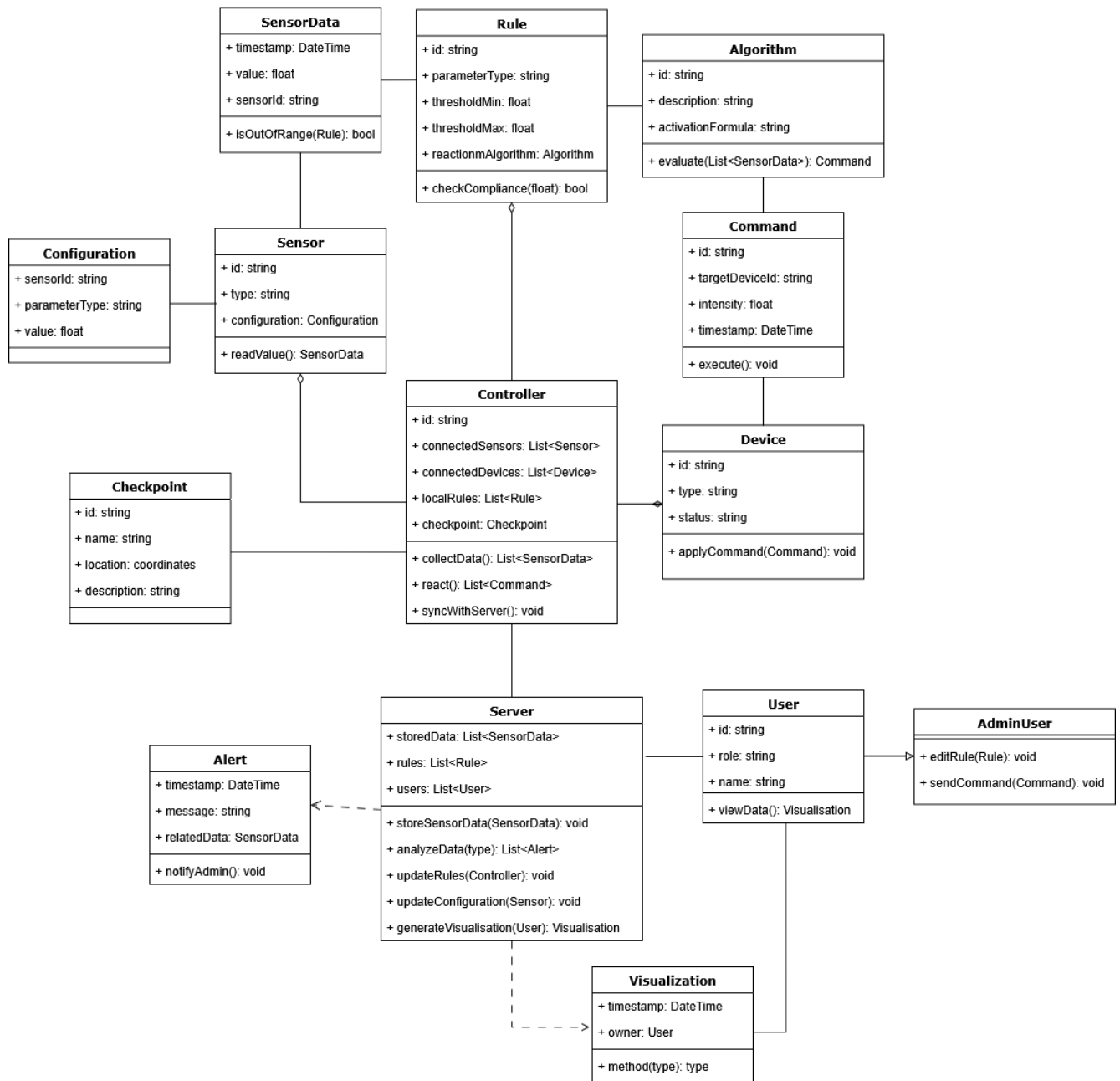


Рисунок 2.8 – «Діаграма класів»

Джерело: розроблено автором

Основні способи використання та дійові особи окреслені на діаграмі прецедентів, що зображена на рисунку 2.9. До них належать:

- Виконавчий пристрій, що здатен отримувати команди на виконання з центрального серверу(або отримувати інструкцію локально), та виконувати ці команди.

- Мікроконтролер, що здатен зчитувати дані з сенсорів; Перевіряти відповідність цих даних згідно локально збережених правил; Застосовувати локальні інструкції з відповідними командами, в залежності від відповідності даних з сенсорів правилам; Синхронізуватися з сервером згідно локально встановленої політики оновлення та отримувати в результаті оновлені правила та конфігурацію.
- Сервер, що здатен зберігати отримані дані на основі результату синхронізації з мікроконтролером та відправкою останнім історії показників сенсорів та застосування пристроїв; Глибше аналізувати отримані дані на відповідності та тренди; Створювати графічне відображення цих показників, що може включати генерацію сповіщення для адміністратора; Надсилати на мікроконтролер оновлені правила та конфігурації.
- Адміністратор, що здатен додавати(підтверджувати) нові пристрої – мікроконтролери та сенсори; Редагувати конфігурацію сенсорів, правил перевірки, політику оновлення тощо. Дані зміни будуть очікувати поки мікроконтролер не направить запит для оновлення; Надсилати ручні команди на виконавчі пристрої; Переглядати аналітику та графічне представлення стану; Отримувати сповіщення.
- Користувач, що здатен переглядати візуалізацію даних з обмеженим доступом.

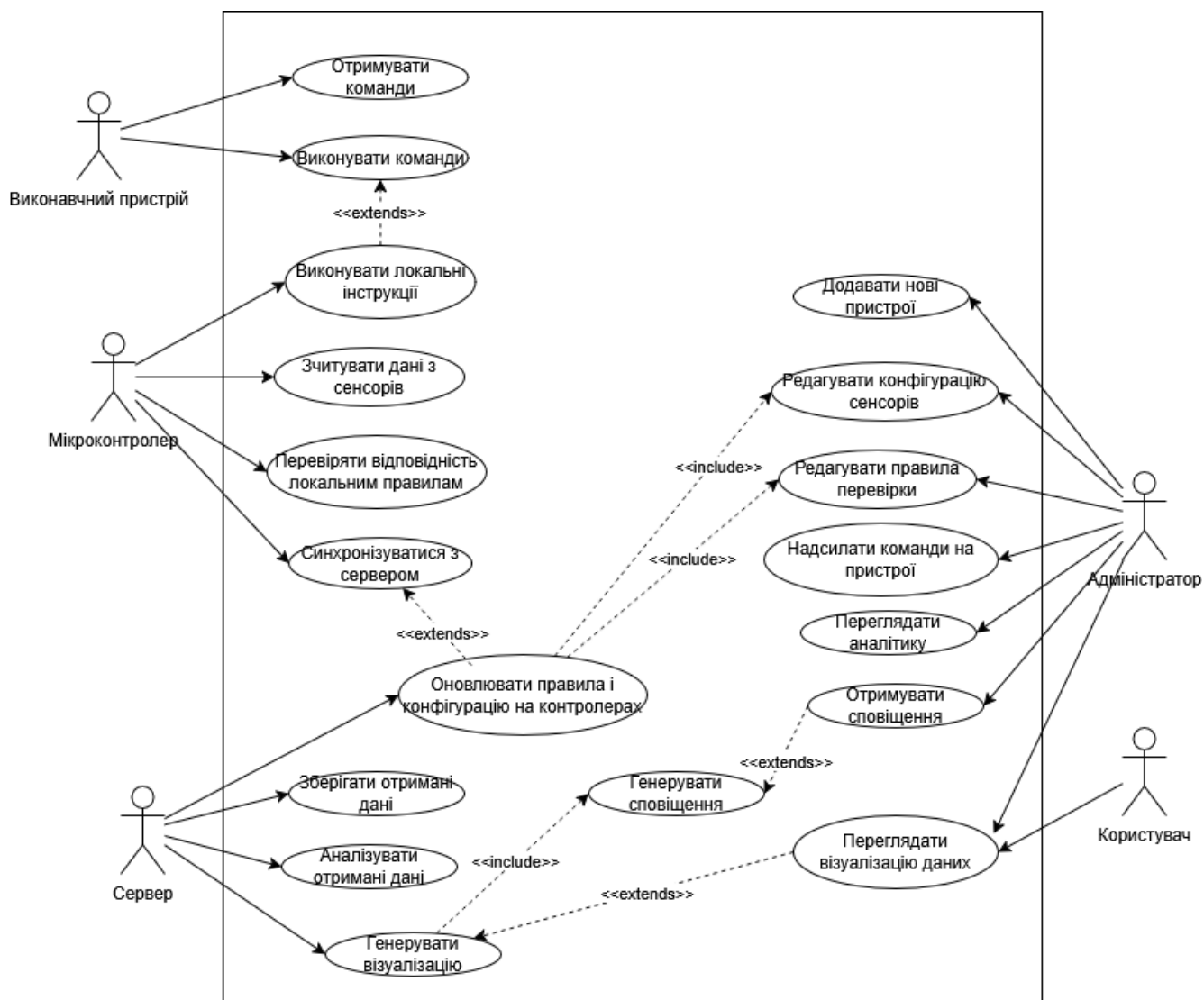


Рисунок 2.9 – «Діаграма прецедентів»

Джерело: розроблено автором

2.2. Методи та моделі в інформаційних управляючих системах і технологіях

ІУС, що функціонують у критично важливих сферах, таких як централізоване водопостачання, потребують не лише надійності й автономності, а й гнучкості у прийнятті рішень. Складність сучасних завдань зумовлює необхідність використання моделей, що поєднують як класичні методи, так і інструментарій штучного інтелекту, з можливістю адаптації до слабоструктурованих і навіть неструктурованих сценаріїв.

У контексті побудови ІУС контролю якості питної води, система повинна забезпечувати:

- аналіз структурованих показників (температура, рН, мутність, вміст хлору тощо);
- адаптацію до варіативних умов (зношеність труб, локальні аномалії, нестабільне енергоживлення);
- реакцію на складні комбінації вхідних параметрів і станів обладнання;
- мінімальне споживання енергії у всіх режимах роботи.

Для структурованих задач використовуються кількісні методи: статистичний аналіз, логічні правила, діапазонні порівняння, алгоритми визначення порушення ГДК. Реалізація передбачає локальні обчислення з використанням таблиць правил і порогів, що дозволяє забезпечити швидку та енергоефективну реакцію без необхідності постійного зв'язку з сервером.

Для слабоструктурованих ситуацій, особливо коли параметри наближаються до порогових значень, реалізується механізм «розширеного порогу» з інтерпретацією у формі зон: «норма», «небезпечне наближення», «критичне відхилення». Це дозволяє уникнути хибних спрацювань і плавно керувати виконавчими механізмами. Наприклад, при досягненні «сірого» рівня значень рН може запускатися алгоритм «м'якої фільтрації» або спостереження із підвищеною частотою збору.

Особливу роль відіграють багаторівневі правила, які враховують комбінації умов. Наприклад, «Якщо каламутність > 4 NTU і рН < 6.5 і останній запуск фільтрації був більше ніж 2 години тому, і клапан подачі реагенту відкритий менше ніж на 30%, то активувати інтенсивну фільтрацію з подвійною перевіркою через 5 хвилин.»

Такі правила можуть задаватися у вигляді вкладених умов із логічними операторами AND/OR/NOT, часом на виконання і повторного аналізу, контролем стану пристроїв тощо. Вони реалізуються у вигляді дерев правил або пріоритетних списків.

Для нестандартних умов, що не вкладаються в заздалегідь описані сценарії (наприклад, коли сенсори передають нестабільні або суперечливі дані, або при

втраті зв'язку), застосовуються резервні режими поведінки. Це може бути перехід у режим частішого моніторингу, або примусове сповіщення адміністратора. Такі підходи забезпечують гнучкість системи в умовах реальних обмежень і невизначеності.

Прогностичні методи (на основі трендів і машинного навчання) розглядаються як перспективний напрям подальшого розвитку. Потенційно, це дозволить прогнозувати зміну якості води на основі історичних даних і сезонних коливань, а також адаптивно змінювати політики контролю в залежності від контексту.

На рівні архітектури ІУС застосовано комбінований підхід: локальна обробка даних, що реалізована на мікроконтролерах, відповідає за автономну реакцію та базові сценарії, а центральний сервер забезпечує глибший аналіз, візуалізацію, корекцію конфігурацій і зберігання історичних даних. Завдяки цьому система поєднує швидкодію, автономність та масштабованість, а також відповідає вимогам щодо енергоефективності та точності управління.

РОЗДІЛ 3. РОЗРОБЛЕННЯ ПРОЄКТНИХ РІШЕНЬ ТА ЇХ РЕАЛІЗАЦІЯ

3.1 Проєктування баз даних для інформаційної управляючої системи

3.1.1 Характеристика даних та вибір моделей зберігання

У межах створення інформаційної системи контролю якості питної води передбачається зберігання двох категорій даних, що відрізняються за структурою, обсягом і частотою оновлення. Порівняння відображено у таблиці 3.1

Таблиця 3.1 – «Порівняння даних, що зберігаються, за категоріями»

Назва	Структура	Обсяг даних	Частота оновлень
Сенсори та пристрої керування	Висока	Прибл 10 полів	Прибл. 100 записів на місяць
Точки контролю	Висока	Прибл 8 полів	Прибл. 10 записів на місяць
Користувачі та права доступу	Висока	Прибл 7 полів	Прибл. 10 записів на місяць
Набір правил реагування	Слабо структуровані дані	Нефіксовано	Прибл. 30 записів на місяць
Конфігураційні параметри контролерів, сенсорів та пристроїв	Слабо структуровані дані	Нефіксовано	Прибл. 150 записів на місяць
Ручні команди виконання	Слабо структуровані дані	Нефіксовано	Прибл. 100 записів на місяць

Історія сенсорних вимірювань	Слабо структуровані дані	Нефіксовано	Прибл. 250 000 записів на місяць
Історична інформація щодо застосування правил	Слабо структуровані дані	Нефіксовано	Прибл. 200 000 записів на місяць
Журнал подій	Слабо структуровані дані	Нефіксовано	Прибл. 500 000 записів на місяць
Сповіднення	Висока	Прибл 7 полів	Прибл. 30 записів на місяць

Джерело: розроблено автором

Загалом, система використовує водночас статичні й конфігураційні дані, та динамічні, потокові та слабо структуровані дані. Тому, доцільним є використання гібридної архітектури з одночасним застосуванням реляційної та документо-орієнтованої баз даних. Для статичних і сильноструктурованих даних обрано реляційну базу даних, яка забезпечує принципи ACID [42], цілісність даних та підтримку зв'язків між сутностями. Для зберігання динамічних та слабоструктурованих даних доцільно використовувати документо-орієнтовану NoSQL [43] базу даних, яка забезпечує високу швидкість запису та читання, гнучкість у структурі документів, горизонтальне масштабування (для Big Data [44]), можливість TTL-видалення старих записів без навантаження на систему тощо [45]. Таким чином, у системі реалізується гібридна модель бази даних: реляційна для основних таблиць конфігурацій та нереляційна для зберігання великих обсягів логів і історії.

