НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ **Ігор БОЛБОТ**	_____ **Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення моніторингу клімату в закритому приміщені»

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми
к.т.н., доцент _____ **Ганна ВАЙГАНГ**
(підпис)

Керівник бакалаврської кваліфікаційної роботи _____ **Володимир ХИЛЕНКО**
(підпис)

Виконав _____ **Михайло КОЛЕСНІКОВ**
(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ

«___» _____ 2026 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ

Колеснікову Михайлу Андрійовичу

Спеціальність «121 Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»
Тема бакалаврської кваліфікаційної роботи
«Програмне забезпечення моніторингу клімату в закритому приміщені»
затверджена наказом від «02» квітня 2026 р. № 938 «С»
Термін подання завершеної роботи на кафедру «22» травня 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): Правила призначення академічних стипендій у Національному університеті біоресурсів і природокористування України від 22.11.2023, Форма додатка до диплома європейського зразка.
Перелік питань, що підлягають дослідженню: моніторингу клімату в закритому приміщені як об'єкт розробки, проєктування інформаційного забезпечення, розробка програмного забезпечення, рекомендації щодо впровадження та експлуатації системи.
Дата видачі завдання «3» квітня 2026 р.

Керівник бакалаврської кваліфікаційної роботи	_____ (підпис)	**Володимир ХИЛЕНКО**
Завдання взяв до виконання	_____ (підпис)	**Михайло КОЛЕСНІКОВ**

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API — Інтерфейс програмування застосунків (Application Programming
Interface)

БД — база даних

ІС — інформаційна система

IoT — Інтернет речей (Internet of Things), мережа фізичних пристроїв, що можуть обмінюватися даними

MQTT — легкий протокол обміну повідомленнями для пристроїв IoT

NodeMCU — мікроконтролер для збору та передачі даних сенсорів

Сенсор — пристрій для вимірювання параметрів мікроклімату (температура, вологість, CO_2 тощо)

ПЗ — програмне забезпечення

СУБД — система управління базами даних

Dashboard— інформаційна панель для візуалізації даних

Node-RED – платформа для оркестрації IoT-потоків без кодування

JSON — JavaScript Object Notation, формат передачі структурованих даних UI — User Interface, інтерфейс користувача

ВСТУП

У сучасному світі комфорт, енергоефективність та безпека в приміщеннях значною мірою залежать від оптимальних параметрів мікроклімату. Контроль температури, вологості, рівня CO_2 та інших показників є важливим завданням для власників житлових і громадських будівель, шкіл, офісів, лабораторій тощо. Традиційні методи контролю, як-от періодичні ручні вимірювання, не дозволяють швидко реагувати на відхилення і приймати ефективні рішення щодо підтримання комфортних умов.

З розвитком інформаційних технологій з’явилася можливість створювати автоматизовані системи моніторингу мікроклімату, які в режимі реального часу збирають, аналізують і візуалізують дані з різних сенсорів, а також швидко інформують користувачів про критичні зміни параметрів. Такі системи покращують управління мікрокліматом, зберігають матеріальні цінності, захищають здоров’я людей і підвищують енергоефективність.

Потреба у простому, надійному та сучасному інструменті для моніторингу мікроклімату стає дедалі актуальнішою для власників приміщень, адміністраторів будівель, персоналу установ і організацій. Сучасні вебтехнології дозволяють створювати такі системи у вигляді зручних багатофункціональних вебзастосунків із доступом з будь-якого пристрою.

З огляду на це виникає потреба у розробці інформаційної системи, яка має:

- автоматично збирати показники мікроклімату з сенсорів у реальному часі;
- зберігати та обробляти дані у хмарному сховищі;
- відображати параметри у вигляді динамічних графіків і таблиць;
- надсилати сповіщення про критичні відхилення;
- забезпечувати багаторівневий доступ та захист даних.

Мета цієї кваліфікаційної роботи — розробка вебзастосунку — інформаційної системи для моніторингу параметрів мікроклімату у закритих приміщеннях, яка зробить управління зручним, функціональним, надійним і швидким.

Об’єкт дослідження — створення програмного забезпечення для моніторингу мікроклімату у приміщеннях.

Предмет дослідження — методи та засоби збору, зберігання, аналізу, візуалізації і швидкого інформування про параметри мікроклімату у вебсередовищі.

Завдання дослідження:
- проаналізувати існуючі рішення у сфері моніторингу мікроклімату;
- сформулювати вимоги до системи;
- обґрунтувати вибір технологій та інструментів;
- розробити архітектуру програмного забезпечення;
- створити вебзастосунок відповідно до технічного завдання;
- провести тестування і оцінити зручність використання системи.

Для реалізації системи використано сучасний вебстек. Основний фреймворк — Next.js 15, що забезпечує швидкий серверний і

клієнтський оброблювальний процес. Серверна логіка, API-запити і рендеринг базуються на Node.js. Інтерфейс створений за допомогою React, а стиль — Tailwind CSS. Для роботи з даними у реальному часі використано Convex, що синхронізує дані з пристроїв і користувачів без затримок. Авторизацію забезпечує Clerk. Обмін даними із сенсорів здійснюється через MQTT-протокол. Оповіщення — через Telegram-бота.

Розроблена система має практичне значення для багатьох користувачів, які займаються управлінням мікроклімату у приміщеннях. Вона дозволяє:

- автоматично отримувати і контролювати параметри;
- зберігати історію у хмарі і мати доступ з будь-якого пристрою;
- швидко реагувати на критичні відхилення з автоматичними сповіщеннями;
- користуватися сучасним і високопродуктивним інтерфейсом.

У рамках цієї бакалаврської роботи створено повний цикл — від аналізу до готового продукту. Розроблена система відповідає сучасним вимогам і може бути впроваджена для моніторингу мікроклімату.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. **Опис предметної області**

Предметом цієї роботи є моніторинг кліматичних параметрів у закритих приміщеннях із застосуванням сучасних апаратних і програмних засобів. Під закритим приміщенням розуміється простір, обмежений стінами, стелею та підлогою, призначений для перебування людей, зберігання матеріалів або проведення виробничих і побутових процесів. До таких приміщень належать житлові кімнати, офіси, аудиторії, лабораторії, склади, виробничі цехи тощо.

Якість мікроклімату визначається рядом параметрів, зокрема температурою повітря, відносною вологістю, рівнем пилу (забрудненості) та атмосферним тиском. Відхилення цих параметрів від нормативних значень може негативно впливати на самопочуття й здоров’я людей, сприяти псуванню матеріалів і продукції, знижувати ефективність праці або навчального процесу.

Задача моніторингу полягає у безперервному контролі параметрів мікроклімату за допомогою сенсорів, зборі, зберіганні й передачі інформації користувачу або системі управління. Для цього використовуються апаратні платформи з вбудованими мікроконтролерами (наприклад, ESP32), цифрові сенсори температури, вологості, тиску, датчики якості повітря, а також програмне забезпечення для обробки, передачі й візуалізації даних (наприклад, через MQTT-брокер, месенджер Telegram тощо).

Завдяки розвитку Інтернету речей (IoT) і доступності недорогих мікроконтролерів з’явилася можливість створювати автоматизовані системи моніторингу, які забезпечують своєчасне інформування користувача про стан мікроклімату, аналізують тенденції та запобігають виникненню несприятливих умов у закритих приміщеннях.

1.2. **Аналіз вимог до програмної системи**

При створенні програмного забезпечення для моніторингу клімату в закритих приміщеннях встановлюються такі вимоги:

Функціональні вимоги:

- Вимірювання параметрів мікроклімату: система має збирати дані з сенсорів температури, вологості, тиску та пилу у режимі реального часу.
- Передача даних користувачу: зібрані дані слід передавати через MQTT-брокер і додатково через Telegram.
- Інформування про критичні значення: при виході параметрів за межі допустимих значень система має надсилати повідомлення користувачу в Telegram.
- Ведення журналу вимірювань: можливість зберігати отримані дані для подальшого аналізу (через MQTT або зовнішню базу даних).
- Масштабованість: можливість підключення додаткових сенсорів або розширення системи для кількох приміщень.

Нефункціональні вимоги:

- Точність: система має гарантувати достовірність даних згідно з характеристиками сенсорів.
- Надійність: забезпечити стабільну роботу протягом тривалого часу без збоїв.
- Швидкодія: передача і обробка даних повинні відбуватися з мінімальними затримками.
- Безпека: захищати передачу даних від несанкціонованого доступу, особливо у публічних мережах.
- Зручність: інтерфейс (наприклад, у Telegram) має бути зрозумілим та інформативним.
- Доступність і дешевизна: використовувати доступні компоненти, такі як ESP32, DHT22, BMP180, і уникати великих витрат на розгортання.

Технічні вимоги:

- MQTT-брокер: для передачі даних від сенсорів до системи.
- Node-RED: платформа для обробки, маршрутизації та візуалізації даних.
- Мікроконтролер ESP32: для підключення сенсорів і збору інформації.
- Сенсори DHT22, BMP180, SDS011 або інші: для вимірювання температури, вологості, тиску та пилу.
- Веб-інтерфейс: для перегляду та аналізу даних через браузер.
- Telegram-бот: для надсилання повідомлень при виході параметрів за межі.
- Живлення: від доступних джерел для сенсорних модулів.
- Підтримка масштабування: додавання нових сенсорів без суттєвої зміни архітектури.
- Операційна система: сумісна з актуальними версіями Windows, Linux або прошивкою мікроконтролерів.

Аналіз вимог допомагає визначити ключові функції та обмеження системи, що дозволяє створити ефективну, масштабовану та зручну систему моніторингу клімату в закритих приміщеннях.

1.3. **Моделювання предметної області**

Моделювання предметної області є важливим етапом у розробці систем моніторингу мікроклімату, оскільки дозволяє формалізувати знання про систему та її взаємодію з навколишнім середовищем. Цей процес допомагає чітко визначити функціональні вимоги, виявити потенційні проблеми на ранніх стадіях та створити міцну основу для подальшого проектування системи.

Уніфікована мова моделювання (UML) забезпечує потужний інструментарій для візуалізації та проектування системи. Для системи обліку параметрів мікроклімату особливо важливі такі діаграми:

Діаграми варіантів використання (Use Case Diagrams):

— відображають взаємодію користувачів (адміністратора, технічного персоналу, пересічних користувачів) з системою

— включають сценарії моніторингу параметрів, генерації звітів, налаштування порогових значень, відправки сповіщень

Діаграми класів (Class Diagrams):

— описують основні сутності системи: "Датчик", "Параметр мікроклімату", "Вимірювання", "Сповіщення", "Користувач"

— зображують зв'язки між цими сутностями, їх атрибути і методи

Діаграми послідовностей (Sequence Diagrams):

— демонструють потоки взаємодії між компонентами системи під час виконання ключових операцій, наприклад, збір даних з датчиків, обробка та збереження у базі даних

Діаграми діяльності (Activity Diagrams):

— візуалізують бізнес-процеси системи, наприклад, алгоритм реагування на критичні значення параметрів
— описують потоки даних між апаратним та програмним забезпеченням

Діаграми станів (State Diagrams):
— відображають життєвий цикл основних об'єктів системи, наприклад, зміну станів датчиків (активний, неактивний, калібрування, помилка)

Для системи обліку параметрів мікроклімату важливі такі абстракції:
— фізичні датчики (температури, вологості, якості повітря)
— вимірювані параметри (поточні значення, історичні дані)
— правила аналізу даних (порогові значення, алгоритми виявлення аномалій)
— користувацькі інтерфейси (веб-дашборд, мобільний додаток, Telegram-бот)
— система сповіщень та звітності

Моделювання цих абстракцій у UML дозволяє чітко визначити архітектуру системи, її компоненти та взаємодії, а також забезпечити узгодженість між технічними вимогами і бізнес-потребами. Це особливо важливо для систем моніторингу мікроклімату, де точність вимірювань, надійність та зручність інтерфейсів є ключовими факторами успішної роботи.