3.1.2. Проектування інфологічних моделей реляційної та нереляційної БД

На етапі інфологічного проєктування було визначено основні сутності предметної області, що належать до реляційної та нереляційної БД. До реляційної належить перелік в таблиці 3.2:

Таблиця 3.2 – «Основні сутності реляційної БД»

Сутність	Опис
Alert	Сповіщення у випадку порушення допустимих значень або невідомої ситуації
Command	Команда, яку можна надіслати контролеру для виконання пристроєм
Configuration	Параметри налаштування мікроконтролера
Device	Виконавчий пристрій, що підключений до мікроконтролера та дозволяє чинити вплив на якість води
Microcontroller	Мікроконтролер, до якого підключені сенсори й пристрої; Встановлює зв'язок з сервером та обмінюється даними
Rule	Структура умов і дій (команд) реагування
Sensor	Фізичний сенсор, закріплений у точці контролю
User	Користувач системи (адміністратор або переглядач)
ControllerSyncCache	Збір інформації, що є актуальною для надсилення мікроконтролеру
Vizualisation	Об'єкт візуалізації показників
SensorParameter	Виконує функцію «тегу». Прив'язує об'єкт сенсора до одного з допустимих типів для подальшої обробки за шаблоном
DeviceFunction	Виконує функцію «тегу». Прив'язує об'єкт пристрою до одного з допустимих типів для подальшої обробки за шаблоном

Джерело: розроблено автором

Визначені інформаційні об'єкти нормалізовано до третьої нормальної форми. Тут наявні основні сутності, що є критично важливими для роботи – Microcontroller, Sensor, Device для моніторингу стану якості води, та Command, Configuration, Rule для задання налаштувань для контролю якості води. Крім того, визначено кілька динамічних структур, що формуються під час роботи в окремих випадках, але по наповненню вони є агрегацією значень, тому прийнято рішення сутності ControllerSyncCache та Vizualisation не проєктувати як окрему сталу одиницю. Крім того, існують сутності-словники SensorParameter та DeviceFunction.

Вони виконують контролюючу функцію для встановлення значень в допустимому діапазоні для відповідних елементів:

В той же час, до нереляційної належать слабоструктуровані сутності, що можуть мати довільну кількість додаткових полів. Оскільки сенсори, пристрої та команди теоретично можуть набувати невизначених форм, а частота та обсяги їх запису та читання можуть сягати значних обсягів, то ефективніше зберігати їх в окремій БД нереляційного типу.(див. Таблицю 3.3.)

Таблиця 3.3 – «Основні сутності нереляційної БД»

Сутність	Опис
SensorLogs	Історія показників сенсорів
DeviceLogs	Історія стану та ступеня задіяння пристроїв
CommandLogs	Історія надсилання та виконання ручних команд
Alerts	Журнал подій і тривог

Джерело: розроблено автором

3.1.3 Обґрунтування вибору СКБД та фізичне проєктування

У процесі розробки інформаційної управляючої системи для моніторингу якості питної води було прийнято рішення про використання гібридної архітектури баз даних, яка поєднує реляційну модель (на базі PostgreSQL) і документо-орієнтовану модель (на базі MongoDB). Такий підхід зумовлений різною природою даних, що зберігаються та обробляються системою.

Для зберігання структурованих, стабільних за схемою конфігураційних і адміністративних даних (користувачі, сенсори, пристрої, точки контролю, правила, конфігурації) обрано PostgreSQL. Ця СКБД забезпечує [46]:

- підтримку транзакцій і дотримання вимог ACID;
- чітке типування і жорстку схему таблиць;
- можливість побудови складних зв'язків через зовнішні ключі;
- індексацію для оптимізації запитів, включно з підтримкою JSONB-типів.

Наявність типу JSONB дозволяє зберігати параметризовані структури, наприклад, складні шаблони правил, без виходу за межі реляційної СКБД. Це зручно при потребі поєднання фіксованої логіки системи з гнучкими елементами бізнес-логіки.

Для обробки динамічних, великооб'ємних, слабо структурованих або взагалі безсхемних даних (історія вимірювань, журнали, логи, події) використовується MongoDB – документо-орієнтована СКБД, що [47]:

- зберігає дані у форматі BSON (розширення JSON), що дозволяє гнучко структурувати документи;
- підтримує TTL-індекси – для автоматичного видалення застарілих логів без навантаження на систему;
- забезпечує горизонтальне масштабування та високу продуктивність на запис/читання – що критично при зберіганні великої кількості сенсорних записів у реальному часі;
- дозволяє зберігати логічно різноманітні об'єкти в рамках однієї колекції – наприклад, події з різними структурами для пристроїв, сенсорів або команд.

MongoDB також забезпечує гнучкий підхід до моделювання історії подій, в якому можуть змінюватися поля в залежності від типу джерела. Це суттєво спрощує розробку та усуває потребу у денормалізації при збереженні великих обсягів даних.

Фізичне проєктування обох моделей виконано у відповідному середовищі. ERD-схема реляційної частини створена в pgModeler (рисунок 3.1). Сюди належать наступні таблиці:

- Microcontroller, що являє собою сутність контролера. Має поля для ідентифікації, зв'язку з конфігурацією, контрольною точкою та станом актуальності;

- Configuration, що являє собою конфігурацію мікроконтролера з відповідними полями. Може бути перевикористано для інших мікроконтролерів;
- Checkpoint, що являє собою фізичне місце контрольної точки мікроконтролера. Може включати кілька контролерів.
- Rule template, що являє собою шаблон правила з умовами та інструкціями;
- Rule assignment, що являє собою конкретизацію шаблону правила під конкретний мікроконтролер, використовуючи систему параметрів та функцій «тегів»;
- Device, що являє собою зареєстрований мікроконтролером виконавчий пристрій для впливу на якість води. Має поле рівня активації;
- Device Function, що являє собою перелік можливих функцій для пристрою. Використовується для параметризації в Rule assignment
- Sensor, що являє собою зареєстрований мікроконтролером сенсор, що збирає дані про характеристики води;
- Sensor Parameter, що являє собою перелік можливих параметрів для сенсорів. Використовується для параметризації в Rule assignment
- Command, що являє собою елемент ручного керування мікроконтролером. Це команда для виконання виконавчим пристроєм або сенсором;
- Alert, що являє собою сповіщення адміністратора через інтерфейс. Викликається у виключних ситуаціях, або при появі нових незареєстрованих пристроїв;
- User, що використовується для авторизації адміністратора.

Структура документів і колекцій MongoDB – згідно з моделями, що є описаними в таблиці 3.3, створена в Haskolade (рисунок 3.2).

Таке поєднання дозволяє оптимально розподілити навантаження між двома типами СКБД і гарантувати баланс між строгістю, надійністю та масштабованістю.

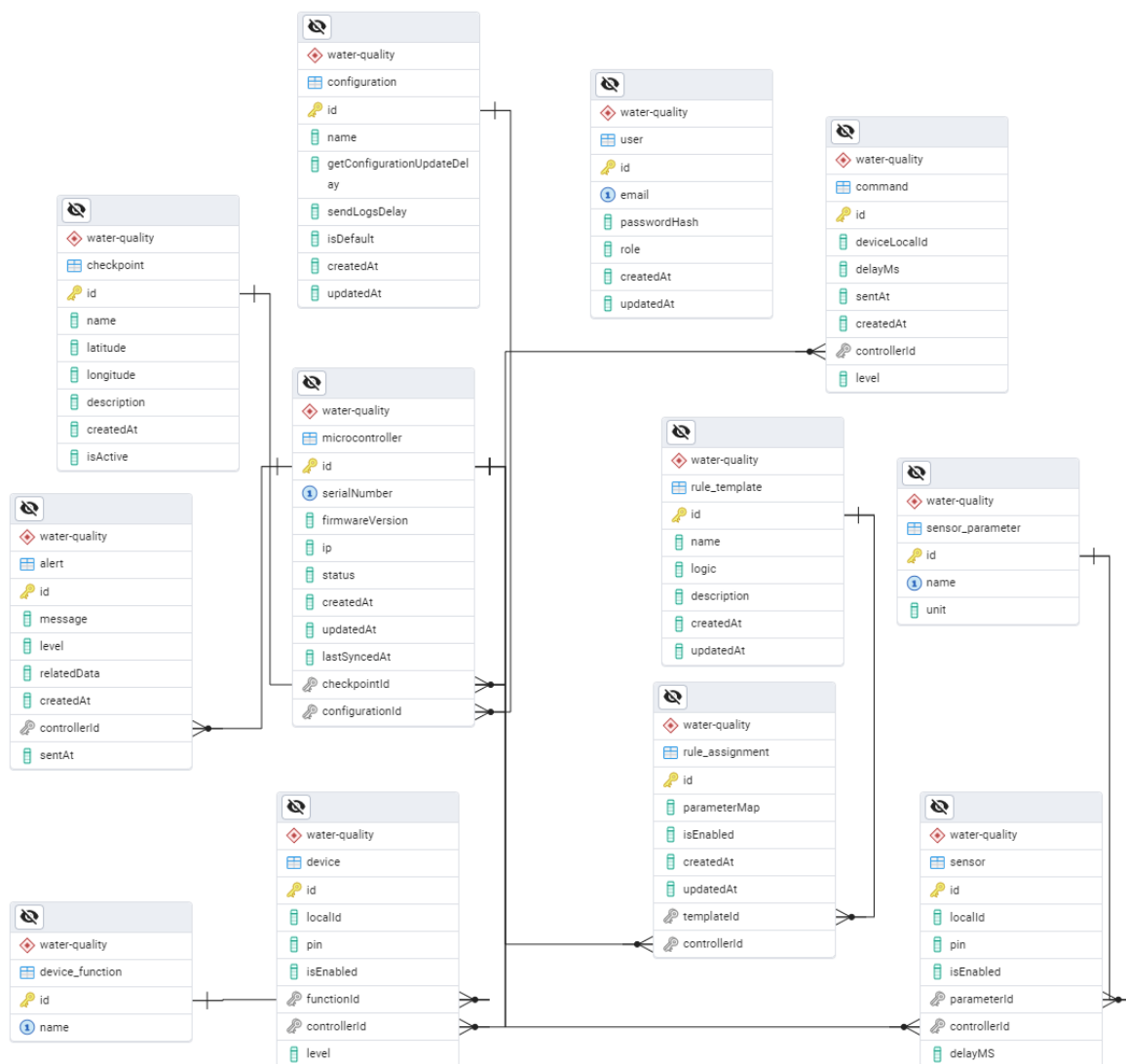


Рисунок 3.1 – «ERD діаграма сутностей в СКБД Postgres»

Джерело: розроблено автором

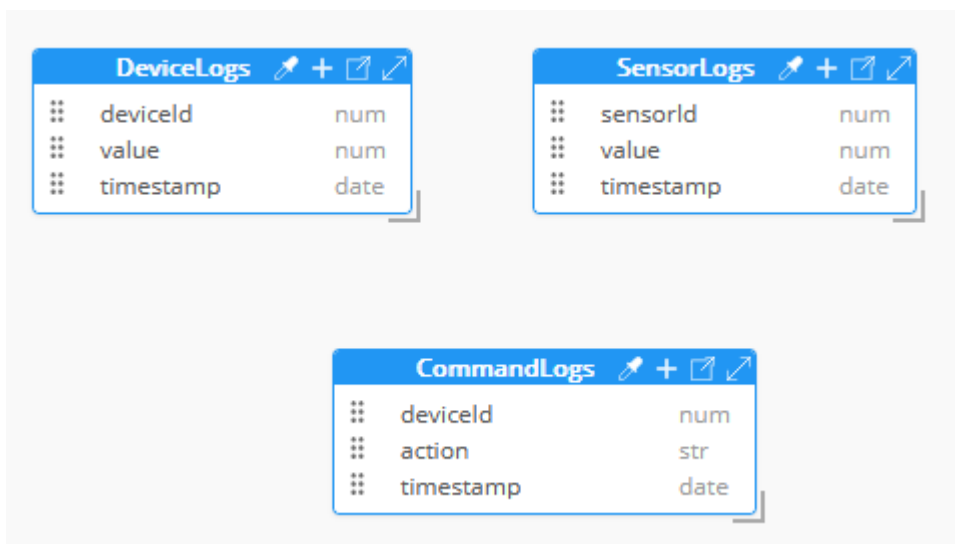


Рисунок 3.3 – «Контрольний приклад БД. Перелік сенсорів, що прив’язані до контролера»

Джерело: розроблено автором

- Отримати перелік всіх сповіщень, що були надіслані з певного мікроконтролера(див. Рисунок 3.3). Результатом є повідомлення, рівень критичності, пов’язані дані, та інформація для ідентифікації мікроконтролера та стану сповіщення.

```

167 SELECT mc."serialNumber",
168 mc."ip",
169 al."createdAt",
170 mc."lastSyncedAt",
171 al."level",
172 al."relatedData",
173 al."acknowledged"
174 FROM "water-quality".alert al
175 LEFT JOIN "water-quality".microcontroller mc ON al."controllerId" = mc.id
176 WHERE "controllerId" = 2;
177

```

	serialNumber character varying	ip character varying	createdAt timestamp without time zone	lastSyncedAt timestamp without time zone	level character varying	relatedData jsonb	acknowledged boolean
1	esp32-ABC124	192.168.0.13	2025-05-13 16:10:46.84417	[null]	CRITICAL	("pH": 13.3, "temperature": 15.3)	false
2	esp32-ABC124	192.168.0.13	2025-05-13 16:10:46.84417	[null]	WARNING	("body": "", "header": "http", "response": "500")	false

Рисунок 3.3 – «Контрольний приклад БД. Перелік сповіщень з мікроконтролера»

Джерело: розроблено автором

- Отримати перелік історичних даних з сенсорів за певний період (див. Рисунок 3.4). Результатом є дані з ідентифікатором сенсору, значення, часовий запис. За потреби додавання інформаційних полів з реляційної БД.

```

{
  "timestamp": { "$gt": ISODate("2025-05-12T16:00:00Z") }
}

```

<pre> { "_id": ObjectId("68234f244069ab1d185cc34"), "sensorId": 1, "value": 6.8, "unit": "pH", "timestamp": 2025-05-13T19:41:00.000+00:00, ...v: 0 } </pre>
<pre> { "_id": ObjectId("68234f244069ab1d185cc35"), "sensorId": 1214, "value": 24.5, "unit": "°C", "timestamp": 2025-05-13T13:54:44.815+00:00, ...v: 0 } </pre>
<pre> { "_id": ObjectId("682350ba4069ab1d185cc37"), "sensorId": 1, "value": 6.8, "unit": "pH", "timestamp": 2025-05-12T19:41:00.000+00:00, ...v: 0 } </pre>
<pre> { "_id": ObjectId("682350ba4069ab1d185cc38"), "sensorId": 1214, "value": 24.5, "unit": "°C", "timestamp": 2025-05-13T14:01:30.313+00:00, ...v: 0 } </pre>

Рисунок 3.4 – «Контрольний приклад БД. Перелік історичних даних з сенсорів за певний період»

Джерело: розроблено автором

Прогнозований обсяг використання реляційної бази даних включає основні таблиці Microcontroller, Sensor, Device, Rule, Configuration, Command, User, Alert, SensorParameter, DeviceFunction. Допустимо припущення, що кількість точок контролю: 100; кожна точка має 5 сенсорів і 3 пристрої; кожен мікроконтролер має ~10 правил, 1 конфігурацію, і ~2 команди в тиждень; Кожен сенсор має 1 параметр, кожен пристрій - 1 функцію; Загалом 10 користувачів; Сповіщення - 10 в день по системі. Прогнозовані обсяги зображені в таблиці 3.3.