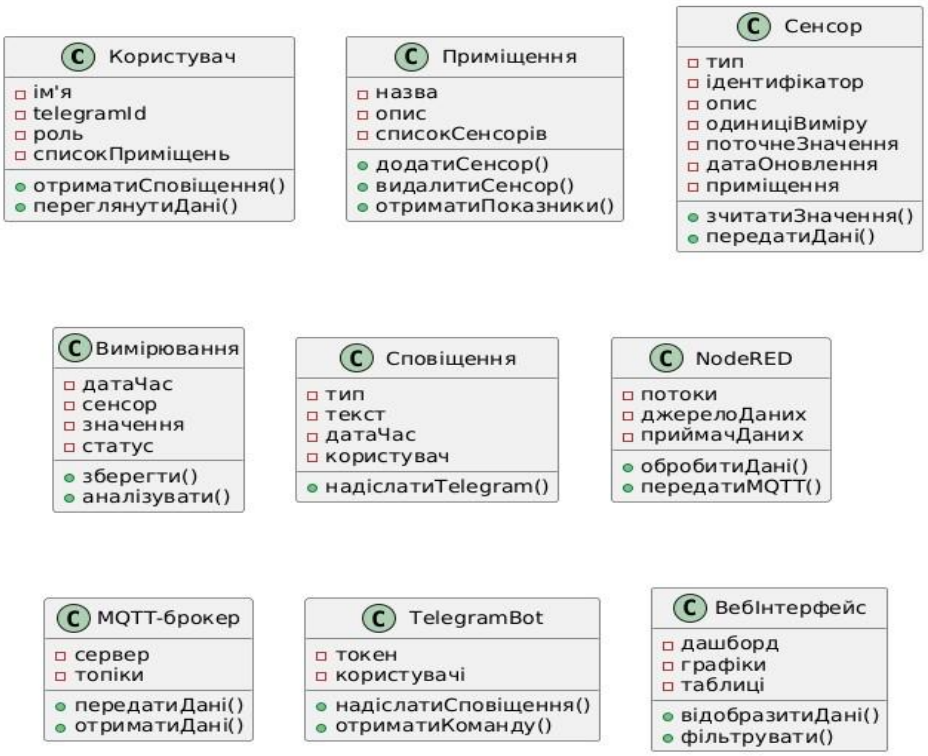

Рис. 1.1. Основні абстракції системи

Абстракція «Користувач» має властивості ім'я, telegramId, роль і список приміщень. Користувач може використовувати методи отриматиСповіщення() і переглянутиДані() для контролю стану середовища у вибраних зонах.

Абстракція «Приміщення» містить властивості: назву, опис та список сенсорів. Ця сутність відповідає за групування технічних засобів контролю у конкретній закритій зоні за допомогою методів addSensor(), removeSensor() та getMetrics().

Абстракція «Сенсор» зберігає такі характеристики, як тип, ідентифікатор, опис, одиниці виміру, поточне значення, дата оновлення та приміщення. Відповідальність цієї абстракції полягає у безпосередній взаємодії з фізичним середовищем за допомогою методів зчитатиЗначення() та передатиДані().

Абстракція «Вимірювання» включає такі властивості, як дата/час, сенсор, значення та статус. Вона відповідає за фіксацію кліматичних параметрів у базі даних за допомогою методів зберегти() та аналізувати().

Абстракція «Сповіщення» містить властивості: тип, текст, дата, час і користувач. Основна функція цієї сутності — швидке повідомлення про стан мікроклімату через метод onSendTelegram().

Діаграма варіантів використання (прецедентів) (Use case diagram) – це одна з графічних діаграм UML, яка застосовується для моделювання програмного забезпечення, зокрема, об’єктно-орієнтованих систем. Вона визначає концептуальну модель системи. Діаграма прецедентів слугує основою для подальшого розроблення ПС у вигляді фізичних і логічних моделей.

На рисунку 1.2 зображена діаграма прецедентів системи моніторингу клімату, яка демонструє взаємодію основних користувачів, апаратних модулів та зовнішніх сервісів. Після автентифікації користувач може взаємодіяти з веб-інтерфейсом, що містить дашборд, графіки і таблиці, реалізуючи прецеденти відобразитиДані() та фільтрувати(). Для автоматизації потоків даних використовується платформа Node-RED, яка через методи обробитиДані() та передатиMQTT() координує маршрутизацію інформації від сенсорів. Обмін повідомленнями між апаратною і серверною частинами забезпечується MQTT-брокером, що оперує сервером і топіками для функцій передатиДані() та отриматиДані(). Системний модуль TelegramBot використовує токен і список користувачів для виконання прецедентів надіслатиСповіщення() у разі відхилення показників від норми та отриматиКоманду() для отримання інформації про клімат за прямим запитом користувача. Усі процеси системи починаються з ініціалізації пристроїв і авторизації користувачів, після чого кожен компонент має доступ до функцій відповідно до своєї ролі в архітектурі моніторингу.

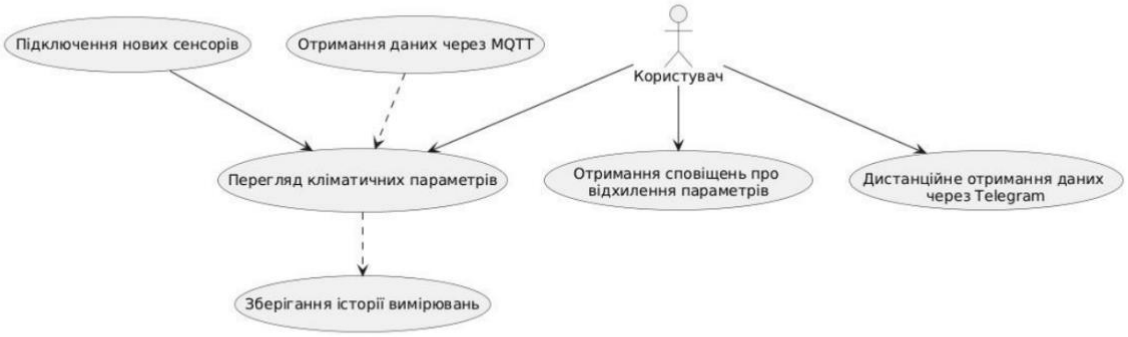

Рис. 1.2. Діаграма прецедентів системи

1.4. Опис основних абстракцій системи моніторингу клімату

Абстракція «Приміщення» (кімната/зона контролю) включає всі властивості об'єкта моніторингу, такі як id, назва, опис, дата створення, дата останнього оновлення, organizationId і перелік підключених віртуальних або фізичних датчиків. Це центральний документ-контейнер системи, відповідальний за структуру кімнат, розмежування прав доступу між користувачами, відображення поточного мікроклімату та координацію спільного перегляду даних у реальному часі.

Абстракція «Користувач». Користувач має властивості id, ім’я, email, роль у системі (Адміністратор / Оператор), avatar та прив’язку до месенджера (telegram_id). Він може реєструватися й авторизуватися через сервіс Clerk, створювати нові картки приміщень для моніторингу, редагувати їхні метадані або видаляти їх, брати участь у спільних сесіях для спостереження за показниками через інтерактивний дашборд, а також налаштовувати профіль для отримання відповідної інформації.

Абстракція «Показник клімату» (метрика) зберігає ідентифікатор, тип параметра (наприклад, температура в °C або відносна вологість у %), поточне числове значення та порогові межі норми. Основна функція цієї абстракції —

структурувати сирі дані, що надходять до системи, і передати актуальні кліматичні метадані клієнтському інтерфейсу (Next.js) та графікам для їх відображення користувачу.

Абстракція «Журнал вимірювань» (Історія замірів) містить такі властивості, як id — ідентифікатор приміщення, значення температури й вологості, дата та точний час фіксації (timestamp), а також джерело даних (інтерфейс системи, автоматичний датчик або Telegram-бот). Вона відповідає за збереження часових рядів у базі Convex, створення історичних даних для аналізу мікроклімату та інтеграцію з сервісами запиту інформації.

Абстракція «Організація» містить ідентифікатор, назву, список учасників (адміністраторів та операторів) і перелік спільних об'єктів (будівель, кімнат, журналів). Ця сутність забезпечує ізоляцію кліматичних даних у межах компанії або підприємства, управління правами доступу, централізоване збереження інформації та координацію роботи персоналу в режимі реального часу.

1.5. Обґрунтування діаграми варіантів використання

Діаграма варіантів використання (прецедентів) (Use case diagram) є однією з графічних діаграм UML, яка застосовується для моделювання програмних систем, зокрема об’єктно-орієнтованих. Вона визначає концептуальну модель системи. Діаграма прецедентів слугує основою для подальшої деталізації програмного забезпечення у вигляді фізичних і логічних моделей.

На рисунку 1.2 зображено діаграму прецедентів системи моніторингу клімату, яка демонструє взаємодію трьох основних акторів: **оператора**, **адміністратора** та **Telegram-бота** (як системного актора).

Оператор після входу в систему користувач може переглядати картки приміщень, відстежувати поточні параметри температури та вологості в реальному часі, виконувати пошук кімнат за ключовими словами через рядок пошуку з debounce-ефектом, аналізувати історію вимірювань за заданий період та взаємодіяти з інтерактивним інтерфейсом спільно з колегами за допомогою Liveblocks. Деякі дії також включають інші функції: наприклад, спільний моніторинг приміщення показує активних користувачів у кімнаті, а запит поточної інформації — перевірку прав доступу користувача в Convex.

Адміністратор. Додатково вона має можливість керувати структурою організаційних об'єктів, створювати нові картки приміщень, видаляти застарілі зони контролю, налаштовувати права доступу для нових співробітників у межах системи Clerk, конфігурувати інтеграцію з API Telegram та управляти загальними параметрами моніторингу.

Telegram-бот виступає автономним інтерфейсом системи, який зв'язується з базою даних через API. Забезпечує виконання прецедентів авторизації чат-користувача, обробляє вхідні команди (наприклад, /status), миттєво зчитує останні збережені дані мікроклімату з таблиць Convex та надсилає текстові повідомлення з поточною температурою і вологою в кімнаті через месенджер

на запит користувача. Усі користувачі починають з автентифікації (через веб-інтерфейс Clerk або верифікацію telegram_id), після чого отримують доступ до функцій моніторингу залежно від своєї ролі.

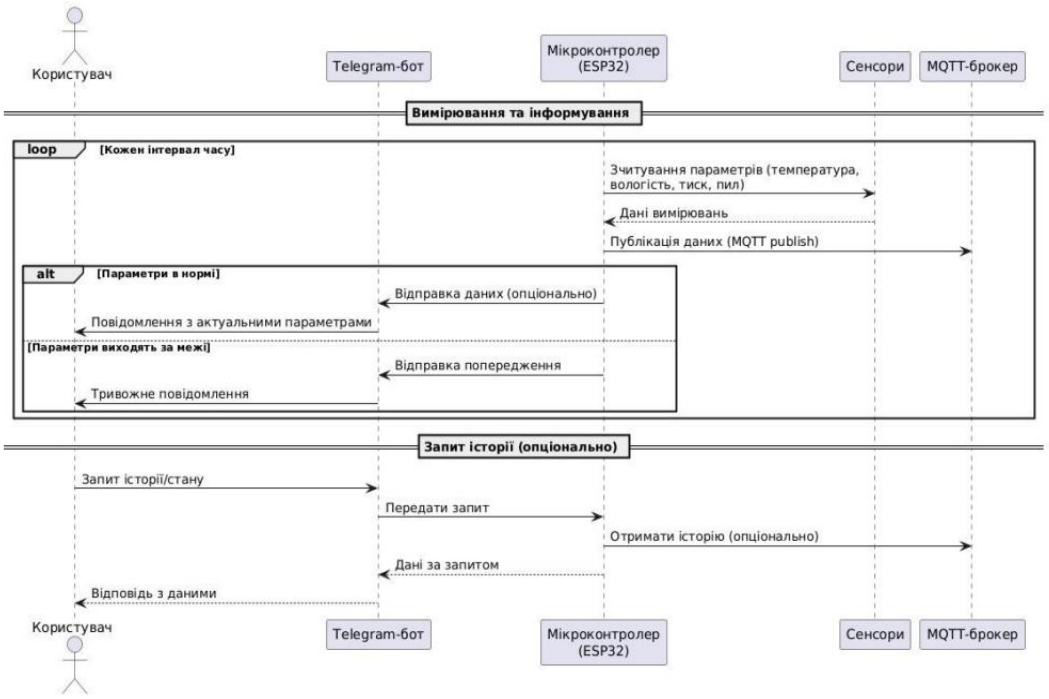

Рис. 1.3. Діаграма послідовності до проектованої системи

Діаграма послідовності (Sequence diagram) відображає часовий порядок обміну повідомленнями між об'єктами та компонентами системи під час ключових процесів моніторингу клімату. На рисунку 1.3 (image_aa67bb.png) показано детальний сценарій взаємодії між користувачем, клієнтським інтерфейсом, маршрутними інструментами та серверною частиною системи.

Процес починається з того, що користувач запускає дію через веб-інтерфейс. Клієнтська частина надсилає запит на відображення актуальних даних мікроклімату до сервера, який у відповідь звертається до БазиДаних Convex через метод отриматиДані(). БазаДаних Convex швидко повертає стан приміщень і актуальні показники клімату на сервер, а той передає структуровану інформацію назад у ВебІнтерфейс для візуалізації користувачем за допомогою динамічних компонентів дашборду.

Одночасно в системі здійснюється безперервний збір даних з обладнання контролю середовища. Платформа автоматизації NodeRED зчитує сирі дані з підключених пристроїв через метод обробитиДані() та трансформує їх у потрібний формат. Після обробки NodeRED викликає метод передатиMQTT() для передачі інформації до MQTT-брокера — центрального вузла комунікації.

MQTT-брокер виконує маршрутизацію повідомлень і через метод передатиДані() надсилає нові кліматичні показники на Сервер додатка. Сервер обробляє отримані дані температури та вологості і викликає функцію зберегти() для фіксації нового запису у журналі вимірювань БазиДаних Convex. Завдяки реактивності СУБД Convex, після успішного збереження даних автоматично активується зворотній зв'язок, і оновлені показники мікроклімату миттєво з'являються у ВебІнтерфейсі, забезпечуючи актуалізацію графіків і індикаторів у реальному часі без потреби ручного оновлення сторінки користувачем.

1.4.1 Аналоги та існуючі рішення

На сучасному ринку автоматизації та клімат-контролю представлено кілька популярних систем, які частково або повністю реалізують функціонал збору, візуалізації та аналізу параметрів мікроклімату в приміщеннях.

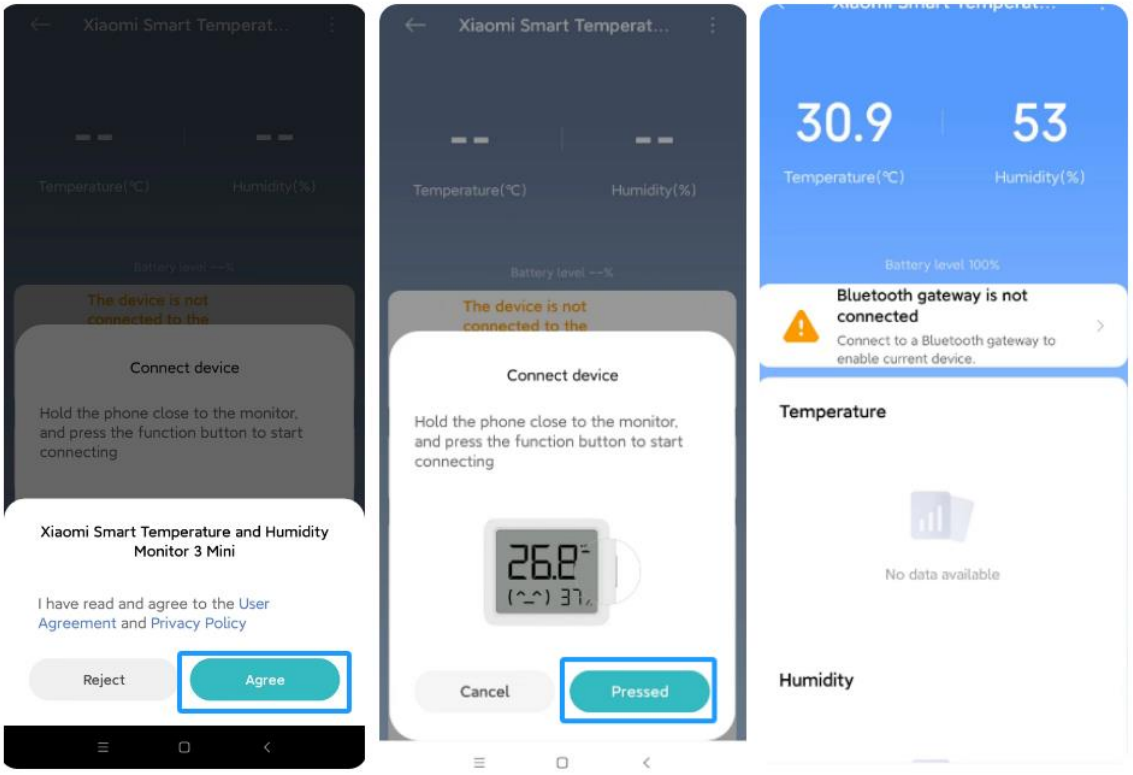

Рис. 1.4. Скріншот роботи Додатка «Xiaomi Mi Home»

Система Xiaomi Mi Home — одна з найпоширеніших екосистем для моніторингу клімату вдома. Вона пропонує мобільний інтерфейс, підтримує бездротові датчики температури та вологості, а також дає змогу створювати автоматичні сценарії. Процес підключення й інтерфейс мобільного додатка під час синхронізації з пристроєм показано на рисунку 1.4, де представлені етапи з'єднання, підтвердження умов використання та виведення актуальних параметрів середовища. Однак важливими недоліками системи є її закрита архітектура, жорстка прив'язка до пропрієтарних серверів, залежність від безпосереднього підключення шлюзів та обмежені можливості кастомізації інтерфейсу дашборду під індивідуальні потреби користувача або підприємства.

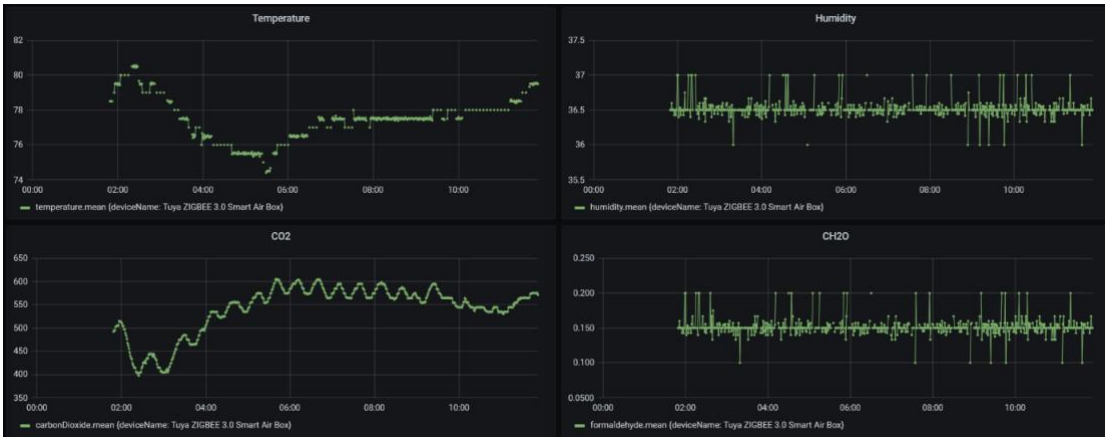

Рис. 1.5. Скріншот роботи сервісу « Tuya Smart »

Платформа Tuya Smart — це універсальна IoT-система, яка дозволяє інтегрувати різноманітні кліматичні сенсори від різних виробників і забезпечує роботу в режимі реального часу. На рисунку 1.5 показано приклад інтерфейсу системи аналітики часових рядів даних для пристроїв лінійки Tuya (image_ab5bfa.png). Графіки забезпечують безперервний моніторинг таких параметрів, як температура, відносна вологість, рівні діоксиду вуглецю (CO2) та формальдегіду (CH2O). Основні недоліки системи — це складність розгортання індивідуальних рішень, залежність від сторонніх хмарних платформ і сервісів аналітики, а також відсутність гнучких інструментів для безшовної координації операторів без додаткових платних модулів.

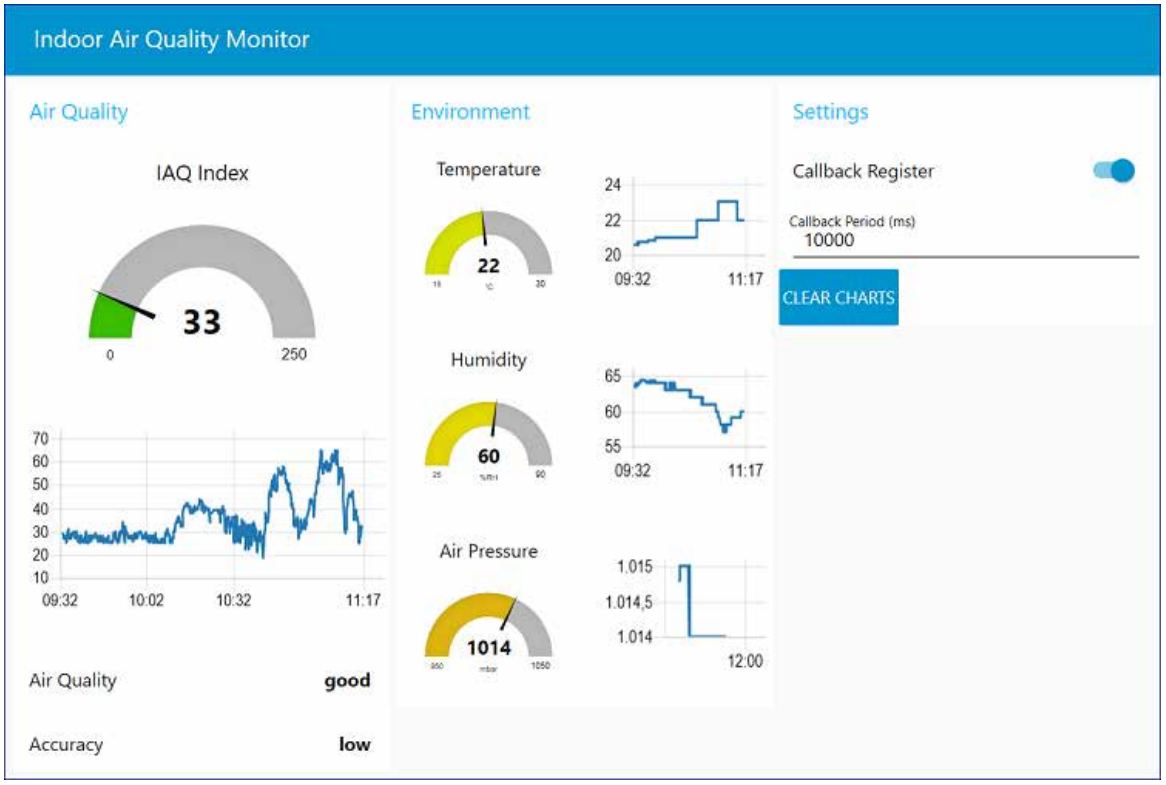

Рис. 1.6. Скріншот роботи сервісу « Node-RED »

Промислові системи моніторингу, такі як Edge-рішення на базі Node-RED, є потужними інструментами для збору та аналізу часових рядів кліматичних даних. Вони забезпечують швидку обробку інформації через MQTT-брокери та детальну графічну візуалізацію. Наприклад, веб-інтерфейс панелі керування на основі сучасної графічної оболонки Grafana, підключеної до мікроконтролера ESP32, показано на рисунку 1.6 (image_ab5bbe.png). Інформаційна панель має стрілочні індикатори вологості, цифрові — для тиску (hPa), температури та загальний графік динаміки змін показників (Overview). Однак такі системи вимагають висококваліфікованого персоналу для адміністрування локальних серверів, не мають вбудованих хмарних модулів багаторівневої автентифікації користувачів і є надмірно громіздкими для швидкого розгортання легких мобільних рішень із елементами координації.