Таблиця 3.3 – «Прогнозовані обсяги використання реляційної БД (в місяць)»

Таблиця	Кількість записів	Розмір одного запису	Обсяг файлу
Microcontroller	100 записів/місяць	1 КБ	100 КБ
Sensor	500 записів/місяць	1 КБ	500 КБ
Device	300 записів/місяць	1 КБ	300 КБ
Rule	2500 записів/місяць	2 КБ (може зберігатись JSON)	5 МБ
Configuration	100 записів/місяць	2 КБ	200 КБ
Command	1000 записів/місяць	1 КБ	1 МБ
User	10 записів/місяць	1 КБ	10 КБ
Alert	30 записів/місяць	1.5 КБ	3 МБ
SensorParameter	10 записів/місяць	0.25 КБ	2,5 КБ
DeviceFunction	10 записів/місяць	0.25 КБ	2,5 КБ

Джерело: розроблено автором

Загалом, Загальний прогнозований обсяг ~9–10 МБ, або 7000-9000 тис. запитів за час використання. При цьому основне використання очікується при

старті та ініціалізації мікроконтролерів, сенсорів, пристроїв та правил. Такий розмір не потребує масштабування

Щодо нереляційної БД, то в даному випадку основними колекціями є SensorLogs, DeviceLogs, CommandLogs, Actions (опційно). Формат типового введення зображений на рисунку 3.5, Він приблизно оцінюється в ~100 байт на запис.

```
{
  "sensorId": "sensor_001",
  "value": 7.13,
  "timestamp": "2025-05-12T12:30:00Z",
  "unit": "pH",
  "status": "ok"
}
```

Рисунок 3.5 – «Приклад введення значень для показників сенсора»
Джерело: розроблено автором

Таблиця 3.4 – «Прогнозовані обсяги використання нереляційної БД (в місяць)»

Таблиця	Кількість записів	Розмір одного запису	Обсяг файлу
DeviceLogs	36 000 записів/день	1 КБ	1.08 ТБ
CommandLogs	7 записів/день	2 КБ	420 КБ
Actions	250 000 записів/день	3 КБ	22.5 ТБ

Джерело: розроблено автором

Таким чином, отримані прогнозовані обсяги дають зрозуміти, що за даних умов використання шардінгу не є обов’язковим, проте варто передбачити TTL або архівування SensorLogs щонайменше раз на 6 місяців, щоб уникнути деградації продуктивності.

3.2 Проектування бази знань інформаційної управляючої системи

З метою реалізації адаптивного керування процесами очищення та контролю якості питної води у складі інформаційної управляючої системи

реалізовано концепцію бази знань, що містить формалізовані логіко-правильні структури прийняття рішень. Така база знань дозволяє не лише виконувати реактивне реагування на порушення граничних значень параметрів, а й реалізувати складні сценарії поведінки на основі поєднання умов, контексту та багаторівневих логічних операторів.

Формування правил у системі ґрунтується на сукупності нормативних, технологічних і емпіричних передумов. Основу для створення логіко-правильних структур становлять нормативні документи. Це може бути як нормативно-правова база у сфері питного водопостачання, так і локальні технологічні регламенти. Ці документи визначають межі допустимих значень параметрів, які трансформуються у умови у правилах системи (наприклад, «Значення водневого показника більше за 6.5" або "Мутність води менше або дорівнює 5").

Другим важливим джерелом є реальні технологічні схеми об'єктів очищення та постачання води. Вони визначають можливі виконавчі дії (наприклад, активація фільтра, відкриття клапану, запуск дозатора реагенту) та взаємозв'язок між параметрами (наприклад, підвищення температури призводить до зниження розчиненого кисню). Ці залежності реалізуються у діях блоку, що вказують, які пристрої активуються при порушенні чи стабілізації параметрів.

Третім джерелом є історичні логи вимірювань та журнали тривог, які накопичуються в ході експлуатації системи. На основі цих даних аналітики та технічні спеціалісти можуть ідентифікувати типові ситуації (наприклад, нічне падіння тиску або сезонне коливання водневого позканика з виявленням трендів відхилень, які передують критичним подіям. Така діяльність дозволить формувати прогностичні або превентивні правила, які реагують до появи фактичної аварії.

У перспективі ці емпіричні підходи можуть бути автоматизовані за допомогою алгоритмів машинного навчання, що сформуєть адаптивні або самоналаштовувані шаблони правил.

Визначальну роль у формуванні правил відіграє також фахова експертиза інженерів, екологів, санітарних лікарів тощо. Вони можуть проводити аналіз особливостей місцевої системи водопостачання та створювати основи правил під

конкретні виробничі потреби. Адміністратор системи через веб-інтерфейс має змогу додати шаблон правила в систему, задати порогові значення, вказати порядок дій та визначити логіку поведінки після стабілізації або при посиленні відхилення.

Правило в системі подається у вигляді вкладеної логічної структури з умовами (if), діями (then), альтернативними гілками (else, else if) та подальшим контролем (after). Такий підхід дозволяє враховувати не лише миттєві відхилення параметрів, а й побудову контекстно-залежних шаблонів поведінки. Формат зберігання правил обрано у вигляді JSON-документів, що надає можливість гнучко структурувати вкладені умови та дії, динамічно інтерпретувати логіку без жорсткої прив'язки до схеми, реалізувати перевикористання та розширюваність шаблонів без зміни структури бази(див. Рисунок 3.6).

```
"if": {
  "all": [
    { "op": ">", "value": 32, "sensor": "temperature" },
    { "op": "<", "value": 10, "sensor": "humidity" },
    { "op": "<", "value": 5.5, "sensor": "ph" }
  ],
  "then": [
    { "device": "heater", "command": "set", "value": 0.3 },
    { "device": "phFilter", "command": "set", "value": 0 },
    { "command": "delay", "duration": 300 }
  ],
  "after": {
    "if": {
      "all": [
        { "op": "<", "value": 30, "sensor": "temperature" },
        { "op": ">=", "value": 10, "sensor": "humidity" },
        { "op": "between", "range": [5.5, 8.5], "sensor": "ph" }
      ]
    },
    "then": [
      { "command": "alert", "message": "DANGER" }
    ],
    "else": [
      { "command": "alert", "message": "UNKNOWN STATE" }
    ]
  }
},
```

Рисунок 3.6 – «Приклад базової логічної структури правила»

Джерело: розроблено автором

З метою забезпечення універсальності шаблонів, у структурі правил замість конкретних ідентифікаторів сенсорів та пристроїв використовуються теги (наприклад, "temperature", "ph", "heater"). Це дозволяє створювати багаторазово придатні шаблони правил, що надалі можуть бути прив'язані до конкретної апаратної конфігурації. Відповідність між тегами та конкретними пристроями задається за допомогою окремої сутності(див. Рисунок 3.7).

```
{
  "temperature": "tempSensor_01",
  "ph": "phSensor_22",
  "heater": "heaterDevice_3",
  "phFilter": "phNeutralizer_7"
}
```

Рисунок 3.7 – «Приклад базового мапінгу параметрів сенсора»

Джерело: розроблено автором

Таким чином, одне й те саме правило може бути використано для десятків мікроконтролерів з різними апаратними налаштуваннями, що забезпечує масштабованість та адаптивність системи.

3.3 Програмне забезпечення

3.3.1 Проєктування програмного забезпечення

Процес створення ІУС контролю якості питної води потребує ретельного проєктування програмного забезпечення, що відповідає сучасним вимогам до систем подібного класу. Основним завданням на етапі розроблення є формування чіткої бізнес-логіки роботи системи, що забезпечує виконання функціональних та нефункціональних вимог, до яких можна віднести системність, комплексність, відкритість, адаптивність, надійність, безпечність, мобільність, простоту в обслуговуванні й навчанні персоналу, а також легкість підтримки та подальшого її супроводу.

Враховуючи складність предметної області та необхідність гнучкості в управлінні бізнес-логікою, у процесі проєктування програмного забезпечення обрано методологію розробки на основі архітектурних патернів Model-View-Controller (MVC) та Model-View-ViewModel (MVVM).

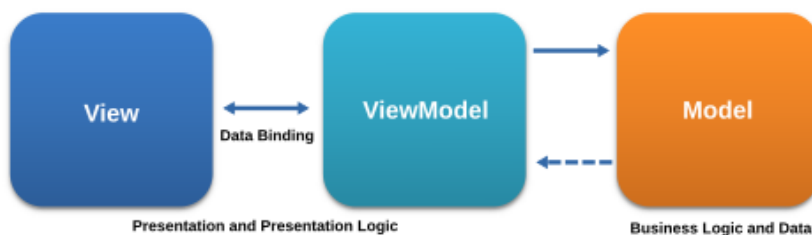


Рисунок 3.8 – «Паттерн MVVC»

Джерело: : використане інтернет-джерело [61]

Архітектурний патерн MVC було обрано для серверного компонента системи, реалізація якого планується за допомогою сучасного фреймворка NestJS [48], побудованого на основі платформи Node.js та мови програмування TypeScript (див. Рисунок 3.9). Вибір саме цього технологічного стека зумовлено низкою переваг, серед яких висока продуктивність у реальному часі, широкі можливості для асинхронної обробки запитів, наявність гнучких механізмів масштабування та інтеграції з різними базами даних, а також велика підтримка від спільноти розробників, що дозволяє ефективно підтримувати та розширювати систему в майбутньому. NestJS має вбудовану підтримку принципів SOLID, що додатково сприяє чіткому розподілу відповідальності між компонентами системи, зменшуючи ризик виникнення технічного боргу. Сервер реалізовано згідно REST, що дозволяє виконувати операції навіть без інтерфейсу.

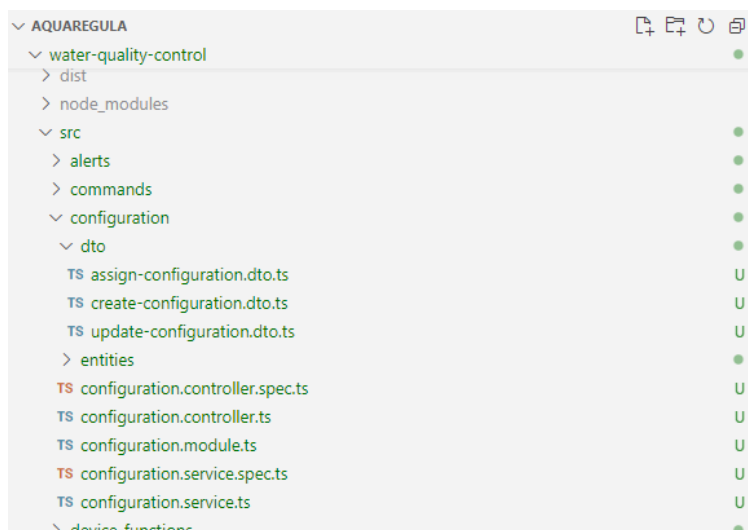


Рисунок 3.9 – «Архітектура проєкту серверної компоненти»

Джерело: розроблено автором

Для клієнтської частини системи, що передбачає взаємодію з адміністратором і кінцевими користувачами, обрано архітектурний патерн MVVM, реалізований за допомогою JavaScript-фреймворка Next.js [49] на основі бібліотеки

React(див. Рисунок 3.10). Такий вибір зумовлено перевагами, які надає Next.js у розробці клієнтських веб-застосунків: швидкий рендеринг сторінок завдяки технології SSR, висока продуктивність, ефективне кешування статичних даних та зручна інтеграція з RESTful API, що значно покращує досвід користувача.

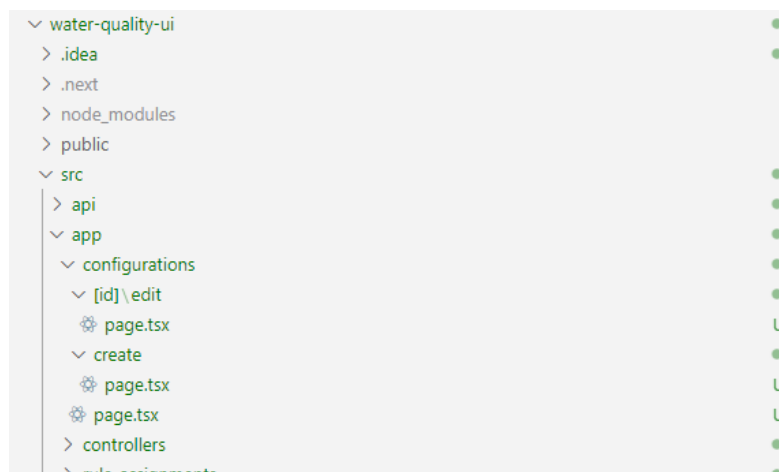


Рисунок 3.10 – «Архітектура проєкту клієнтської компоненти»

Джерело: розроблено автором

Сам MVVM патерн забезпечує додаткове відокремлення бізнес-логіки (моделі та View-Model) від рівня відображення (View). Проміжний шар View-Model є реактивним і відповідає за перетворення даних, отриманих від центрального серверу, в оптимальну форму для їх подальшого представлення користувачу, що спрощує процедуру розробки, тестування, та подальше масштабування клієнтського додатку.