Таблиця 1.1 – Порівняння аналогів інформаційної системи NoteMinds

Критерій порівняння	**Розроблюване програмне забезпечення**	**Система Xiaomi Mi Home**	**Платформа Tuya Smart**	**Комплекс Node-RED та Grafana**
Тип інтерфейсу користувача	Веб-орієнтований мобільний дашборд	Мобільний додаток для смартфонів	Мобільний додаток та веб-панель	Веб-панель керування (Dashboard)
Архітектура бази даних	Реактивна хмарна СУБД Convex	Закриті пропрієтарні сервери	Хмарна інфраструктура Tuya IoT	Локальна СУБД часових рядів
Підтримка реального часу	Так, автоматична синхронізація WebSocket	Так, через внутрішні протоколи	Обмежено, залежно від тарифного плану	Так, через постійне опитування шлюзів
Спільний моніторинг кількома операторами	Так, інтегрована інфраструктура Liveblocks	Ні, доступ за замовчуванням лише для одного профілю	Ні, обмежено функціями сімейного доступу	Ні, перегляд спільний але без відстеження дій
Автентифікація та поділ ролей	Так, інтегрований хмарний сервіс Clerk	Ні, авторизація прив'язана до єдиного аккаунту	Обмежено, базові налаштування доступу пристроїв	Ні, базовий пароль на всю панель без поділу ролей
Сповіщення користувачів	Автономний інтегрований Telegram-бот	Внутрішні пуш-сповіщення додатка	Пуш-сповіщення або платні SMS-шлюзи	Через сторонні плагіни та ручне

Критерій порівняння	**Розроблюване програмне забезпечення**	**Система Xiaomi Mi Home**	**Платформа Tuya Smart**	**Комплекс Node-RED та Grafana**
				налаштування
Кастомізація під конкретне приміщення	Повна, відкритий вихідний код компонента	Ні, жорстко обмежена шаблонами виробника	Частково, через конфігуратор віджетів Tuya	Висока, вимагає програмування графічних віджетів
Складність розгортання та адміністрування	Низька, автоматичний деплой на Vercel	Низька, орієнтована на побутового користувача	Середня, вимагає реєстрації в системі розробника	Висока, необхідне ручне налаштування серверів та MQTT

1.4.2 Відмінності запропонованого рішення

Програмне забезпечення для моніторингу клімату в закритих приміщеннях, що розробляється в рамках цієї бакалаврської роботи, має низку важливих технологічних та архітектурних переваг, які відрізняють його від наявних комерційних і побутових рішень. Запропонована система орієнтована на максимальну доступність і простоту взаємодії користувача без потреби у встановленні складних мобільних додатків чи додатковому налаштуванні локальних шлюзів. До інтерактивного дашборду, динамічних графіків та журналів вимірювань можна отримати доступ через будь-який сучасний веб-браузер на мобільних або десктопних пристроях. Важливою особливістю архітектури є використання реактивної хмарної бази даних Convex, яка автоматично оновлює показники температури та вологості для всіх підключених користувачів через стабільне WebSocket-з'єднання. Це виключає необхідність у важких періодичних запитах і зменшує навантаження на сервер. У систему інтегровано інфраструктуру Liveblocks для спільного моніторингу простору, що дозволяє кільком спеціалістам, операторам або інженерам одночасно відстежувати зміни параметрів мікроклімату в реальному часі, бачити присутність інших користувачів у кімнаті контролю та координувати дії під час аварій. Безпека даних і ізоляція кліматичних об'єктів забезпечуються завдяки інтеграції хмарного сервісу автентифікації Clerk. Система дозволяє гнучко розподіляти користувачів за ролями Адміністратор та Оператор у межах однієї організації, що запобігає несанкціонованому доступу сторонніх до журналів. Створено автономного Telegram-бота, який слугує швидким і зручним інтерфейсом для взаємодії з базою даних і дозволяє користувачам миттєво отримувати останні параметри мікроклімату в конкретному приміщенні в месенджері за допомогою простої команди, зменшуючи навантаження на інтернет і час очікування.

1.6. **Постановка завдання**

Основною метою цієї роботи є створення програмного забезпечення для моніторингу клімату в закритому приміщенні, яке забезпечує автоматизований збір, збереження, обробку та відображення даних про стан середовища в реальному часі для підвищення ефективності контролю та зручності управління мікрокліматом. Система повинна бути орієнтована на операторів, інженерів та адміністраторів закритих приміщень, забезпечуючи швидкий доступ до інформації як через веб-інтерфейс, так і через мобільний месенджер.

Щоб досягти поставленої мети, потрібно виконати такі ключові завдання:

Розробити веб-інтерфейс із сучасним і зрозумілим дашбордом користувача, орієнтованим на наочне відображення динамічних графіків і таблиць із показниками температури, вологості та тиску в приміщеннях.

Реалізувати механізм реєстрації та автентифікації користувачів із підтримкою розподілу за ролями «Адміністратор» та «Оператор» у межах конкретної організації.

Забезпечити архітектуру збору даних із фізичних або віртуальних сенсорів у режимі реального часу, використовуючи протокол MQTT та платформу Node-RED для оркестрації й маршрутизації інформаційних потоків.

Реалізувати спільну роботу операторів над моніторингом об'єктів у реальному часі на основі технології Liveblocks, включаючи миттєву синхронізацію змін і відображення активних користувачів у кімнаті контролю.

Вбудувати систему пошуку та фільтрації приміщень, яка дає змогу оперативно знаходити потрібні зони контролю за ключовими словами через індексовану структуру хмарної бази даних.

Розробити модуль збереження часових рядів даних

Інтегрувати систему сповіщень і взаємодії на базі автономного Telegram-бота, який дозволяє користувачам отримувати оперативні показники мікроклімату в приміщенні за допомогою текстових команд безпосередньо в месенджері.

Розробити модуль адміністрування, що дає змогу керувати обліковими записами, створювати та конфігурувати картки приміщень, підключати нові датчики та налаштовувати загальні параметри моніторингу.

У підсумку очікується створення гнучкої, масштабованої та економічно ефективної інформаційної системи, що відповідає актуальним потребам автоматизації та контролю параметрів мікроклімату в закритих приміщеннях.

1.7. **Висновки до розділу 1**

Аналіз предметної області підтвердив зростаючу потребу у створенні надійних і доступних цифрових інструментів для безперервного автоматизованого збору, зберігання та відображення параметрів мікроклімату в закритих приміщеннях. Це зумовлено високими вимогами до підтримання оптимальних умов праці, збереження обладнання та підвищення загальної енергоефективності об'єктів. Детальний аналіз вимог дозволив чітко сформулювати комплекс функціональних критеріїв, серед яких вимірювання показників у реальному часі, гнучка фільтрація приміщень, спільний моніторинг та оперативне інформування користувачів, а також нефункціональних параметрів, таких як висока швидкодія, надійність, безпека та масштабованість архітектури. Формування моделі предметної області з виділенням ключових абстракцій, таких як Користувач, Приміщення, Сенсор, Вимірювання та Сповіщення, забезпечило структуроване теоретичне уявлення про компоненти майбутнього програмного забезпечення та логіку їхньої безпосередньої взаємодії в системі. Проведений огляд інформаційних джерел і наявних на ринку комерційних та відкритих рішень дозволив порівняти розроблювану систему з популярними аналогами, такими як екосистеми Xiaomi Mi Home, Tuya Smart та комплекси на базі Node-RED і Grafana. Це дало змогу обґрунтувати унікальні переваги запропонованого продукту: використання відкритої веб-архітектури, реактивну хмарну синхронізацію без затримок, надійний поділ ролей та наявність легкого автономного Telegram-бота для швидкого зчитування температури. Чітка постановка завдання окреслила повний комплекс подальших інженерних робіт: від Liveblocks до створення адаптивного веб-інтерфейсу за допомогою Next.js 15, побудови алгоритмів маршрутизації потоків через MQTT та налаштування модуля сповіщень. Таким чином, на основі проведеного системного аналізу та моделювання сформовано вичерпну теоретичну та концептуальну базу для проєктування архітектури та подальшої практичної реалізації програмного забезпечення у наступних розділах кваліфікаційної роботи.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Логічна модель даних у вигляді ER-діаграми

Логічна модель даних відображає структуру хмарного сховища, типи атрибутів і логічні зв'язки між сутностями, що забезпечують збереження та синхронізацію інформації в системі моніторингу клімату. На рисунку 2. 2.1 — це схема даних, де кожна таблиця описується як набір документів із чіткою типізацією полів.

Центральне місце в моделі даних посідає таблиця користувачів USER, яка містить унікальний ідентифікатор id типу string, що є первинним ключем (PK), системне ім'я name типу string, посилання на месенджер через поле telegramId типу string, а також службове поле ролі користувача role типу string. З цією таблицею логічно пов'язана сутність системних сповіщень NOTIFICATION, що містить первинний ключ id типу string, тип сповіщення type типу string, текстовий вміст повідомлення text типу string, а також мітку часу dateTime типу datetime. Зв'язок між таблицями USER та NOTIFICATION реалізується через відношення один до багатьох, що дозволяє одному користувачеві отримувати багато сповіщень за допомогою методу отримує.

Облік об'єктів контролю середовища реалізовано через таблицю приміщень ROOM, що містить первинний ключ id типу string, назву об'єкта name типу string та детальну характеристику або опис приміщення description типу string. Таблиця USER пов’язана з таблицею ROOM відношенням один до багатьох через логічний зв’язок має список, оскільки один користувач може адмініструвати або переглядати кілька кімнат одночасно.

Кожне приміщення логічно пов'язане з таблицею сенсорів SENSOR, реалізуючи зв'язок один до багатьох через відношення містить список, що дозволяє закріпити за однією кімнатою кілька фізичних або віртуальних датчиків.

Таблиця SENSOR містить метадані пристроїв, включаючи унікальний первинний ключ id типу string, тип датчика type типу string, текстовий опис пристрою description типу string, одиниці виміру параметрів клімату unit типу string, поточне зчитане числове значення показника currentValue типу float, а також дату й час останнього оновлення даних у системі fields updatedAt типу datetime.

Для збереження історії та ретроспективного аналізу параметрів мікроклімату в системі проєктована таблиця вимірювань MEASUREMENT, пов’язана з конкретним датчиком через відношення, яке вона генерує. Кожен документ цієї таблиці реєструє унікальний первинний ключ id типу string, точний час фіксації кліматичного параметра dateTime типу datetime, числове значення показника value типу float, а також текстовий статус проведеного виміру status типу string, що дозволяє накопичувати безперервні часові ряди даних для відображення графіків та аналітики у веб-інтерфейсі системи.