Зважаючи на сучасні стандарти розроблення програмного забезпечення та архітектурні вимоги до інформаційних систем, ІУС спроектовано з використанням принципу розподіленої архітектури, що складається з серверного (Backend) та клієнтського (Frontend) компонентів, які взаємодіють через RESTful API(див. Рисунок 3.11). Серверна частина забезпечує централізовану обробку даних, виконання бізнес-логіки, збереження конфігурацій та реалізацію складних процедур реагування. Клієнтський компонент забезпечує зручний користувацький інтерфейс для моніторингу стану системи, керування правилами реагування та візуалізації інформації в інтерактивному форматі.

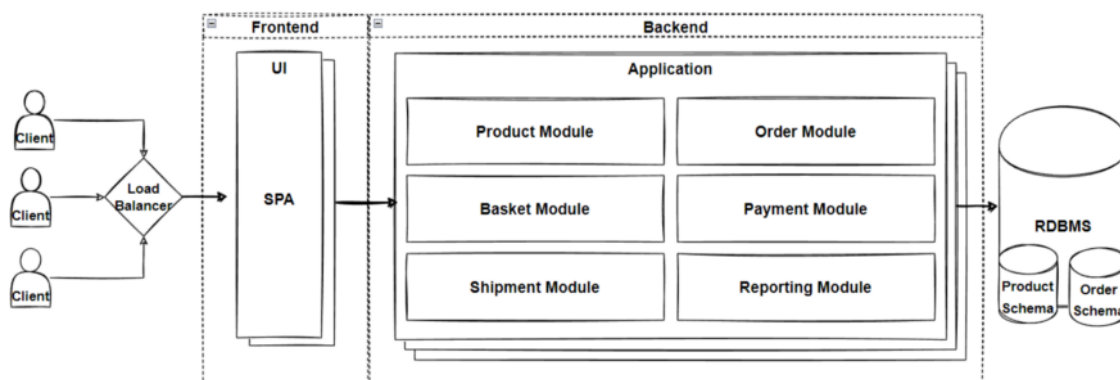


Рисунок 3.11 – «Архітектура розділеної серверної та клієнтської компоненти»
Джерело: : використане інтернет-джерело [62]

Вибір мови програмування TypeScript для розробки обох компонентів забезпечує типізацію, ранню перевірку помилок на етапі компіляції, покращену підтримку інтегрованих середовищ розробки (IDE), зокрема Visual Studio Code, що спрощує процес розробки, налагодження і підтримки програмного продукту. При цьому мова активно розвивається, має широку підтримку та значну кількість бібліотек для подальшої інтеграції нових механік.

Було звернуто увагу на вибір структурних шаблонів програмування (design patterns), які забезпечують чітке розмежування відповідальності між компонентами програмного забезпечення. У серверній компоненті активно використовується патерн Dependency Injection (інверсія залежностей), реалізований в NestJS, що дозволяє легко управляти залежностями компонентів, здійснювати модульне тестування, розширювати функціональність без значних змін у кодовій базі. Для клієнтського компонента на основі Next.js і React застосовуються патерни Composition, Observer та Singleton, що забезпечують реактивну взаємодію компонентів, простоту підтримки та подальше масштабування.

Окрему увагу в рамках програмного забезпечення ІУС приділено реалізації вбудованого програмного компонента для периферійних мікроконтролерів ESP32. Програмне забезпечення контролерів розробляється з використанням мови програмування C++ у середовищі розробки PlatformIO (див. Рисунок 3.12), що забезпечує ефективність роботи пристроїв, високу продуктивність та зручність підтримки [50]. Контролери мають виконувати ряд ключових завдань: реєстрацію

у центральному сервері, отримання та інтерпретацію правил, збереження й періодичне передавання на сервер журналів роботи, а також періодичну перевірку наявності оновлень конфігурації та правил. Цей компонент програмного забезпечення забезпечує реалізацію гнучкої логіки локального керування, що доповнює централізовану бізнес-логіку, визначену на серверному рівні.

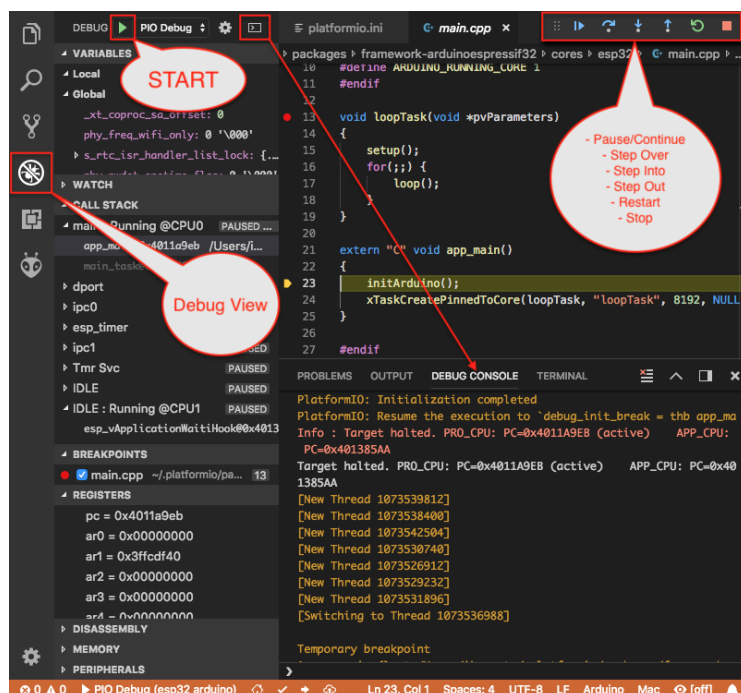


Рисунок 3.12 – «Інтерфейс PlatformIO»

Джерело: використане інтернет-джерело [51]

3.3.2 Вибір архітектури програмного забезпечення

Сучасні інформаційні системи можуть бути реалізовані як у вигляді монолітних, так і розподілених рішень. Монолітна архітектура, попри свою простоту на початкових етапах розробки, має значні обмеження у масштабуванні, оновленні, гнучкості та підтримці у разі зростання функціональних потреб системи. Розподілена архітектура, зокрема використання мікросервісного підходу, надає значно ширші можливості для розвитку, підтримки та інтеграції окремих компонентів системи, що особливо важливо з огляду на специфіку завдання, складність логіки реагування та можливість майбутнього розширення системи додатковими сервісами або модулями.

З огляду на ці особливості, обґрунтованим рішенням є використання розподіленої (мікросервісної) архітектури для реалізації інформаційної управляючої системи (див. Рисунок 3.13). В межах даного проекту серверна частина, реалізована за допомогою NestJS, функціонує як самостійний мікросервіс, що забезпечує обробку, аналіз, зберігання даних та взаємодію із зовнішніми клієнтами через RESTful API. Клієнтська частина, що реалізується за допомогою фреймворка Next.js, розташовується на окремому сервері й відповідає за графічний інтерфейс взаємодії з адміністратором і користувачами системи. Така архітектура дозволяє максимально ефективно розподіляти ресурси, спрощує розгортання оновлень, забезпечує високу надійність, адаптивність та відкритість системи, що відповідає основним принципам побудови сучасних інформаційних управляючих систем.

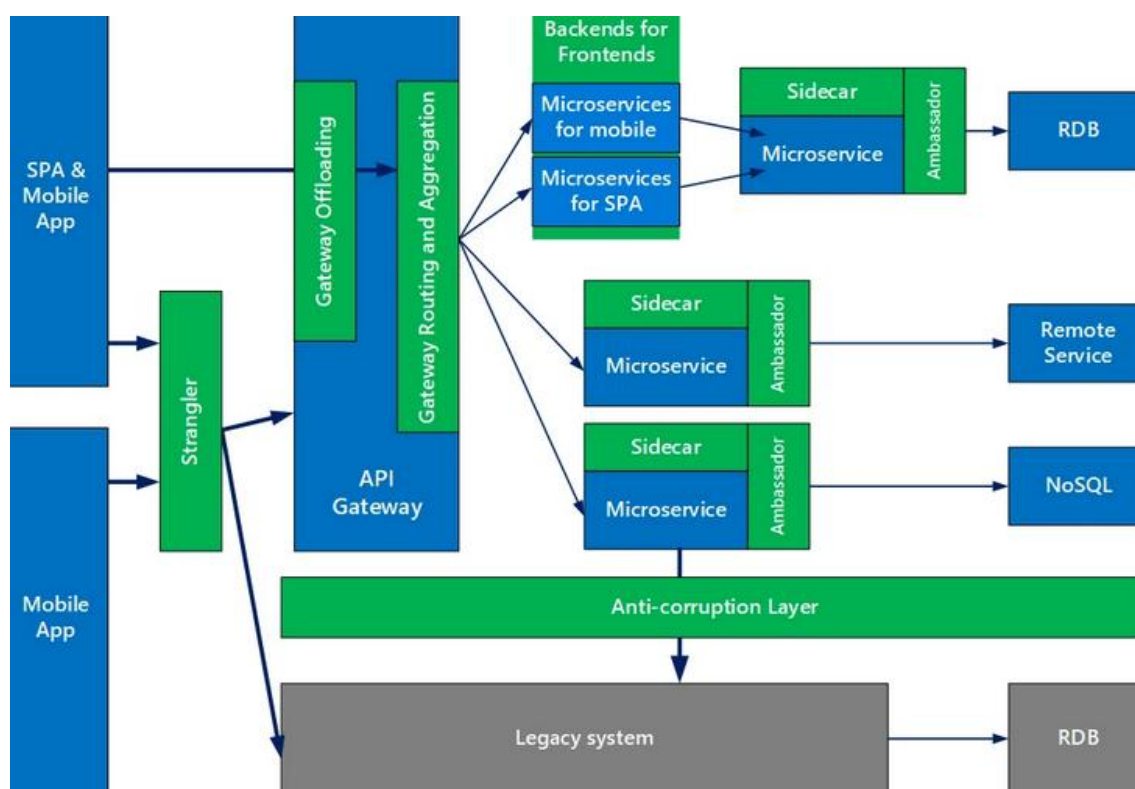


Рисунок 3.13 – «Ілюстрація мікросервісної архітектури»

Джерело: використане інтернет-джерело [63]

Вибір розподіленої архітектури тісно пов'язаний із питаннями корпоративної безпеки, ідентифікації та авторизації користувачів. Для забезпечення надійності та безпечності обміну інформацією між сервером і клієнтом у межах розподіленого середовища, в даній системі застосовується криптографічний протокол SSL/TLS, який гарантує цілісність та конфіденційність даних при їх передачі через мережу [51].

Окрему увагу приділено питанням аутентифікації та авторизації користувачів. Оскільки система є розподіленою, класичні методи аутентифікації на основі сесій (sessions) та cookie-файлів не забезпечують достатнього рівня безпеки і масштабованості. Натомість доцільно використовувати підхід, що базується на JSON Web Token (JWT). JWT є сучасним захисним механізмом, що містить інформацію про користувача та його права доступу, що дозволяє серверу ефективно перевіряти автентичність запитів без збереження стану сесії [52]. Даний підхід особливо добре підходить для реалізації мікросервісних архітектур завдяки тому, що кожен мікросервіс може незалежно перевіряти токени без необхідності додаткових запитів до центрального серверу аутентифікації.

3.3.3 Інструментальні засоби розробки для створення прикладного програмного забезпечення інформаційних управляючих систем

Для серверної частини програмного забезпечення ІУС було обрано сучасний JavaScript-фреймворк NestJS, побудований на платформі Node.js із використанням мови програмування TypeScript. Вибір саме цієї технології пов'язаний із високою продуктивністю і можливостями асинхронної роботи, що відповідає вимогам ІУС щодо швидкості обробки великих обсягів даних у режимі реального часу. TypeScript було обрано через наявність суворої типізації, що дозволяє мінімізувати кількість помилок, оптимізувати кодову базу та забезпечити зручність супроводу і подальшого масштабування програмного продукту.

Для збереження й організації структурованих конфігураційних та адміністративних даних використовується реляційна система керування базами

даних PostgreSQL. З метою забезпечення ефективної взаємодії із БД, контролю з'єднань, а також зменшення навантаження на базу даних, було використано ORM TypeORM. Цей інструмент дозволяє автоматично перетворювати записи таблиць БД в об'єкти предметної області, що значно спрощує розробку і підтримку бізнес-логіки системи [53]. Крім того, для забезпечення високої швидкості доступу до даних було реалізовано механізм асинхронного підходу на основі можливостей мови TypeScript та платформи Node.js, що дозволило значно зменшити затримки під час обробки численних одночасних запитів до бази даних.

З огляду на необхідність ефективного зберігання й обробки великих обсягів слабоструктурованих даних (сенсорні вимірювання, історичні логи, журнали подій), у системі паралельно застосовується нереляційна документо-орієнтована база даних MongoDB. Використання MongoDB дозволило реалізувати гнучке масштабування та адаптивність до зміни структури даних, що відповідатиме динамічним вимогам предметної області. Взаємодія серверної частини системи з MongoDB здійснюється через ORM-систему Mongoose для Node.js, що забезпечує системі високу швидкість читання та запису, а також дозволяє гнучко формувати запити і структуру документів [54].

Для створення клієнтської частини системи було обрано JavaScript-фреймворк Next.js на основі бібліотеки React, що дозволяє будувати сучасні реактивні веб-додатки з високою продуктивністю та незалежністю від платформи. React надає засоби для ефективної реалізації реактивного програмування, зокрема, механізми передачі та оновлення даних між компонентами через використання станів та властивостей. Next.js використовує принцип SSR, що гарантує швидке завантаження і рендеринг веб-сторінок, потенційною оптимізацію SEO та ефективну взаємодію з REST API серверної частини системи.

Для забезпечення передачі даних між основними компонентами клієнт-серверної архітектури у Next.js використовуються вбудовані механізми генерації HTML-шаблонів на основі JSX-технології, що дозволяє описувати структуру та логіку взаємодії компонентів максимально декларативно і зрозуміло. JSX у поєднанні з React дозволяє ефективно реалізовувати гнучкий і зрозумілий

користувацький інтерфейс із чітким розподілом відповідальності між компонентами [55].

Обрана розподілена архітектура інформаційної управляючої системи додатково враховує наявність окремого рівня периферійних пристроїв (мікроконтролерів ESP32), що мають власні логіку та функції в межах загальної архітектури. Такий підхід дає змогу винести частину обчислювального навантаження на периферійні пристрої та забезпечити високу автономність і відмовостійкість системи.

3.3.4 Тестування програмного забезпечення

Для досягнення зазначених цілей було здійснено поетапне тестування програмного забезпечення з використанням як ручних, так і автоматизованих методик. Комплексність тестування включала в себе три основні рівні: модульне, інтеграційне та системне тестування.

Модульне тестування серверної частини було виконано з використанням спеціалізованих бібліотек для мови програмування TypeScript, зокрема Jest, що забезпечує зручні механізми написання та виконання модульних тестів [56]. В межах NestJS для цих цілей було створено ізольовані середовища з тестовими екземплярами баз даних (PostgreSQL та MongoDB) та використовувався механізм створення тестових фікстур і моків. Модульні тести покрили ключові компоненти системи, такі як контролери, сервіси та репозиторії. Це дозволило забезпечити високу якість і стабільність окремих компонентів шляхом перевірки різних варіантів використання, та гарантувати, що зміни в одному модулі не впливатимуть на інші частини системи та сприяти швидкій ідентифікації й локалізації потенційних проблем у бізнес-логіці.

На рівні клієнтської частини (Next.js + React) також було реалізовано модульне й інтеграційне тестування. Тут було використано бібліотеку React Testing Library. Дана бібліотека дозволяє ефективно перевіряти коректність рендерингу та логіки роботи окремих компонентів інтерфейсу [57]. Інтеграційні тести

забезпечили перевірку взаємодії компонентів з API, включно із симуляцією різних сценаріїв роботи користувачів у системі.

Наступним етапом стало проведення системного тестування, що включало ручні тест-кейси і сценарії, які моделювали реальні умови використання ІУС. Для цього було налаштоване окреме тестове середовище, максимально наближене до умов експлуатації. Використано інструмент Postman, інтернет-перегляд Mozilla Firefox з консоллю розробника, а також перегляд з мобільного пристрою. Ручне тестування включало перевірку базових функціональних сценаріїв роботи кінцевих користувачів (моніторинг стану якості води, керування правилами, адміністрування сенсорних точок тощо). Приклад результатів тестування з покриттям відображений на рисунку 3.14.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	81.45	65.95	43.11	81.05	
src	100	100	100	100	
app.controller.ts	100	100	100	100	
app.service.ts	100	100	100	100	
src/alerts	62.96	100	9.09	56.52	
alerts.controller.ts	72.22	100	16.66	68.75	20,25,30,35,40
alerts.service.ts	44.44	100	0	28.57	8-24
src/alerts/dto	100	100	100	100	
create-alert.dto.ts	100	100	100	100	
update-alert.dto.ts	100	100	100	100	
src/commands	62.96	100	9.09	56.52	
commands.controller.ts	72.22	100	16.66	68.75	12,17,22,27,32
commands.service.ts	44.44	100	0	28.57	8-24
src/commands/dto	100	100	100	100	
create-command.dto.ts	100	100	100	100	
update-command.dto.ts	100	100	100	100	
src/configuration	100	100	100	100	
configuration.controller.ts	100	100	100	100	

Рисунок 3.14 – «Результати тестування проєкту з покриттям»

Джерело: розроблено автором

3.4 Технічне забезпечення

3.4.1 Мета та технічні вимоги до розроблювального пристрою

Метою розроблення технічного пристрою є реалізація автономної периферійної точки контролю, що забезпечує автоматичний моніторинг параметрів якості питної води, локальну обробку даних відповідно до гнучких правил реагування, що отримуються з сервера, а також взаємодію із серверною частиною ІУС. Застосування такого пристрою дозволяє оперативно виявляти порушення якості води безпосередньо у місці вимірювання та здійснювати автоматичні дії, спрямовані на усунення або пом'якшення негативних впливів.

Технічні вимоги до пристрою включають: наявність Wi-Fi-модуля для стабільного зв'язку з сервером через бездротову мережу з підтримкою стандарту 802.11 b/g/n та швидкістю передачі не менше 1 Мбіт/с; наявність процесора із частотою роботи не менше 160 МГц для забезпечення швидкого обчислення логіки правил; оперативну пам'ять не менше 320 КБ для розміщення логіки правил і буферизації даних; підтримку безпечного підключення з використанням SSL/TLS; точність обчислюваних значень з датчиків на рівні $\pm 1\%$ у робочому діапазоні; допустимий діапазон робочих температур від -10 до $+50$ °C, вологості до 90%, стійкість до помірних електромагнітних завад.

3.4.2 Обґрунтування вибору технічного забезпечення

Для вибору технічної основи розроблюваного пристрою проведено огляд наявних технічних рішень у предметній області. Розглянуто платформи Arduino Uno, Arduino Mega, Raspberry Pi, ESP8266-01 та ESP32. Arduino Uno та Mega є простими та надійними рішеннями, проте мають обмежені обчислювальні можливості та недостатню оперативну пам'ять для роботи з великими обсягами логіки й мережевих запитів. Raspberry Pi, попри високі обчислювальні можливості, має значно вищу собівартість і споживання електроенергії, що робить його менш оптимальним для масштабування і автономних сценаріїв роботи.

ESP8266-01 є прийнятним рішенням за ціною, однак має обмежену кількість входів-виходів і недостатню обчислювальну потужність для складних багаторівневих правил. Оптимальним рішенням було обрано мікроконтролер ESP32, який має баланс низької вартості, високих обчислювальних можливостей (двоядерний процесор Xtensa з частотою до 240 МГц), вбудовану пам'ять 520 КБ, достатню для зберігання й обробки JSON-структур з правилами, а також вбудований Wi-Fi-модуль із повною підтримкою протоколу HTTPS. Платформа ESP32 володіє низьким енергоспоживанням і широкими можливостями для інтеграції з різноманітними сенсорами, що робить її оптимальним рішенням для побудови периферійних точок контролю.

3.4.3 Опис розробленого технічного пристрою

Розроблений пристрій являє собою модуль автоматизованого моніторингу та керування на базі ESP32, оснащений портами для підключення сенсорів вимірювання температури, рН, вологості та інших параметрів якості води. Він містить модуль живлення, що підтримує напругу від 3.3 до 5 В, вбудований стабілізатор напруги, схему живлення та узгоджуючі елементи для підключення периферійних пристроїв.

Принцип роботи пристрою передбачає його автоматичну реєстрацію на сервері за допомогою унікального ключа, отримання від сервера конфігураційних даних та логіки правил у JSON-форматі, автономне виконання отриманих правил, періодичну передачу результатів моніторингу на сервер, а також регулярну перевірку оновлень. Структурна схема пристрою (див Рисунок 3.15) показує основні компоненти та зв'язки між ними. Загалом, ця структура є досить узагальненою, оскільки структура пристрою є залежною від наявних сенсорів та пристроїв.

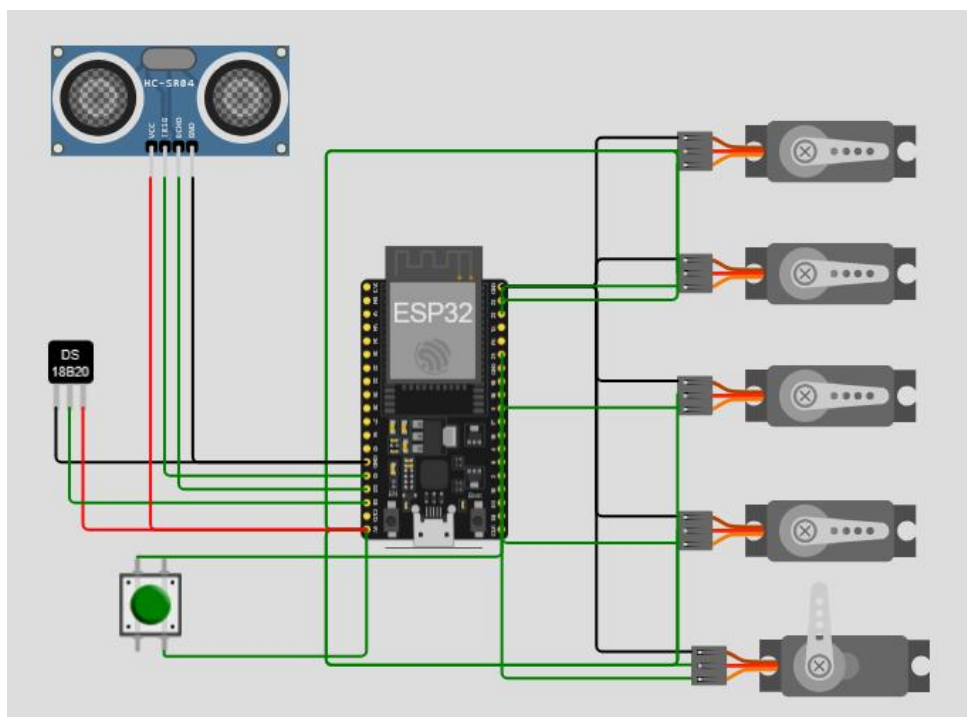


Рисунок 3.15 – «Структурна схема пристрою»

Джерело: розроблено автором

3.4.4 Методика виконання дослідно-експлуатаційного випробування технічного пристрою

Для виконання дослідно-експлуатаційних випробувань розробленого пристрою було використано середовище симуляції Wokwi у поєднанні з PlatformIO IDE, що дозволяє повністю емулювати роботу ESP32 без фізичного пристрою [58]. У межах випробувань перевірено функції реєстрації пристрою на сервері, отримання та виконання логіки правил, формування та надсилання журналів роботи на сервер, а також стабільність та продуктивність роботи пристрою при різних режимах навантаження. Тестування було виконано з імітацією різних значень сенсорних показників і перевірено коректність спрацювання логіки відповідно до встановлених правил. Але внаслідок обмеження платформи, тестування відбувалось окремо від серверної частини, застосовуючи емуляцію запитів та відповідей.

3.4.5 Настанова щодо використання технічного пристрою

Для експлуатації пристрою необхідно виконати попередню конфігурацію мережевого підключення та реєстрацію у серверній частині системи. Після цього пристрій автоматично отримує необхідні правила й конфігурацію, здійснює моніторинг сенсорів та виконує автономну логіку реагування. Рекомендується встановлювати пристрій у захищених від прямого потрапляння вологи місцях із стабільним покриттям Wi-Fi. Обслуговування пристрою включає періодичну перевірку стану мережевого з'єднання, працездатності підключених сенсорів та забезпечення стабільного живлення. За потреби конфігурацію параметрів можна виконувати дистанційно через серверну частину ІУС.

3.5 Реалізація інформаційної системи

Реалізація ІУС для моніторингу та контролю якості питної води здійснена на основі поєднання концептуальних положень автоматизації, сучасного архітектурного й програмного підходу та використання мікроконтролерних засобів у поєднанні з веб-орієнтованими технологіями. У результаті дослідження було розроблено, протестовано та задокументовано повноцінний програмно-апаратний прототип системи, який охоплює як серверну логіку управління та обробки даних, так і автономну периферійну точку контролю на базі ESP32, що реалізує локальне прийняття рішень.

Серверна частина системи реалізована за допомогою фреймворка NestJS, що забезпечило високий рівень модульності, підтримку структурованої бізнес-логіки та можливість легкої інтеграції з фронтенд-компонентом на базі Next.js. На клієнтському рівні розроблено веб-інтерфейс за допомогою фреймворку NextJS, який дозволяє здійснювати адміністрування контрольних точок, створення та редагування правил, перегляд журналів подій та поточних параметрів якості води в реальному часі. Комунікація між серверною та клієнтською частиною реалізована через RESTful API з використанням HTTP-протоколу та авторизації на основі JWT, що забезпечує належний рівень безпеки та ізоляції доступу.

Вона має наступну структуру(див. Рисунок 3.16):

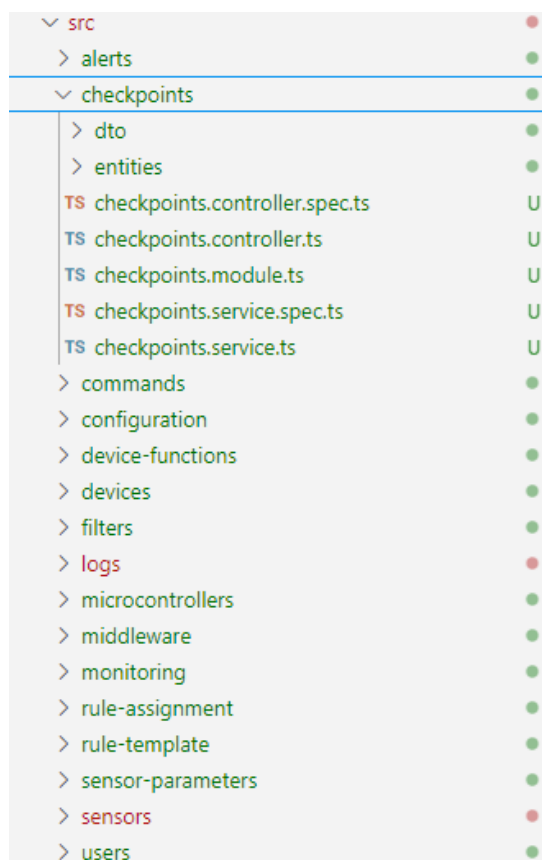


Рисунок 3.16 – «Структура серверного компоненту»

Джерело: розроблено автором

Принцип роботи: мікроконтролер подає запит на сервер з інформацією про себе (див. Рисунок 3.17)

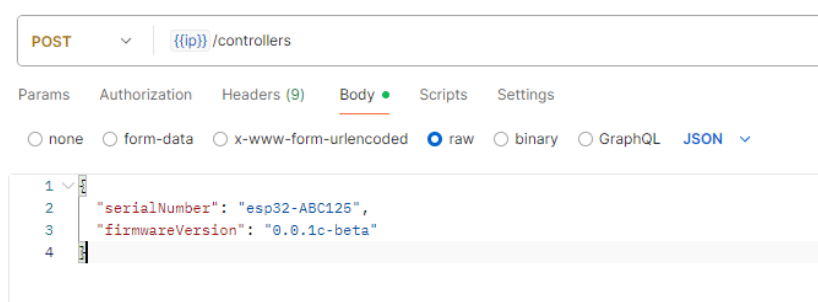


Рисунок 3.17 – «Серверний запит на реєстрацію мікроконтролера»