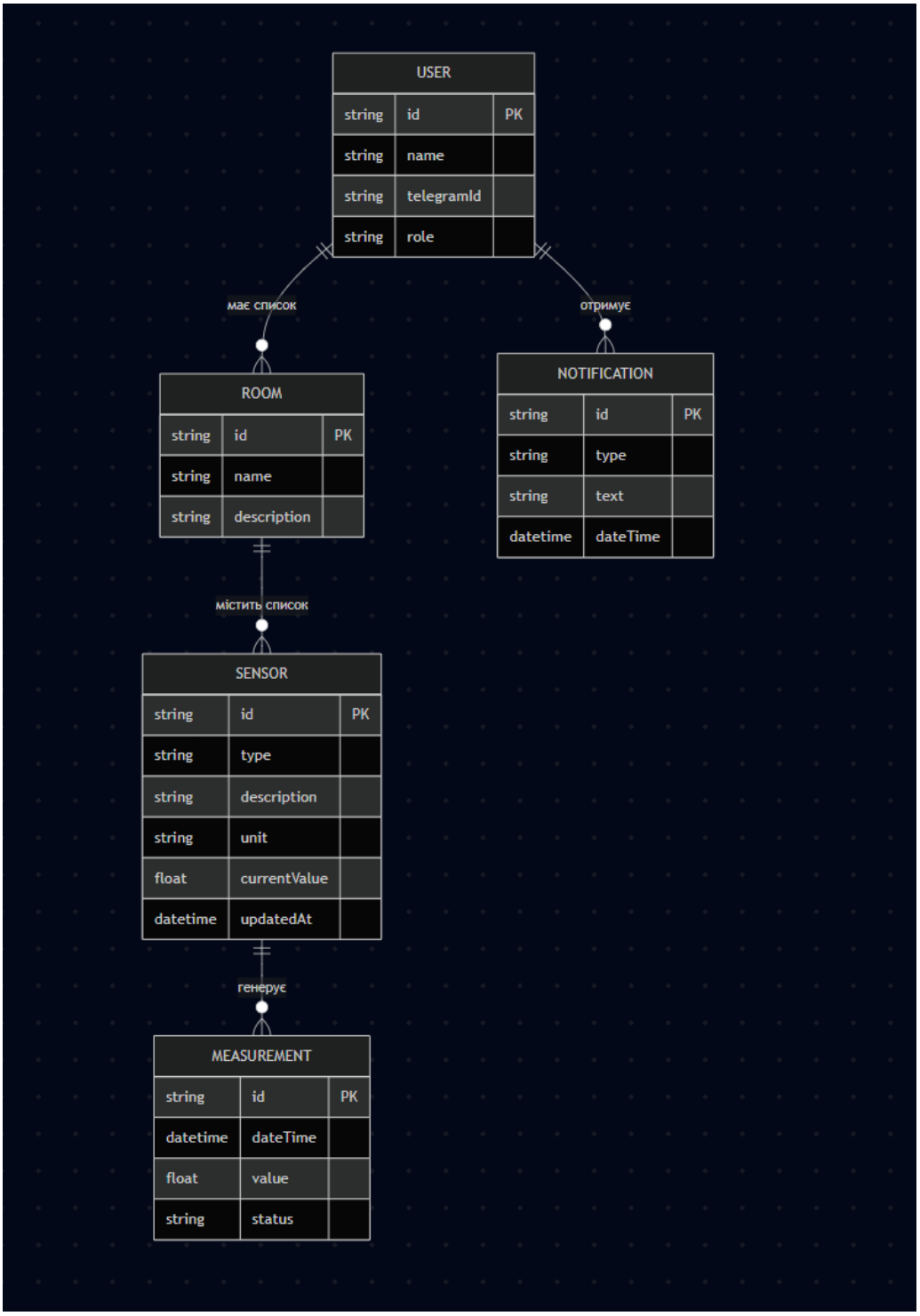

Рис. 2.1. Логічна модель даних у вигляді ER-діаграми

2.2. Діаграма класів та кооперацій

Діаграма класів і кооперацій описує статичну архітектуру системи моніторингу клімату, зокрема внутрішню структуру об'єктів, їхні атрибути, методи та спосіб взаємодії для забезпечення цілісності системи. На рисунку 2. 2.2 показує логічну взаємодію між елементами користувацького інтерфейсу, хмарним сховищем Convex і зовнішніми сповіщеннями.

Основною логікою управління правами доступу та сесіями користувачів займається клас Користувач, який має атрибути ім'я, telegramId та роль. Взаємодія з цим класом реалізується через методи отриматиСповіщення() і переглянутиДані(), що дозволяє співробітникам взаємодіяти з підключеними об'єктами. Клас Користувач пов’язаний асоціативним зв’язком із класом Приміщення, який є основним структурним елементом. Клас Приміщення оперує атрибутами назва, опис і список Сенсорів, а його стан змінюється та зчитується за допомогою методів додатиСенсор(), видалитиСенсор() і отриматиПоказники().

Клас Приміщення агрегує об'єкти класу Сенсор — моделює поведінку фізичних і віртуальних датчиків. Сенсор містить атрибути: тип, ідентифікатор, опис, одиниці виміру, поточне значення та дату останньої фіксації у полі датаОновлення. Логіка класу Сенсор базується на методах зчитатиЗначення() і передатиДані(), які отримують сирі метрики та передають відповідну інформацію. Кожен сенсор пов’язаний із класом Вимірювання, який зберігає часи, значення та статус і має методи зберегти() та аналізувати().

Для оповіщення про критичні відхилення параметрів середовища передбачений клас Сповіщення, який містить атрибути тип, текст, датаЧас, користувач і має метод надіслатиTelegram() для швидкої передачі повідомлень.

Динамічні зв'язки між компонентами системи координуються через інтерфейси та шлюзи. Веб-інтерфейс об'єднує модулі дашборду, графіків і таблиць, надаючи методи відобразити() і фільтрувати() для швидкого моніторингу. Збір і маршрутизація даних з датчиків здійснюються класом NodeRED, який має атрибути конфігурації, список потоків і використовує методи обробитиДані() і передатиMQTT(). Передача даних у реальному часі через MQTT-брокер реалізується методами передатиДані() і отриматиДані(). Мобільний доступ до поточних показників забезпечує клас TelegramBot, що використовує атрибути токен і користувачі та викликає методи надіслатиСповіщення() і отриматиКоманду() для взаємодії з оператором через текстові запити.

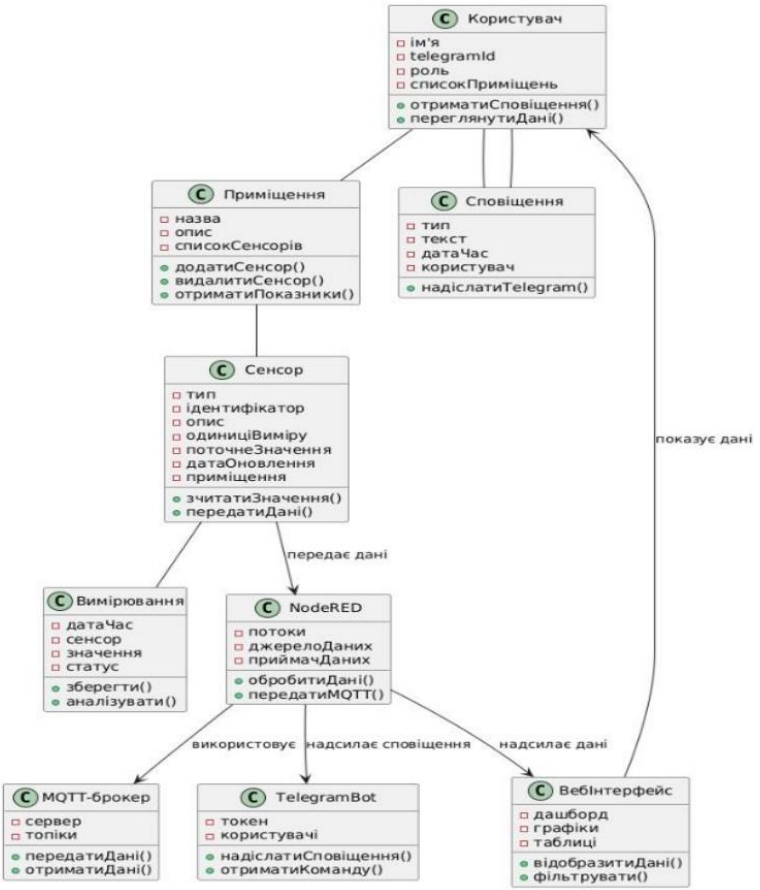

Рис. 2.2. Діаграма класів

2.3. Діаграма пакетів

Діаграма пакетів (Package diagram) відображає структуру компонентів програмного забезпечення на рівні підсистем і логічних модулів, дозволяючи згрупувати класи за їхньою функціональною належністю та визначити залежності між ними. На рисунку 2.3 наведено діаграму пакетів системи моніторингу клімату, яка демонструє модульну архітектуру розподілу обов'язків під час збору та обробки інформації. Головним елементом архітектури є пакет "Керування системою", у якому інкапсульовано базовий керуючий клас "ГоловнаПрограма". Цей пакет координує роботу всіх підсистем, виступаючи єдиною точкою оркестрації та визначаючи логіку роботи від ініціалізації пристроїв до збереження результатів вимірювань у сховищі даних. Пакет "Керування системою" взаємодіє з іншими пакетами через чітко визначені функціональні зв'язки. Зокрема, з пакетом "Підключення до мережі" реалізується встановлення зв'язку: через клас "МережевийМодульWiFi" здійснюється ініціалізація Wi-Fi, а через клас "MQTTМодуль" — надсилаються дані для забезпечення безперервного потоку передачі інформації за бездротовими протоколами. Збір кліматичних показників здійснює пакет "Збір кліматичних даних", який містить клас "КліматичнийСенсор". Взаємодія цього пакета з керуючим полем описується залежністю "отримує показники", що відображає безперервне зчитування температури, вологості, тиску та пилу з фізичного простору. Отримані дані передаються до пакета "Зберігання параметрів", який містить клас "ПараметриМікроклімату". Взаємодія з цим пакетом здійснюється через відношення "формує об'єкт", що дозволяє структурувати часові ряди даних, перетворювати сирі заміри на валідні інформаційні об'єкти та підставляти їх для довгострокового зберігання і реактивного відображення на дашборді користувача.

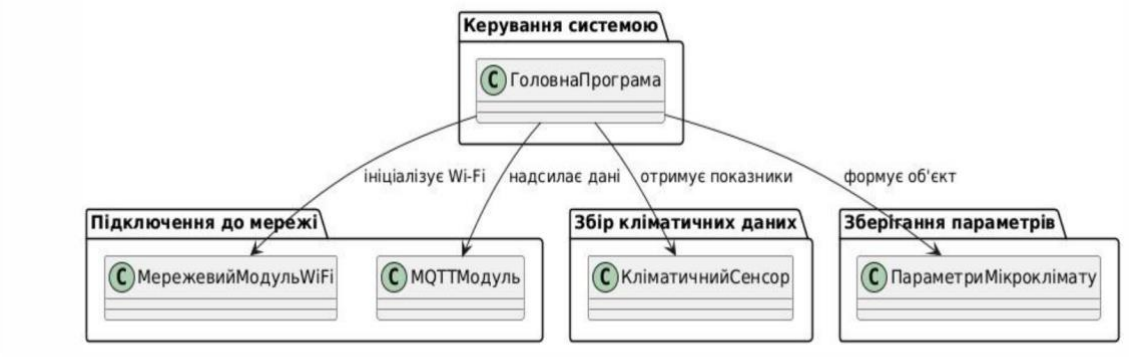

Рис. 2.3. Діаграма пакетів

2.4. Діаграма компонентів

Діаграма компонентів показує фізичну структуру програмного забезпечення, визначаючи модулі коду, архітектурні елементи системи, їхні інтерфейси та залежності, що забезпечують функціонування системи моніторингу мікроклімату як цілісного комплексу. На рисунку 2.4 показано створену діаграму компонентів, яка ілюструє взаємодію між клієнтським веб-додатком, Node-RED та апаратними пристроями.

Основу архітектури користувацького інтерфейсу становить веб-компонент ВебІнтерфейс, який містить підмодулі дашборду, графіків і таблиць, надаючи оператору зручні засоби для візуалізації параметрів середовища. Цей компонент реалізує інтерфейс відобразитиДані() та використовує залежність фільтрувати(). Веб-інтерфейс безпосередньо взаємодіє з серверною частиною через компонент Сервер, що забезпечує перегляд даних() та внутрішню логіку автентифікації користувачів через підключення до хмарного сервісу Clerk.

Збір і накопичення історичних часових рядів даних покладаються на компонент БазаДаних Convex, який має таблиці користувачів, приміщень, сенсорів, вимірювань і сповіщень. Сервер взаємодіє з цим компонентом через інтерфейси зберегти() і аналізувати(), забезпечуючи реактивне оновлення інформації у реальному часі. Також Сервер підтримує зв'язок із компонентом TelegramBot через інтерфейси надіслатиСповіщення() та отриматиКоманду() для обробки текстових запитів користувачів і відправлення екстрених повідомлень у месенджер.