Джерело: розроблено автором

Сервер реєструє мікроконтролер та зберігає в базі даних, присвоюючи унікальний ідентифікатор. Крім того, мікроконтролер паралельно реєструє перелік власних сенсорів та пристроїв (див. Рисунок 3.18)

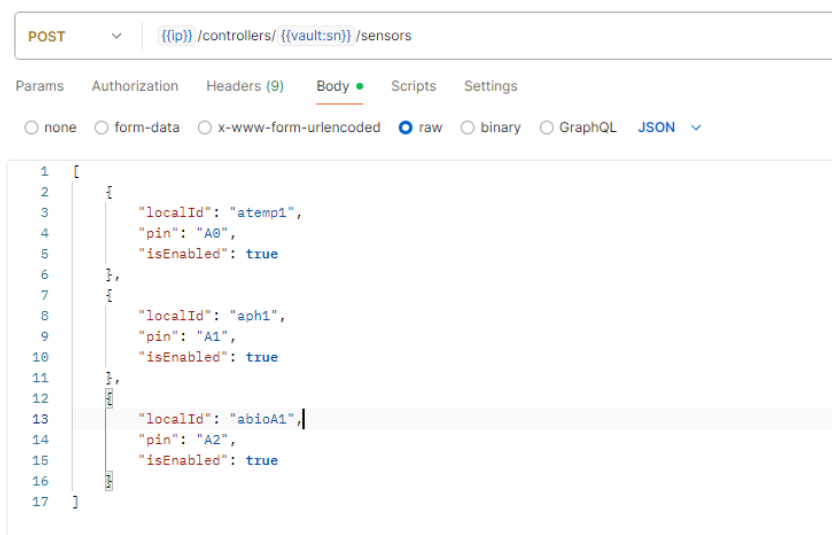


Рисунок 3.18 – «Серверний запит на реєстрацію мікроконтролера»

Джерело: розроблено автором

Далі, адміністратор через адміністративний інтерфейс підтверджує або відмовлює в реєстрації, також є можливість вказати додаткову ідентифікаційну інформацію (див. Рисунок 3.19).

СЕРІЙНИЙ НОМЕР	ВЕРСІЯ ПРОШИВКИ	КОНТРОЛЬНА ТОЧКА	IP-АДРЕСА	СТАН	СТВОРЕНО	ОНОВЛЕНО	СИНХРОНІЗОВАНО	ДІ
esp32-ABC123	0.0.1-beta	-		ACTIVE	2025-05-15T14:04:48.493Z	2025-05-16T07:53:35.760Z	-	Редигувати Сенсори Пристрої
esp32-ABC124	0.0.1a-beta	-		PENDING	2025-05-15T14:04:55.101Z	2025-05-16T09:07:16.322Z	-	Редигувати Сенсори Пристрої
esp32-ABC125	0.0.1c-beta	-		ACTIVE	2025-05-15T14:04:58.638Z	2025-05-16T09:09:49.129Z	-	Редигувати Сенсори Пристрої

Рисунок 3.19 – «Сторінка переліку мікроконтролерів»

Джерело: розроблено автором

Далі, адміністратор керує можливими точками контролю, конфігураціями, шаблонами правил та присвоєннями правил. Керує станами та процесом роботи (див. Рисунок 3.20)

ID	НАЗВА	ШИРОТА	ДОВГОТА	АКТИВНИЙ	СТВОРЕНО	ДІЇ
4	New checkpoint 2	50.4584	30.5245	true	2025-05-16T08:03:44.140Z	Редагувати
1	Test checkpoint 1	50.3245345	30.3453454	true	2025-05-15T13:50:08.219Z	Редагувати
3	Test checkpoint 2	50.12324	30.23234	true	2025-05-15T17:07:59.856Z	Редагувати
2	Test checkpoint 3 edited	50.12356	30.5231	false	2025-05-15T14:25:47.780Z	Редагувати

Рисунок 3.20 – «Сторінка переліку мікроконтролерів»

Джерело: розроблено автором

Крім того, користувач може переглянути інтерактивну карту з точками контролю та останніми показниками якості питної води (див. Рисунок 3.21)

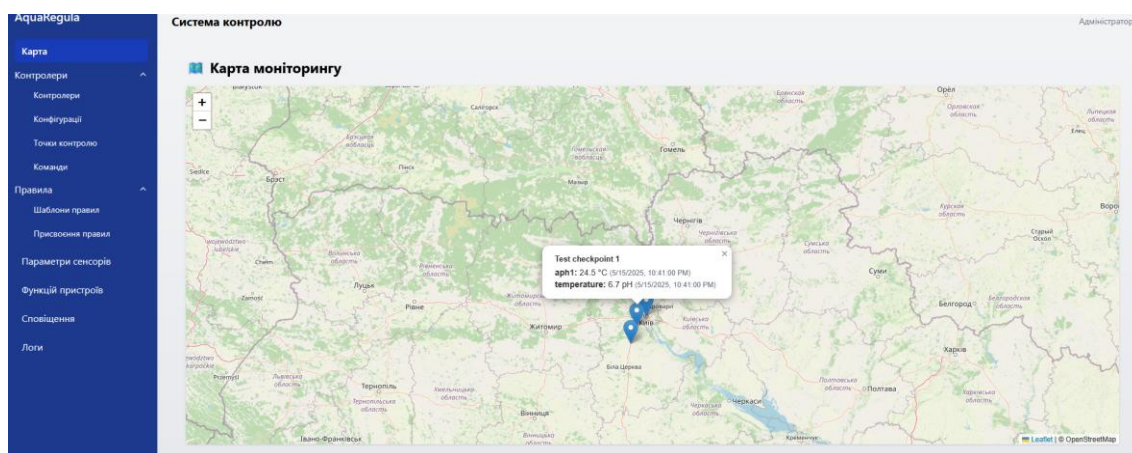


Рисунок 3.21 – «Сторінка інтерактивної карти»

Джерело: розроблено автором

Програмне забезпечення для ESP32-контролера створене з урахуванням необхідності забезпечення автономної роботи при втраті зв'язку з сервером. Контролер виконує логіку обробки правил локально, зберігає журнали подій у буфері та періодично передає їх на сервер. Це забезпечує підвищену надійність системи в умовах нестабільного зв'язку та дозволяє пристроям самостійно здійснювати вплив на якість води без затримок.

Новизна розробленого проєкту полягає у створення відкритої системи, що інтегрується з мікроконтролером, та дозволяє централізовано керувати його роботою, при цьому забезпечуючи максимальну автономність. Крім того, запропонована система гнучких правил у форматі JSON, що дозволяє створювати

сценарії адаптивного реагування з вкладеною логікою. Ці правила мають здатність бути перевикористаними завдяки механізму тегів, що дозволяє застосовувати єдиний шаблон правила для кількох мікроконтролерів без потреби в зміні самих правил. Система розроблена з урахуванням потреби в гнучкості та швидкодії. Використано гібридну систему зберігання даних (реляційна та документо-орієнтована база), що дозволяє ефективно обробляти як конфігураційні, так і потокові історичні дані.

Одержані результати мають прикладний характер і можуть бути використані для побудови систем екологічного моніторингу, автоматизованих станцій контролю якості води, а також як основа для розширених IoT-платформ у сфері смарт-інфраструктури. Структура системи підтримує масштабування, зручна в супроводі й дає змогу реалізовувати різні політики реагування без зміни коду, що відповідає критеріям практичної доцільності та оптимальності.

Висновки

В результаті проведеного дослідження було реалізовано ІУС для моніторингу на контролю якісних характеристик питної води систем централізованого водопостачання. Дана система дозволяє автоматизовано та автономно проводити мікроконтролерами збір показників серверів, їх обробку, аналіз, та відповідність встановленим нормам. Крім того, централізований сервер дає можливість керувати даним процесом дистанційно, при цьому забезпечуючи максимальну ефективність використання цих мікроконтролерів.

Наукова новизна дослідження полягає в комплексній реалізації рішень, які поєднують сенсорні вимірювання, автономну обробку даних на мікроконтролерах, централізовану логіку керування на сервері, візуалізацію параметрів у веб-інтерфейсі, а також підтримку гнучкого налаштування багаторівневих правил. Особливої значущості розробка набуває в умовах кліматичних викликів, урбанізації, зносу інфраструктури та загроз національного рівня (зокрема, війни).

До основних результатів роботи віднесено:

1. Досліджено сучасні підходи до контролю якості води в централізованих системах, нормативно-правову базу України та ЄС, виявлено слабкі сторони поточних рішень, зокрема відсутність технічної реалізації автоматизованих систем з можливістю автономного моніторингу у режимі реального часу.
2. Проаналізовано наявні SCADA-системи, IoT-платформи та спеціалізовані рішення. На основі порівняльної таблиці визначено оптимальний варіант – створення власної гнучкої системи на базі мікроконтролерів ESP32 із серверним ядром на NestJS, що забезпечує найкращий баланс між вартістю, надійністю, масштабованістю та автономністю.
3. Розроблено концептуальну архітектуру ІУС, що включає автономні точки контролю з сенсорами, підключеними до ESP32; серверний компонент для зберігання правил, конфігурацій та історії вимірювань; адміністративний веб-інтерфейс для налаштування та контролю; публічний інтерфейс для перегляду відповідної інформації користувачем.

4. Спроектовано базу даних із гібридною архітектурою, що поєднує PostgreSQL для конфігураційних і структурованих даних та MongoDB для зберігання історії сенсорних вимірювань і подій. Проведено прогнозування обсягів даних і обґрунтовано вибір СКБД з урахуванням продуктивності, масштабованості та енергоефективності.
5. Реалізовано базу знань для керування логікою системи – сформовано структуру правил у форматі вкладених JSON-документів з підтримкою тегів, що дозволяє багаторазове використання шаблонів незалежно від конкретного апаратного конфігуратора. Забезпечено підтримку багаторівневих умов з логікою AND/OR, часовими параметрами та зонами невизначеності.
6. Розроблено серверне і клієнтське програмне забезпечення – реалізовано NestJS-сервер із REST API, клієнт на базі Next.js для адміністраторів і користувачів, протестовано інтеграцію компонентів. Передбачено механізми автентифікації на основі JWT, передачу даних через HTTPS, використання шаблонів MVC та MVVM.
7. Створено прошивку для мікроконтролера ESP32, яка забезпечує першу реєстрацію на сервері, включно з сенсорними компонентами та пристроями впливу; локальне збереження конфігурації та правил; використання логіки прийняття рішень на основі сенсорних даних; автономну реакцію на відхилення без необхідності тримання постійного зв'язку з сервером; періодичну синхронізацію з сервером.

Таким чином можна зробити висновок, що розроблена ІУС може бути ефективно застосована для моніторингу якості питної води в умовах реального використання. Система дозволяє знизити ризики, пов'язані із затримкою реагування, забезпечує прозорість і доступ до інформації, підтримує адаптивні сценарії реагування, знижує витрати на обслуговування та є придатною до масштабування.

Практична значущість розробки полягає у можливості її впровадження у комунальних підприємствах, екологічних службах, санітарних установах і

громадах як засобу оперативного контролю, прогнозування та оптимізації процесів водопостачання. Система також може слугувати базовою платформою для розвитку «розумної» інфраструктури та екологічного моніторингу інших середовищ (повітря, ґрунту).

Подальші напрямки дослідження можуть включати розширення функціональності системи за рахунок застосування методів машинного навчання для прогнозування змін якості води, глибшу інтеграцію з мобільними застосунками, розширення системи типів сенсорів і алгоритмів реагування для створення ще складніших правил без втрати структури, розробку інструментарію для глибокого аналізу історичних даних.

Розроблена система відповідає поставленим вимогам до дослідження, а також сучасним технічним, екологічним та нормативним вимогам. Крім того, її використання показало ефективність застосованих інженерних і програмних рішень, тому система має високий потенціал до практичного використання в реальних умовах.

Список використаних джерел

1. ВПЛИВ ЯКОСТІ ВОДИ НА ЗДОРОВ'Я ЛЮДИНИ. Т. П. Гармаш, К. С. Кушніренко. *Збірка тез доповідей XVII Міжнародної науково-практичної конференції студентів, аспірантів і молодих учених «Екологія. Людина. Суспільство».* м. Київ, Україна, 2014. URL: <https://www.ecoconference.kiev.ua/wp-content/uploads/2023/10/Eco2014-final.pdf#page=46> (дата звернення: 25.11.2024).
2. ВСТАНОВЛЕННЯ ПРИЧИННО-НАСЛІДКОВИХ ЗВ'ЯЗКІВ МІЖ ЗАХВОРЮВАНІСТЮ НАСЕЛЕННЯ ТА ЯКІСТЮ ПИТНОЇ ВОДИ ДЖЕРЕЛ НЕЦЕНТРАЛІЗОВАНОГО ВОДОПОСТАЧАННЯ. Валерко, Р.А. та ін., Науково-практичний журнал «Екологічні науки» № 1(52), Том 2, 2024. URL: <https://doi.org/10.32846/2306-9716/2024.eco.1-52.2.4>
3. Проблеми забезпечення права населення на доступ до інформації про якість питної води. О. В. Гафурова, Т. С. Новак, С. М. Голуб. *Науковий вісник Ужгородського університету: серія: Право* / голов. ред. Ю. М. Бисага – Ужгород, 2023. –Т. 1. Вип. 79. – С. 356-361. – Бібліогр.: с. 360-361 (18 назв). URL: <http://visnyk-pravo.uzhnu.edu.ua/article/view/288634/282294>
4. АНАЛІЗ ПОКАЗНИКІВ ЯКОСТІ ВОДИ: СУЧАСНІ АСПЕКТИ І ВИКЛИКИ. Шестопапов О.В та ін. *Науково-практичний журнал «Екологічні науки»* № 3(54), с. 76–82, 2024. URL: <https://doi.org/10.32846/2306-9716/2024.eco.3-54.10>
5. Сучасний стан водних ресурсів басейну Дніпра у порівнянні з річками інших країн. Кануннікова Н.О. та ін. *Український журнал природничих наук* № 9, , с. 314–322, 2024. URL: <https://doi.org/10.32782/naturaljournal.9.2024.32>
6. Вдосконалення роботи та методика дослідження водопостачання. Епоян С.М. та ін. *Комунальне господарство міст.*, №4(185), с. 133–140. URL: <https://doi.org/10.33042/2522-1809-2024-4-185-133-140>
7. Про затвердження Державних санітарних норм та правил "Гігієнічні вимоги до води питної, призначеної для споживання людиною" (ДСанПіН 2.2.4-