Взаємодія з апаратним рівнем здійснюється через компонент NodeRED, який вміщує конфігурацію та списокПотоків, здійснюючи первинну обробку даних через інтерфейс обробитиДані(). NodeRED підключається до MQTT-брокера через інтерфейс передатиMQTT(). Сам MQTT-брокер має налаштування сервера, топіки та інтерфейси передатиДані() і отриматиДані(), координуючи

стабільний потік кліматичних даних від бездротових модулів. Кінцевий пункт збору даних — апаратний компонент Сенсор, який містить дані про пристрій, його тип і поточне значення, реалізуючи інтерфейси зчитатиЗначення() і передатиДані() для передачі стану кімнат у систему.

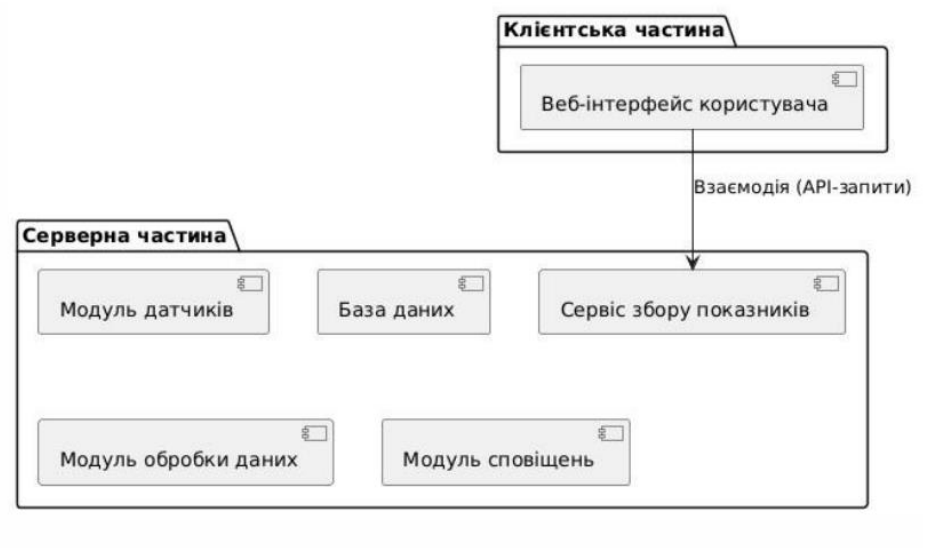

Рис. 2.4. Діаграма компонентів

2.5. **Висновок до розділу 2**

У другому розділі кваліфікаційної роботи виконано повний комплекс завдань з логічного, системного й архітектурного проєктування програмного забезпечення для моніторингу клімату у закритому приміщенні на основі методології об'єктно-орієнтованого аналізу й моделювання UML. Детальний аналіз вимог дозволив побудувати статичну структуру системи у вигляді діаграми класів і кооперацій, визначивши основні сутності, такі як Користувач, Приміщення, Сенсор, Вимірювання і Сповіщення, разом із їхніми атрибутами й методами управління. Також розроблено логіку інтеграції допоміжних інтерфейсних і комунікаційних об'єктів (ВебІнтерфейс, NodeRED, MQTT-брокер, TelegramBot), що забезпечило цілісність усіх рівнів обробки кліматичних метрик. Створено логічну модель даних у вигляді ER-діаграми, адаптовану до специфіки реактивного хмарного сховища Convex, зі структурами таблиць USER, ROOM, SENSOR, MEASUREMENT і NOTIFICATION, із суворою типізацією ключів і атрибутів, та визначено логічні зв'язки один до багатьох для забезпечення надійного збереження та швидкого доступу до історичних часових рядів. За допомогою діаграми пакетів реалізовано модульну архітектуру програми, яка окремо визначає підсистеми управління, мережевого підключення, збору кліматичних даних і зберігання параметрів, що сприяє чіткому розмежуванню відповідальності модулів та мінімізації зв'язків між апаратною й серверною частинами. Описано фізичну структуру через діаграму компонентів, яка показала взаємодію між веб-клієнтом Next.js, сервером автентифікації Clerk, реактивною базою даних Convex, інфраструктурою Node-RED і кінцевими датчиками. Визначені програмні інтерфейси й залежності підтвердили масштабованість, швидке оновлення інтерфейсів через WebSocket і готовність системи до безпосередньої реалізації.

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Система управління інформаційною базою

Важливим етапом проєктування системи моніторингу клімату є вибір та впровадження системи управління базами даних (СУБД), яка має забезпечувати швидку обробку часових рядів, надійне збереження даних та можливість оновлення інтерфейсів користувачів у реальному часі. Для реалізації обрано сучасну хмарну платформу Convex, орієнтовану на документоорієнтовану архітектуру. Вибір SУБД Convex обумовлений її перевагами, які ідеально підходять для задач Інтернету речей (IoT) та розподіленого моніторингу. На відміну від реляційних баз даних, Convex підтримує реактивні запити через постійні WebSocket-з'єднання. Це означає, що при нових вимірюваннях від мікроконтролера, через проміжні сервери, база даних автоматично оновлює стан на всіх активних веб-інтерфейсах без повторного опитування сервера або ручного оновлення сторінки. Управління транзакціями та логікою вибірки реалізовано за допомогою атомарних функцій запитів і мутацій, що виконуються безпосередньо в хмарі на базі рушія V8, забезпечуючи високу продуктивність.

.

3.2. Розробка інформаційної бази

Розробка фізичної структури інформаційної бази базується на попередньо спроєктованій логічній ER-діаграмі. У середовищі Convex схема даних реалізована за допомогою вбудованого інструментарію TypeScript валідації, що гарантує сувору типізацію кожного документа у колекціях.

База даних складається з п'яти основних колекцій. Колекція users зберігає ідентифікатори користувачів, їхні імена, системні ролі та зв’язки з зовнішнім провайдером автентифікації Clerk. Колекція rooms є логічним контейнером для об'єктів контролю, зберігаючи назви та описи приміщень. Фізичні та віртуальні пристрої реєструються в колекції sensors, де для кожного пристрою фіксуються його тип, одиниці виміру, ідентифікатор приміщення та мітка часу останньої активності.

Найбільш навантаженою частиною бази є колекція measurements, яка функціонує як журнал часових рядів. Кожен документ у цій колекції містить унікальний ідентифікатор сенсора, точне значення кліматичного параметра (температури, вологості, тиску або рівня пилу) у форматі числа з плаваючою крапкою та точну мітку часу. Для швидкої вибірки даних для побудови графіків у веб-інтерфейсі поля міток часу та ідентифікаторів сенсорів додатково індексуються. Окрема колекція notifications відповідає за збереження історії системних повідомлень та екстрених сповіщень, пов'язуючи їх із конкретними користувачами.

3.3. Вибір інструментарію для створення прикладного ПЗ

Для успішної реалізації системи моніторингу клімату застосовано сучасний технологічний стек, який охоплює апаратні, проміжні і клієнтські рівні розробки. Для створення клієнтського веб-інтерфейсу (фронтенду) обрано Next.js версії 15, який базується на бібліотеці React. Це забезпечує високу швидкість рендерингу сторінок і ефективну маршрутизацію. Для стилізації інтерфейсу використано Tailwind CSS, що спрощує створення адаптивного дизайну. Багаторівнева автентифікація реалізована через хмарний сервіс Clerk, а модуль спільного моніторингу у реальному часі інтегрований через Liveblocks. На проміжному рівні обробки телеметрії застосовують Node-RED — візуальне середовище для створення логічних потоків маршрутизації даних від мікроконтролерів до бази даних і Telegram-бота. Передача даних мікрокліматичних показників здійснюється за протоколом MQTT через MQTT-брокера. Програмування апаратної частини (мікроконтролерів на архітектурі ESP) виконується мовою C++ із використанням бібліотек для роботи з датчиками та бездротовими мережами.

3.4. Алгоритмізація та програмування програмних модулів

Розробка програмної частини розділена на три незалежні функціональні модулі: збір даних на апаратному рівні, маршрутизація телеметрії та відображення інформації у веб-інтерфейсі.

Першим модулем є алгоритм зчитування та передавання кліматичних даних мікроконтролером. Відповідно до розробленої логіки, після ініціалізації Wi-Fi та MQTT-з'єднання пристрій переходить у безперервний робочий цикл. Блок-схему цього процесу наведено на рисунку 3.1. Алгоритм передбачає послідовне опитування сенсорів: спочатку зчитується температура, потім — вологість, після чого виконується формування або емуляція даних щодо атмосферного тиску та рівня дрібнодисперсного пилу. Зібрані метрики пакуються у повідомлення та відправляються на відповідні топіки MQTT-брокера кожні 15 секунд, що забезпечує стабільний потік даних без перевантаження мережі.

Рис. 3.1. Блок-схема алгоритму збору та передавання кліматичних даних

Другий програмний модуль реалізує серверну обробку та маршрутизацію даних. Вона виконана у середовищі Node-RED, архітектура потоків якого зображена на рисунку 3.2. Система містить вхідні MQTT-вузли, які підписані на топіки температури, вологості, тиску та пилу (наприклад, iotfront/temperature). Отримані повідомлення проходять етап форматування та перенаправляються до вузлів відображення і до хмарної бази даних Convex. Окремим потоком у Node-RED реалізовано Telegram-бота, який через вузол прийому команд відстежує запит користувача, збирає актуальні показники з

глобального контексту, форматує їх у зручний текстовий формат і миттєво надсилає зворотне повідомлення в чат.

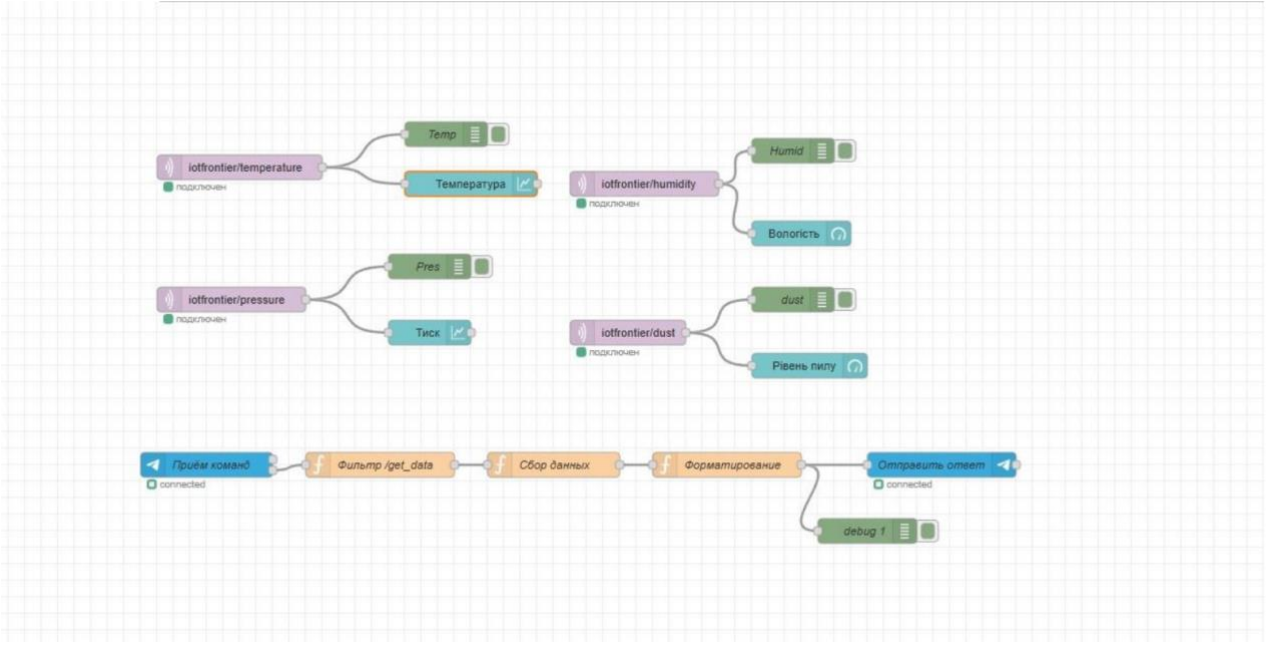

Рис. 3.2. Візуальна схема потоків маршрутизації даних у середовищі Node-RED

Третім модулем є клієнтський веб-інтерфейс для моніторингу. Його програмна реалізація зосереджена на наочності та реактивності. На рисунку 3.3 наведено розроблений дашборд оператора. Верхня панель містить інтерактивні кругові віджети поточного стану, де, наприклад, відносна вологість стилізована під рівень рідини у резервуарі, а рівень пилу має колірне зонування для швидкої оцінки безпеки повітря. Нижня частина екрану містить модулі лінійних графіків (Time-series charts) для ретроспективного аналізу температури та тиску. Завдяки WebSocket-з'єднанню з базою даних Convex, під час надходження нових пакетів даних графіки автоматично добудовують нові точки, а цифри на індикаторах оновлюються без перезавантаження сторінки.

Рис. 3.3. Клієнтський веб-інтерфейс моніторингу мікроклімату

3.5. Висновки до розділу 3

У третьому розділі виконано комплекс робіт із практичної розробки інформаційного та програмного забезпечення системи моніторингу клімату. Обґрунтовано вибір реактивної документоорієнтованої СУБД Couchbase як основного сховища даних та детально описано фізичну структуру створених колекцій. Проведено вибір оптимального інструментарію розробки, який включає Next.js для веб-клієнта, Node-RED для маршрутизації та MQTT для передачі телеметрії. У ході алгоритмізації та програмування успішно реалізовано мікроконтролерний алгоритм безперервного збору показників середовища. Створено серверну логіку маршрутизації потоків даних та розроблено підсистему мобільного інформування на базі Telegram-бота. Розроблено реактивний клієнтський веб-інтерфейс, який забезпечує наочну візуалізацію динаміки мікроклімату через систему кругових індикаторів та лінійних графіків, що повністю відповідає поставленим завданням дипломної роботи.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУТАЦІЇ СИСТЕМИ

4.1. Тестування системи

Тестування розробленого програмного забезпечення моніторингу клімату проводилося у декілька етапів для забезпечення надійності роботи як апаратної, так і програмної частин. На першому етапі було виконано модульне тестування апаратного комплексу. Перевірялася коректність зчитування даних із сенсорів температури, вологості, тиску та пилу мікроконтролером, а також стабільність його підключення до локальної мережі Wi-Fi. Було налаштовано виведення діагностичних повідомлень через послідовний порт (Serial Monitor) для фіксації успішного формування та відправки MQTT-пакетів кожні 15 секунд.

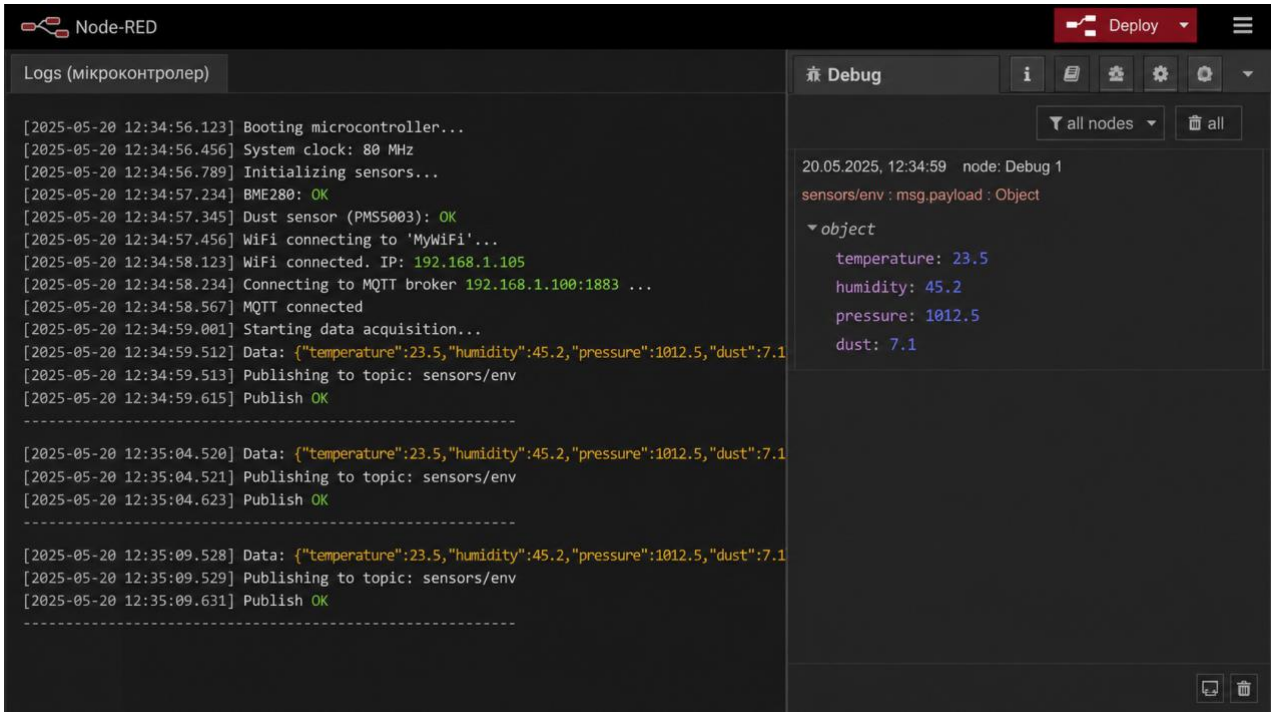

Рис. 4.1. Логування процесів мікроконтролера та перехоплення MQTT-пакетів у панелі Debug середовища Node-RED

На другому етапі проводилося інтеграційне тестування взаємодії між MQTT-брокером, платформою маршрутизації Node-RED та хмарною базою даних Convex. Перевірялася правильність розподілу повідомлень по топіках,

безпомилковість їх перетворення у необхідний JSON-формат перед записом у базу, а також стійкість системи до тимчасової втрати з'єднання. Окремо тестувалася швидкість реакції та коректність роботи підсистеми Telegram-бота на виконання команди отримання актуальних даних.

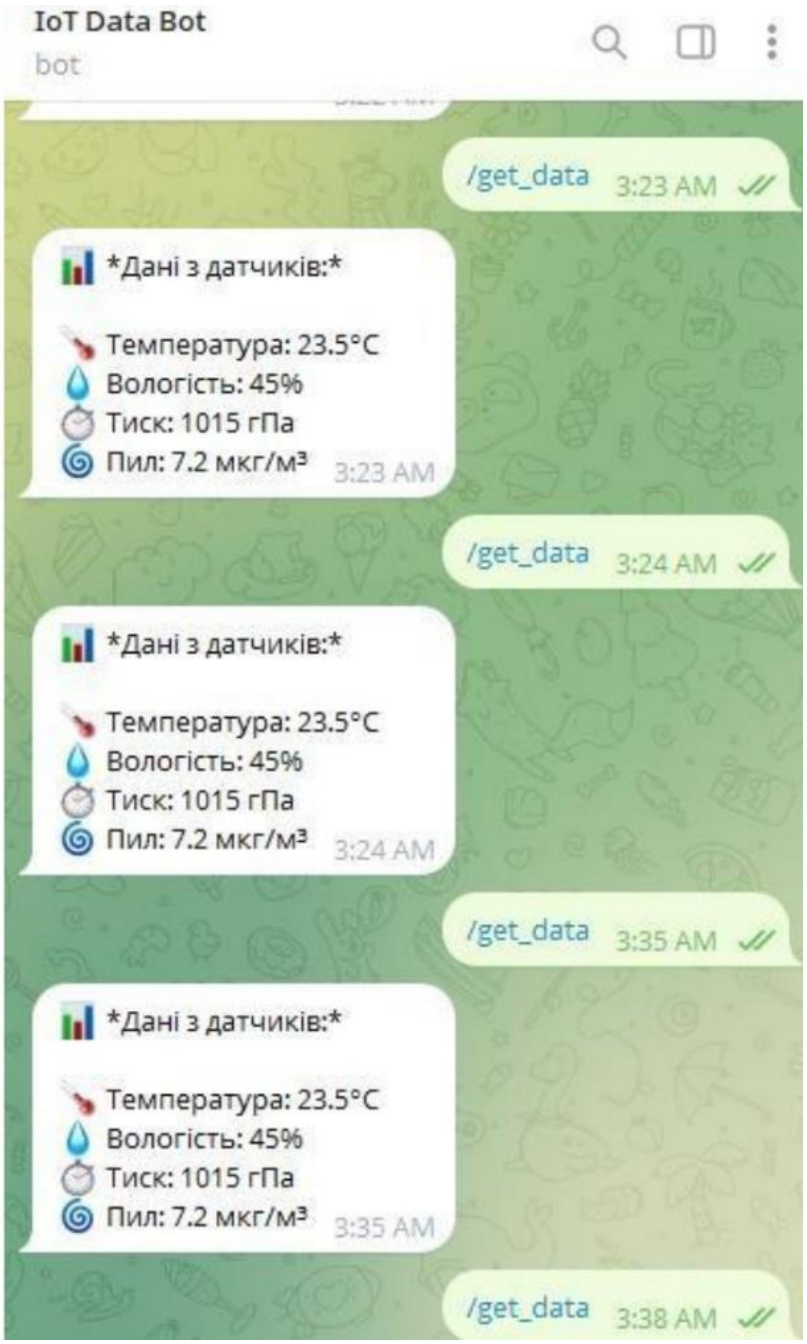

Рис. 4.2. Результати тестування підсистеми інформування через Telegram-бот

Третім етапом стало системне тестування клієнтського веб-інтерфейсу на базі Next.js. Перевірялася адаптивність дашборду на різних пристроях (мобільні телефони, планшети, десктопні монітори), коректність відображення часових рядів на лінійних графіках та правильність заповнення кругових індикаторів. Особлива увага приділялася тестуванню реактивного оновлення даних через WebSocket: було підтверджено, що інтерфейс миттєво відображає нові показники без необхідності ручного перезавантаження сторінки. Результати тестування підтвердили стабільну роботу системи в умовах безперервного потоку даних.

Рис. 4.3. Результати тестування клієнтського веб-інтерфейсу в режимі реального часу

4.2. Вимоги до апаратного та програмного забезпечення

Для успішної реалізації, швидкого розгортання та безперебійної роботи системи моніторингу клімату необхідно дотримуватися певних технічних вимог до апаратної платформи та програмного середовища.

Щодо апаратної частини (фізичного IoT-вузла): потрібен мікроконтролер із підтримкою бездротового зв'язку (рекомендуються плати серії ESP32 або ESP8266), датчик температури та вологості (наприклад, DHT22 або BME280), датчик атмосферного тиску (BMP280), а також датчик рівня дрібнодисперсного пилу. Для стабільної роботи пристрою необхідне стабільне джерело харчування з напругою 5В (наприклад, USB-адаптер) та доступ до Wi-Fi з підключенням до Інтернету.

Що стосується серверного програмного забезпечення: для розгортання проміжного сервера обробки потрібна операційна система на основі Linux (Ubuntu, Debian) або Windows, встановлене середовище виконання Node.js (версії 18.0 або вище), платформа Node-RED та активний MQTT-брокер (наприклад, Eclipse Mosquitto), розміщений у мережі або на локальній машині. Хмарна інфраструктура (база даних Convex та хостинг Vercel) не вимагає локальних ресурсів, але потребує стабільного Інтернет-з'єднання.

Щодо клієнтського забезпечення: програмне забезпечення є кросплатформним, тому до комп'ютера або смартфона оператора не пред'являється особливих апаратних вимог. Важливо мати сучасний веб-браузер (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge), що підтримує HTML5, JavaScript і WebSocket. Для отримання швидких сповіщень та доступу до мобільних функцій користувач повинен мати встановлений месенджер Telegram.