- 171-10). *Офіційний вебпортал парламенту України*. URL: <https://zakon.rada.gov.ua/laws/show/z0452-10> (дата звернення: 13.11.2024).
8. Як здійснюється контроль якості та безпечності питної води. Що таке виробничий контроль якості води питної?. Головна | *Головне управління Держпродспоживслужби в Тернопільській області*. URL: <https://dpss-te.gov.ua/news/yak-zdiisniuietsia-kontrol-iakosti-ta-bezpechnosti-pytnoi-vody-shcho-take-vyrobnychy-kontrol-iakosti-vody-pytnoi> (дата звернення: 02.02.2025).
 9. Детальніше про механічне очищення стічних вод. *Акваполімер інжиніринг | у гармонії з довкіллям*. URL: <https://aquapolymer.com.ua/blog/detalnyshe-pro-mehanichne-ochyshhennya-stichnyh-vod/> (дата звернення: 02.02.2025).
 10. Фізико-хімічні методи очищення стічних вод. *ЗИКО*. URL: <https://ziko.com.ua/fizyko-khimichni-metody-ochyshchennia-stichnykh-vod/> (дата звернення: 08.02.2025).
 11. Особливості біологічного очищення стічних вод. *ЗИКО*. URL: <https://ziko.com.ua/all-article-biologichne-ochyshchennya-stichnykh-vod/> (дата звернення: 08.02.2025).
 12. Структура системи водопостачання, призначення її елементів. *Навчальні матеріали ТДАТУ*. URL: https://elib.tsatu.edu.ua/dep/mtf/tsapk_2/page5.html (дата звернення: 14.02.2025).
 13. Про орган: Держпродспоживслужба. *Ліга 360*. URL: <https://ips.ligazakon.net/document/TO000612> (дата звернення: 14.02.2025).
 - 14.3 1 липня в Україні запрацюють Центри контролю та профілактики хвороб | Центр громадського здоров'я. *Центр громадського здоров'я України | МОЗ*. URL: <https://phc.org.ua/news/z-1-lipnya-v-ukraini-zapracuyuyut-centri-kontrolyu-ta-profilaktiki-khvorob> (дата звернення: 26.02.2025).
 15. Директива Ради 98/83/ЄС від 3 листопада 1998 року про якість води, призначеної для споживання людиною : Директива Європ. Союзу від 03.11.1998 № 98/83/ЄС. URL:

https://zakon.rada.gov.ua/laws/show/994_963#Text (дата звернення: 27.02.2025).

16.ПРАВОВЕ РЕГУЛЮВАННЯ ЯКОСТІ І БЕЗПЕКИ ПИТНОЇ ВОДИ В ЄС

ТА УКРАЇНИ. Вітів В. А. . *Young Scientist*. 2018. № 12 (64) URL: <https://doi.org/10.32839/2304-5809/2018-12-64-45> (дата звернення: 11.03.2024).

17.Огляд року війни для водних ресурсів України – Екологія Право Людина.

Екологія Право Людина. URL: <https://epl.org.ua/announces/oglyad-roku-vijny-dlya-vodnyh-resursiv-ukrayiny/> (дата звернення: 14.03.2025).

18.Національна доповідь про якість питної води у 2023 році. Міністерство розвитку громад та територій України. URL:

<https://mtu.gov.ua/files/%D0%9D%D0%B0%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%20%D0%B4%D0%BE%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D1%8C%20%D0%BF%D1%80%D0%BE%20%D1%8F%D0%BA%D1%96%D1%81%D1%82%D1%8C%20%D0%BF%D0%B8%D1%82%D0%BD%D0%BE%D1%97%20%D0%B2%D0%BE%D0%B4%D0%B8%20%D1%83%202023%20%D1%80%D0%BE%D1%86%D1%96.pdf> (дата звернення: 26.11.2024).

19.Заступниця МОЗ Марина Слободніченко: про євроінтеграцію, медичний канабіс, трансплантацію та ліки від старіння, про які мріє Путін.

«Українські новини», 02 листопада 2024. URL: <https://ukranews.com/ua/interview/3151-zastupnytsya-moz-maryna-slobodnichenko-yevropejtsi-kazhut-pokazhit-spozhyvachiv-bo-v-nyh-tam-lezhat> (дата звернення: 28.11.2024).

20.7 способів перевірити якість води у домашніх умовах. AquaPro. URL:

<https://aquapro.ua/blog/7-sposobov-proverit-kachestvo-vody-v-domashnikh-usloviyakh> (дата звернення: 14.03.2025).

21.Визначення показників води в домашніх умовах. ECOSOFT. URL:

<https://ecosoft.ua/ua/blog/opredelenie-pokazateley-vody-v-domashnikh-usloviyakh/> (дата звернення: 14.03.2025).

22. НАУКОВІ ПІДХОДИ В МОДЕРНІЗАЦІЇ СИСТЕМ СІЛЬСЬКОГОСПОДАРСЬКОГО ВОДОПОСТАЧАННЯ І ВОДОВІДВЕДЕННЯ В УМОВАХ ЗМІН КЛІМАТУ. Хоружий, П. Д., Стасюк, С. Р., Харланов, Д. І., Левицька, В. Д., Мосійчук, Я. Б., Нор В. В. *Управління водними ресурсами в умовах змін клімату*, 2017
23. Обсяги скидів забруднених (забруднених без очистки та недостатньо очищених) стічних вод у водні об'єкти. Портал відкритих даних. URL: <https://data.gov.ua/dataset/a4a57fc5-51d3-4a72-bbdf-9d3a2df9b2b4> (дата звернення: 30.11.2024).
24. Аналіз сучасних технологій доочищення питної води. Бартківська В. В. *Экологическая безопасность государства: тезисы докладов XIII*. 2020.
25. Аналіз показників якості води: сучасні аспекти і виклики(2024). Шестопапов, О. В., Сакун, А. О., Лізанта, П. С., Кануннікова, Н. О., Гайдучек, О. Г., Томашевський, Р. С., & Воробйов, Б. В.. *Екологічні науки*, 3(54), 76-82.
26. What is SCADA? Supervisory Control and Data Acquisition. Inductive Automation. URL: <https://inductiveautomation.com/resources/article/what-is-scada> (дата звернення: 21.03.2025).
27. Getting Start with FUXA - Web Based SCADA Tool | Seeed Studio Wiki. Seeed Studio Wiki. URL: https://wiki.seeedstudio.com/reTerminal-DM_intro_FUXA/ (дата звернення: 22.03.2025).
28. ЦИФРОВИЙ МОНІТОРИНГ ЯКОСТІ ВОДИ, ВИКЛИКИ ТА РІШЕННЯ. Корніюк А.В., Пацева І.Г. *ЕКОЛОГІЧНИЙ МОНІТОРИНГ*. URL: <https://doi.org/10.32846/2306-9716/2023.eco.4-49.4>
29. Aquacomputer. URL: <https://aquacomputer.de/software.html> (дата звернення: 26.03.2025).
30. Ayyeka : solutions for real-time decisions and field data. URL: <https://www.ayyeka.com/> (дата звернення: 26.03.2025).
31. AquaWatch | Water monitoring tool| Insights and management. URL: <https://www.aquawatchsolutions.com/> (дата звернення: 26.03.2025).

- 32.Synox, IoT solutions editor for companies and smart cities. URL: <https://www.synox.io/en/> (дата звернення: 27.03.2025).
- 33.Disrupt-X | Leading global IoT solutions platform. URL: <https://disrupt-x.io/> (дата звернення: 27.03.2025).
- 34.Specialist air quality monitoring company | Envira. URL: <https://envira.global/> (дата звернення: 28.03.2025).
- 35.Energy Monitoring System | IoT based. URL: <https://www.energyly.com/> (дата звернення: 01.04.2025).
- 36.Smart water systems – 24\7 Water monitoring & Flood detection. URL: <https://connectedsensors.com/> (дата звернення: 01.04.2025).
- 37.Zhang, Qingqing, and Xue-yi You. "Recent advances in surface water quality prediction using artificial intelligence models." *Water Resources Management* 38, no. 1 (2024): 235-250.
- 38.A cloud model-based approach for water quality assessment. Wang, D., Liu, D., Ding, H., Singh, V. P., Wang, Y., Zeng, X., ... & Wang, L. (2016). *Environmental research*, 148, 24-35.
- 39.What is Arduino in IoT?. Skyfi Labs. URL: <https://www.skyfilabs.com/resources/what-is-arduino-in-iot>(дата звернення: 12.04.2024).
- 40.Design and Implementation of ESP32-Based IoT Devices. Hercog, D., Lerher, T., Truntič, M., & Težak, O. (2023). *Sensors*, 23(15), 6739. URL: <https://doi.org/10.3390/s23156739>
- 41.IoT Innovations in Sustainable Water and Wastewater Management and Water Quality Monitoring: A Comprehensive Review of Advancements, Implications, and Future Directions. A. Alshami та ін. *IEEE Access*. 2024. С. 1. URL: <https://doi.org/10.1109/access.2024.3392573> (дата звернення: 02.12.2024).
- 42.Complete ESP32 Programming Guide: From Basics to Advanced (ESP32-C3, S3, WROOM, CAM, OTA & Arduino IDE). *Rayming*. URL: <https://www.raypcb.com/esp32-programming-circuit/> (дата звернення: 05.04.2025).

43. A Guide to ACID Properties in Database Management Systems. *MongoDB*. URL: <https://www.mongodb.com/resources/basics/databases/acid-transactions> (дата звернення: 15.04.2025).
44. Understanding SQL vs NoSQL Databases. *MongoDB*. URL: <https://www.mongodb.com/resources/basics/databases/nosql-explained/nosql-vs-sql> (дата звернення: 16.04.2025).
45. How are Big Data and the Internet of Things connected?. *Nexus group*. URL: <https://www.nexusgroup.com/how-are-big-data-and-the-internet-of-things-connected/> (дата звернення: 18.04.2025).
46. Keeping Databases Lean with TTL: Practical Insights and Strategies. Leela Kumili, 2024. *Medium*. URL: <https://medium.com/@leela.kumili/keeping-databases-lean-with-ttl-practical-insights-and-strategies-ec6d9d09f91a> (дата звернення: 26.04.2025).
47. Understanding PostgreSQL architecture and attributes. Prisma. URL: <https://www.prisma.io/dataguide/postgresql/getting-to-know-postgresql> (дата звернення: 26.04.2025).
48. MongoDB: What it is, How it works, and What it's used for. Cooper Hong. *Knowi*. URL: <https://www.knowi.com/blog/mongodb-what-is-mongodb/> (дата звернення: 28.04.2025).
49. NestJS: The Perfect Javascript Backend Framework for Structure, Clean Code, and Modularity. AJ Software engineering, 2024. *Medium*. URL: <https://medium.com/@ajonesb/nestjs-the-perfect-javascript-backend-framework-for-structure-clean-code-and-modularity-ae1b3a6e1418> (дата звернення: 01.05.2025).
50. Mastering Next.js: A Comprehensive Guide. Rushabh Dabhade, 2024. *Medium*. URL: https://medium.com/@Rushabh_/mastering-next-js-a-comprehensive-guide-7799c3b8f968 (дата звернення: 05.05.2025).
51. Your Gateway to Embedded Software Development Excellence. *PlatformIO*. URL: <https://docs.platformio.org/en/latest/> (дата звернення: 08.05.2025).

52. What is SSL, TLS & HTTPS? *Digicert*. URL: <https://www.digicert.com/what-is-ssl-tls-and-https> (дата звернення: 10.05.2024).
53. Як використовувати JSON Web Tokens (JWT) для автентифікації. 2019. *DevZone*. URL: <https://devzone.org.ua/post/iak-vykorystovuvaty-json-web-tokens-jwt-dlia-avtentyfikatsiyi> (дата звернення: 10.05.2024).
54. TypeORM. URL: <https://typeorm.io/> (дата звернення: 12.04.2024).
55. Mongoose ODM. URL: <https://mongoosejs.com/> (дата звернення: 12.04.2024).
56. Writing Markup with JSX. *React*. URL: <https://react.dev/learn/writing-markup-with-jsx> (дата звернення: 05.05.2024).
57. Getting Started – Jest. *Jest*. URL: <https://jestjs.io/docs/getting-started> (дата звернення: 08.05.2025).
58. React Testing Library. *React Testing Library*. URL: <https://testing-library.com/docs/react-testing-library/intro/> (дата звернення: 09.05.2025).
59. Wowki – World’s most advanced ESP32 Simulator. URL: <https://wokwi.com/> (дата звернення: 12.05.2025).
60. Вимоги до якості питної води. *Ecosoft*. URL: <https://ecosoft.ua/ua/blog/trebovaniya-k-kachestvu-pitevoy-vody/> (дата звернення: 03.02.2024).
61. Model–view–viewmodel. *Wikipedia*. URL: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel> (дата звернення: 15.11.2024).
62. Headless Architecture with Separated UI for Backend and Frontend. Mehmet Ozkaya, 2023. *Medium*. URL: <https://medium.com/design-microservices-architecture-with-patterns/headless-architecture-with-separated-ui-for-backend-and-frontend-f9789920e112> (дата звернення: 03.03.2025).
63. Design patterns for microservices, 2017. *Microsoft Azure*. URL: <https://azure.microsoft.com/en-us/blog/design-patterns-for-microservices/> (дата звернення: 15.03.2025).

Додатки

Додаток А. Приклади програмного коду

```

9 |   Logger,
10 | } from '@nestjs/common';
11 | import { MicrocontrollersService } from './microcontrollers.service';
12 | import { RegisterMicrocontrollerDto } from './dto/register-controller.dto';
13 | import { AdminUpdateMicrocontrollerDto } from './dto/admin-update-controller.dto';
14 | import { UpdateStatusDto } from './dto/update-status.dto';
15 | import { ControllerStatus } from './entities/microcontrollers.entity';
16 |
17 | @Controller('controllers')
18 | export class MicrocontrollersController {
19 |   private readonly logger = new Logger(MicrocontrollersController.name);
20 |
21 |   constructor(private readonly service: MicrocontrollersService) {}
22 |
23 |   @Get('/statuses')
24 |   getStatuses() {
25 |     this.logger.log('Sending statuses');
26 |     return ControllerStatus;
27 |   }
28 |
29 |   @Post('')
30 |   register(@Body() dto: RegisterMicrocontrollerDto) {
31 |     this.logger.log(`Incoming registration from SN=${dto.serialNumber}`);
32 |     return this.service.registerOrUpdate(dto);
33 |   }
34 |
35 |   @Patch(':serial/status')
36 |   updateStatus(@Param('serial') sn: string, @Body() dto: UpdateStatusDto) {
37 |     this.logger.log(`Updated status for SN=${sn} to ${dto.status}`);
38 |   }
39 |
40 |   @Get()
41 |   findAll() {
42 |     this.logger.debug('Client requested list of all microcontrollers');
43 |     return this.service.findAll();

```

Рисунок А.1 – «Контролер мікроконтролера»

Джерело: розроблено автором

```

18  @Injectable()
19  export class MicrocontrollersService {
20    private readonly logger = new Logger(MicrocontrollersService.name);
21
22    constructor(
23      @InjectRepository(Microcontroller)
24      private controllerRepo: Repository<Microcontroller>,
25      private configsService: ConfigurationService,
26      private checkpointsService: CheckpointsService,
27    ) {}
28
29    async registerOrUpdate(dto: RegisterMicrocontrollerDto) {
30      const controller = await this.controllerRepo.findOneBy({
31        serialNumber: dto.serialNumber,
32      });
33
34      if (controller) {
35        this.logger.log(
36          `Controller ${dto.serialNumber} already exists. Updating info...`,
37        );
38        controller.firmwareVersion =
39          dto.firmwareVersion ?? controller.firmwareVersion;
40        controller.ip = dto.ip ?? controller.ip;
41        controller.status = ControllerStatus.PENDING;
42        return this.controllerRepo.save(controller);
43      }
44
45      this.logger.log(`Registering new controller SN=${dto.serialNumber}`);
46      const created = this.controllerRepo.create({
47        serialNumber: dto.serialNumber,
48        firmwareVersion: dto.firmwareVersion ?? '',
49        ip: dto.ip ?? '',
50      });
51      return this.controllerRepo.save(created);
52    }
53
54    async findAll() {
55      const result = await this.controllerRepo.find();
56      this.logger.debug(`Fetched ${result.length} microcontrollers`);
57      return result;

```

Рисунок А.2 – «Сервіс мікроконтролера»

Джерело: розроблено автором

```

16  export enum ControllerStatus {
17      PENDING = 'PENDING',
18      ACTIVE = 'ACTIVE',
19      DISABLED = 'DISABLED',
20      ERROR = 'ERROR',
21  }
22
23  @Entity()
24  export class Microcontroller {
25      @PrimaryGeneratedColumn()
26      id: number;
27
28      @Column({ unique: true })
29      serialNumber: string;
30
31      @Column()
32      firmwareVersion: string;
33
34      @Column({ nullable: true })
35      ip: string;
36
37      @ManyToOne(() => Checkpoint, { nullable: true })
38      checkpoint: Checkpoint;
39
40      @OneToMany(() => RuleAssignment, (ra) => ra.controller, { nullable: true })
41      assignedRules: RuleAssignment[];
42
43      @ManyToOne(() => Configuration, { nullable: true })
44      configuration: Configuration | null;
45
46      @Column({
47          type: 'enum',
48          enum: ControllerStatus,
49          default: ControllerStatus.PENDING,
50      })
51      status: ControllerStatus;

```

Рисунок А.3 – «Сутність мікроконтролера»

Джерело: розроблено автором

```

1  import { EntityTable } from "@components/entity/EntityTable";
2  import { controllersApi } from "@api/controllers";
3  import Link from "next/link";
4
5  export default async function ControllersPage() {
6    const controllers = await controllersApi.getAll();
7
8    return (
9      <main className="p-8">
10     <div className="flex justify-between items-center mb-4">
11       <h1 className="text-2xl font-bold">Мікроконтролери</h1>
12     </div>
13     <EntityTable
14       data={controllers}
15       columns={[
16         { key: "serialNumber", label: "Серійний номер" },
17         { key: "firmwareVersion", label: "Версія прошивки" },
18         { key: "checkpoint", label: "Контрольна точка" },
19         { key: "ip", label: "IP-адреса" },
20         { key: "status", label: "Стан" },
21         { key: "createdAt", label: "Створено" },
22         { key: "updatedAt", label: "Оновлено" },
23         { key: "lastSyncedAt", label: "Синхронізовано" },
24       ]}
25       actions={({item}) => (
26         <><Link
27           href={` /controllers/${item.serialNumber}/edit`}
28           className="text-blue-600 hover:underline"
29         >
30           Редагувати
31         </Link> <Link
32           href={` /controllers/${item.serialNumber}/sensors`}
33           className="text-blue-600 hover:underline"
34         >
35           Сенсори
36         </Link> <Link
37           href={` /controllers/${item.serialNumber}/devices`}
38           className="text-blue-600 hover:underline"
39         >
40           Пристрої
41         </Link></>

```

Рисунок А.4 – «Клієнтська сторінка переліку контролерів»

Джерело: розроблено автором

Звіт подібності

метадані

Назва організації

Kyiv National Economic University named after Vadym Hetman KNEU

Заголовок

Розроблення інформаційної системи для контролю якості питної води централізованого водопостачання

Автор

Науковий керівник / Експерт

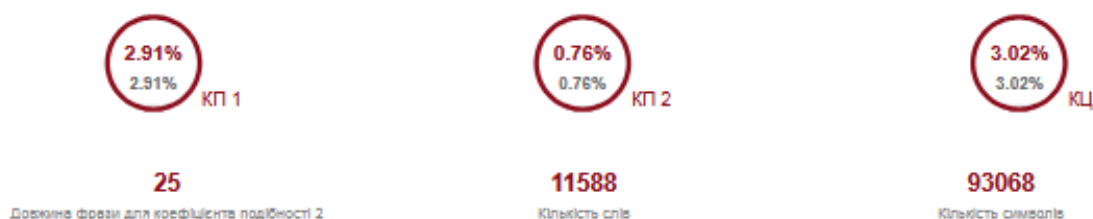
Кучер Владислав Васильович Устенко С.В.

Підрозділ

кафедра інформаційних систем в економіці

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		0
Інтервали		0
Мікропробіли		1
Білі знаки		0
Парафрази (SmartMarks)		9

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз

Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КОЛЬКІСТЬ ЩЕПІТНИХ СЛІВ (ПЧА МЕНТІВ)
1	https://ecolog-ua.com/consultation/za-yakymy-pokaznykamy-neobhidno-zabezpechyty-pislyaproektnyy-monitoring-vody-iz	61 0.53 %
2	https://ua-referat.com/uploaded/kvalifikacijna-bakalavrseka-robota/index5.html	27 0.23 %
3	Інформаційна управляюча система інтелектуального підбору персоналу на підприємстві 6/16/2020 Kyiv National Economic University named after Vadym Hetman KNEU (кафедра інформаційних систем в економіці)	20 0.17 %

4	https://mtu.gov.ua/files/%D0%9D%D0%B0%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%B8%D1%8C%D0%BD%D0%B0%20%D0%B4%D0%BE%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D1%8C%20%D0%BF%D1%80%D0%BE%20%D1%8F%D0%BA%D1%96%D1%81%D1%82%D1%8C%20%D0%BF%D0%B8%D1%82%D0%BD%D0%BE%D1%97%20%D0%B2%D0%BE%D0%B4%D0%B8%20%D1%82%D0%B0%20%D1%81%D1%82%D0%B0%D0%BD%20%D0%BF%D0%B8%D1%82%D0%B0%D0%BE%D0%B3%D0%BE%20%D0%B2%D0%BE%D0%B4%D0%BE%D0%BF%D0%BE%D1%81%D1%82%D0%B0%D1%87%D0%B0%D0%BD%D0%BD%D1%8F%20%D1%82%D0%B0%20%D0%B2%D0%BE%D0%B4%D0%BE%D0%B2%D1%96%D0%B4%D0%B2%D0%B5%D0%B4%D0%BD%D1%8F%20%D0%B2%20%D0%A3%D0%BA%D1%80%D0%B0%D1%97%D0%BD%D1%96%20%D1%83%202023%20%D1%80.pdf	14 0.12 %	
5	МЕТОДИКА АНАЛІЗУ РЕЖИМІВ РОБОТИ КОНВЕЄРНОЇ ЛІНІЇ ЗАСОБАМИ SCADA-СИСТЕМИ WONDERWARE.docx 12/8/2019 Kryvyi Rih National University (Кафедра комп'ютерних систем та мереж)	14 0.12 %	
6	https://dayr.gov.ua/news/malezchna-kilkist-fa-yakist-pitno-vodi-yak-skladova-ekologichni-stijkosli-ukraini-perebuyaye-pid-postijnim-monitoringom-derzhvodagentstva	13 0.11 %	
7	https://mtu.gov.ua/files/%D0%9D%D0%B0%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%B8%D1%8C%D0%BD%D0%B0%20%D0%B4%D0%BE%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D1%8C%20%D0%BF%D1%80%D0%BE%20%D1%8F%D0%BA%D1%96%D1%81%D1%82%D1%8C%20%D0%BF%D0%B8%D1%82%D0%BD%D0%BE%D1%97%20%D0%B2%D0%BE%D0%B4%D0%B8%20%D1%82%D0%B0%20%D1%81%D1%82%D0%B0%D0%BD%20%D0%BF%D0%B8%D1%82%D0%B0%D0%BE%D0%B3%D0%BE%20%D0%B2%D0%BE%D0%B4%D0%BE%D0%BF%D0%BE%D1%81%D1%82%D0%B0%D1%87%D0%B0%D0%BD%D0%BD%D1%8F%20%D1%82%D0%B0%20%D0%B2%D0%BE%D0%B4%D0%BE%D0%B2%D1%96%D0%B4%D0%B2%D0%B5%D0%B4%D0%BD%D1%8F%20%D0%B2%20%D0%A3%D0%BA%D1%80%D0%B0%D1%97%D0%BD%D1%96%20%D1%83%202023%20%D1%80.pdf	12 0.10 %	
8	Інформаційна управляюча система Інтелектуального підбору персоналу на підприємстві 6/16/2020 Kyiv National Economic University named after Vadym Hetman KNEU (кафедра Інформаційних систем в економіці)	12 0.10 %	
9	Веб платформа корпоративної соціальної мережі 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	12 0.10 %	
10	Інформаційна управляюча система Інтелектуального підбору персоналу на підприємстві 6/16/2020 Kyiv National Economic University named after Vadym Hetman KNEU (кафедра Інформаційних систем в економіці)	11 0.09 %	
з бази даних RefBooks (0.00 %)			
позначення нових*	заг. список	кількість цитованих слів (total mentions)	
з домашньої бази даних (1.10 %)			
позначення нових*	заг. список	кількість цитованих слів (total mentions)	
1	Інформаційна управляюча система Інтелектуального підбору персоналу на підприємстві 6/16/2020 Kyiv National Economic University named after Vadym Hetman KNEU (кафедра Інформаційних систем в економіці)	109 (11) 0.94 %	
2	Розроблення інформаційної управляючої системи для страхової компанії 5/13/2022 Kyiv National Economic University named after Vadym Hetman KNEU (кафедра Інформаційних систем в економіці)	10 (1) 0.09 %	



IN 43028
від 06.10.2023

СЕРТИФІКАТ

учасника конференції



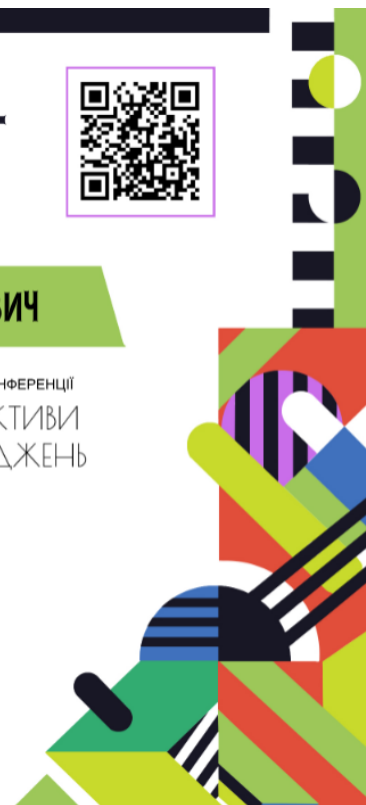
КУЧЕР ВЛАДИСЛАВ ВАСИЛЬОВИЧ

ВЗЯВ(-ЛА) УЧАСТЬ У V МІЖНАРОДНІЙ СТУДЕНТСЬКІЙ НАУКОВІЙ КОНФЕРЕНЦІЇ
АКТУАЛЬНІ ПИТАННЯ ТА ПЕРСПЕКТИВИ
ПРОВЕДЕННЯ НАУКОВИХ ДОСЛІДЖЕНЬ
6 ЖОВТНЯ 2023 РІК • М. ТЕРНОПІЛЬ, УКРАЇНА

В рамках участі було опубліковано тези доповіді учасника на тему:

**ВИКОРИСТАННЯ ХМАРНИХ ТЕХНОЛОГІЙ ДЛЯ
ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОЗРОБКИ
ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

ДИРЕКТОР МОЛОДІЖНОЇ НАУКОВОЇ ЛІГИ
ГОЛОВА ОРГКОМІТЕТУ КОНФЕРЕНЦІЇ
ІГОР КОРЕНЮК


IN 94055
від 25.04.2025

СЕРТИФІКАТ

учасника конференції



КУЧЕР ВЛАДИСЛАВ ВАСИЛЬОВИЧ

ВЗЯВ(-ЛА) УЧАСТЬ У IX МІЖНАРОДНІЙ СТУДЕНТСЬКІЙ НАУКОВІЙ КОНФЕРЕНЦІЇ
СУЧАСНІ АСПЕКТИ ТА ПЕРСПЕКТИВНІ
НАПРЯМКИ РОЗВИТКУ НАУКИ
25 КВІТНЯ 2025 РІК • М. ТЕРНОПІЛЬ, УКРАЇНА

В рамках участі було опубліковано тези доповіді учасника на тему:

**ЦИФРОВІЗАЦІЯ ТА КІБЕРБЕЗПЕКА - ВИКЛИКИ І
ПЕРСПЕКТИВИ ДЛЯ УКРАЇНИ**

ДИРЕКТОР МОЛОДІЖНОЇ НАУКОВОЇ ЛІГИ
ГОЛОВА ОРГКОМІТЕТУ КОНФЕРЕНЦІЇ
ІГОР КОРЕНЮК