4.3. Склад інсталяційного пакету

Оскільки система має розподілену хмарну веб-архітектуру та складається з апаратно-програмних мікросервісів, класичне поняття інсталяційного файлу (.exe) змінюється на набір вихідних кодів, конфігураційних файлів і скриптів для розгортання. Склад інсталяційного пакета включає: файли мікроконтролера — вихідний код прошивки мовою C (.ino), підготовлений для компіляції у Arduino IDE або PlatformIO, та список сторонніх бібліотек для MQTT-клієнта й сенсорів; файл маршрутизації у форматі flows.json з налаштуваннями вузлів обробки MQTT, логікою Telegram-бота, форматуванням повідомлень і параметрами для хмарної БД; клієнтський веб-додаток — структура із вихідним кодом Next.js, залежностями в package.json, компонентами UI, маршрутизацією, схемами бази даних і функціями мутацій у папці convex, а також файл .env.example із шаблоном змінних; технічна документація — інструкція з процесу компіляції прошивки, імпорту конфігурацій у Node-RED, встановлення NPM-пакетів і деплою веб-додатку на Vercel.

4.4. Висновки до розділу 4

У четвертому розділі дипломної роботи надано детальні рекомендації щодо впровадження, розгортання та подальшої експлуатації програмного забезпечення для моніторингу клімату. Описано методологію системного тестування, яка включає модульну перевірку мікроконтролера, інтеграційне тестування серверних маршрутів у Node-RED і стрес-тестування реактивного веб-інтерфейсу. Результати випробувань підтвердили безпомилковість алгоритмів, стабільність передачі метрик і коректність роботи дашбордів у режимі реального часу. Інакше сформульовані чіткі вимоги до апаратного й програмного забезпечення, що доводять економічну доцільність рішення, адже система базується на доступних мікроконтролерах сімейства ESP, безкоштовному проміжному ПЗ і не вимагає встановлення важких клієнтських додатків, працюючи через стандартний браузер. Детально описаний склад інсталяційного пакету включає вихідні коди прошивки, конфігураційні JSON-файли для маршрутизації потоків, репозиторій веб-додатку Next.js і технічну документацію, що забезпечує швидке розгортання, масштабування й комерційну або академічну експлуатацію на реальних закритих об'єктах.

ВИСНОВКИ

У кваліфікаційній роботі вирішено важливе науково-прикладне завдання: проєктування й розробка комплексного програмного та інформаційного забезпечення для безперервного моніторингу параметрів мікроклімату в закритих приміщеннях. Використання сучасного стеку технологій Інтернету речей (IoT) та хмарних сервісів дозволило створити надійну, масштабовану й реактивну інформаційну систему.

На першому етапі роботи проведено системний аналіз предметної області. Детальне дослідження існуючих аналогів і систем розумного дому допомогло виявити їхні недоліки та сформулювати унікальні вимоги до майбутнього продукту. Створена концептуальна модель абстракцій і діаграма послідовності заклали теоретичний фундамент для правильної організації інформаційних потоків між апаратним забезпеченням, сервером і клієнтськими інтерфейсами.

На етапі логічного й архітектурного проєктування за допомогою UML побудовано статичну структуру системи. Діаграма класів і кооперацій визначили поведінку основних об'єктів системи, а діаграми пакетів і компонентів дали змогу розподілити зони відповідальності модульно. Важливим стало створення логічної ER-моделі даних, яка повністю відповідає конструкції документоорієнтованої бази даних Convex, що забезпечує швидкий індекс і зберігання часових рядів показників мікроклімату.

Практична реалізація включає створення програмно-апаратного комплексу. На нижньому рівні розроблено алгоритм роботи мікроконтролера, який безперервно опитує сенсори температури, вологості, тиску та пилу й передає дані через протокол MQTT. Серверна маршрутизація реалізована в середовищі Node-RED, яке виконує функції агрегації, обробки й трансляції телеметрії. Для користувачів створені два способи взаємодії: адаптивний веб-інтерфейс на базі Next.js, що показує графіки та індикатори у режимі реального часу через

WebSocket, та автономний Telegram-бот для швидкого доступу до кліматичних даних за запитом.

На завершальному етапі виконано тестування системи, включно з перевіркою роботи мікроконтролера, логіки маршрутизації й стрес-тестуванням веб-інтерфейсу. Результати підтвердили безпомилкову роботу алгоритмів і стабільність передачі даних. Вимоги до апаратного й програмного забезпечення та інсталяційний пакет свідчать про готовність системи до розгортання й експлуатації в комерційних або академічних цілях.

Отже, усі завдання кваліфікаційної роботи виконано, а створене програмне забезпечення відповідає сучасним стандартам якості, швидкодії й безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Буч Г., Рамбо Д., Джекобсон І. UML. Посібник користувача. 2-ге вид. Москва: ДМК Пресс, 2020. 496 с.
2. Максимов М. В. Інтернет речей: апаратні та програмні платформи: навчальний посібник. Одеса: Наука і техніка, 2021. 245 с.
3. Соммервілл І. Інженерія програмного забезпечення. 10-те вид. Київ: Форс, 2019. 816 с.
4. Фаулер М. Архітектура корпоративних програмних додатків. Київ: Видавництво, 2020. 544 с.
5. Clerk Authentication Documentation. Офіційний сайт Clerk. URL: (дата звернення: 18.05.2026).
6. Convex – The backend application platform. Офіційний сайт Convex. URL: https://docs.convex.dev/ (дата звернення: 16.05.2026).
7. ESP32 Technical Reference Manual. Espressif Systems. URL: https://www.espressif.com/en/support/documents/technical-documents (дата звернення: 05.05.2026).
8. Liveblocks – Collaborative infrastructure. Офіційний сайт Liveblocks. URL: https://liveblocks.io/docs (дата звернення: 21.05.2026).
9. MQTT Version 5.0. OASIS Standard. URL: https://mqtt.org/mqtt-specification/ (дата звернення: 10.05.2026).
10. Next.js by Vercel – The React Framework. Офіційний сайт Next.js. URL: https://nextjs.org/docs (дата звернення: 15.05.2026).
11. Node-RED Documentation. Вузли та маршрутизація даних в IoT. Офіційний сайт Node-RED. URL: https://nodered.org/docs/ (дата звернення: 12.05.2026).
12. Perry L. Internet of Things for Architects. Packt Publishing, 2018. 326 p.
13. Tailwind CSS – Rapidly build modern websites. Офіційна документація. URL: https://tailwindcss.com/docs (дата звернення: 22.05.2026).
14. Telegram Bot API. Telegram Core. URL: https://core.telegram.org/bots/api (дата звернення: 20.05.2026).

ДОДАТКИ

ДОДАТОК А

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>

const char* ssid = "YOUR_WIFI_SSID";
const char* password = "YOUR_WIFI_PASSWORD";
const char* mqtt_server = "192.168.1.100";

WiFiClient espClient;
PubSubClient client(espClient);

#define DHTPIN 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

void setup_wifi() {
delay(10);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
}
}

void reconnect() {
while (!client.connected()) {
```

```
if (client.connect("ESP32_Climate_Client")) {
client.subscribe("iotfront/commands");
} else {
delay(5000);
}
}
}

void setup() {
Serial.begin(115200);
setup_wifi();
client.setServer(mqtt_server, 1883);
dht.begin();
}

void loop() {
if (!client.connected()) {
reconnect();
}
client.loop();

float t = dht.readTemperature();
float h = dht.readHumidity();
float p = 1012.5;
float d = 7.1;

if (!isnan(t) && !isnan(h)) {
String payload = "{";
payload += ""temperature": " + String(t) + ", ";
```

```
payload += ""humidity": " + String(h) + ", ";
payload += ""pressure": " + String(p) + ", ";
payload += ""dust": " + String(d);
payload += "}";

client.publish("iotfront/sensors", payload.c_str());
}

delay(15000);
}
```

ДОДАТОК Б

```
import { defineSchema, defineTable } from "convex/server";
import { v } from "convex/values";

export default defineSchema({
users: defineTable({
name: v.string(),
telegramId: v.optional(v.string()),
role: v.string(),
clerkId: v.string(),
email: v.string(),
}).index("by_clerkId", ["clerkId"]),

rooms: defineTable({
name: v.string(),
description: v.string(),
userId: v.id("users"),
}),

sensors: defineTable({
type: v.string(),
description: v.string(),
unit: v.string(),
currentValue: v.number(),
updatedAt: v.number(),
roomId: v.id("rooms"),
}).index("by_room", ["roomId"]),
```

```
measurements: defineTable({
sensorId: v.id("sensors"),
dateTime: v.number(),
value: v.number(),
status: v.string(),
}).index("by_sensor_and_time", ["sensorId", "dateTime"]),

notifications: defineTable({
userId: v.id("users"),
type: v.string(),
alertText: v.string(),
dateTime: v.number(),
isRead: v.boolean(),
}).index("by_user", ["userId"]),
});
```

ДОДАТОК В

```
import { useQuery } from "convex/react";
import { api } from "../convex/_generated/api";
import { LineChart, DonutChart } from "@/components/Charts";

export default function ClimateDashboard({ roomId }) {
const sensors = useQuery(api.sensors.getByRoom, { roomId });
const measurements = useQuery(api.measurements.getRecent, { limit: 50
});

if (!sensors || !measurements) {
return Завантаження даних системи...;
}

return (

Клімат у кімнаті

 <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-4
mb-8">
   {sensors.map((sensor) => (
    <div key={sensor._id} className="bg-white p-4 rounded-xl shadow-
sm">
      <h3 className="text-gray-500 text-sm font-
medium">{sensor.type}</h3>
      <div className="mt-2 flex items-baseline gap-2">
        <span className="text-3xl font-semibold text-gray-900">
          {sensor.currentValue}
```

```
          </span>
          <span className="text-gray-500">{sensor.unit}</span>
        </div>
        <p className="text-xs text-gray-400 mt-2">
          Оновлено: {new Date(sensor.updatedAt).toLocaleTimeString()}
        </p>
      </div>
    ))}
  </div>

  <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
    <div className="bg-white p-6 rounded-xl shadow-sm">
      <h2 className="text-lg font-semibold mb-4">Динаміка температури</h2>
      <LineChart data={measurements} dataKey="temperature" color="#ef4444" />
    </div>
    <div className="bg-white p-6 rounded-xl shadow-sm">
      <h2 className="text-lg font-semibold mb-4">Рівень пилу</h2>
      <DonutChart data={sensors.filter(s => s.type === 'Dust')} />
    </div>
  </div>
</div>
);
}
```

ДОДАТОК Г


```
[
{
"id": "e2c3b5a1.8f4d9",
"type": "mqtt in",
"z": "climate_monitoring_flow",
"name": "iotfront/sensors",
"topic": "iotfront/sensors",
"qos": "0",
"datatype": "json",
"broker": "b3a2c1d0.7e6f8",
"nl": false,
"rap": true,
"x": 160,
"y": 120,
"wires": [
[
"f1e2d3c4.9b8a7",
"d4c3b2a1.0e9f8"
]
]
},
{
"id": "f1e2d3c4.9b8a7",
"type": "debug",
"z": "climate_monitoring_flow",
"name": "Панель Debug",
"active": true,
"tosidebar": true,
65
"console": false,
"tostatus": false,
"complete": "payload",
"targetType": "msg",
"x": 420,
"y": 80,
```

"wires": []
},
{
"id": "d4c3b2a1.0e9f8",
"type": "function",
"z": "climate_monitoring_flow",
"name": "Форматування Convex",
"func": "msg.payload = {\n sensorType: "ESP32_Complex",\n
temperature: msg. payload.temperature,\n humidity:
msg. payload.humidity,\n timestamp: Date.now()\n};\nreturn msg;",
"outputs": 1,
"noerr": 0,
"initialize": "",
"finalize": "",
"x": 450,
"y": 160,
"wires": [
[
"a1b2c3d4.5e6f7"
]
]
}
]

ДОДАТОК Д

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

Я, Колесніков Михайло Андрійович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему «Програмне забезпечення моніторингу клімату в закритому приміщенні» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з
чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що:
інструменти ШІ Gemini були використані для:
▪ перекладу іншомовних джерел;
▪ допомоги у написанні/відлагодженні програмного коду;

Я розумію, що надання недостовірної інформації може призвести до скасування
результатів захисту та відрахування з числа здобувачів НУБіП України.

«___»_____ 2026 р. _____ **Михайло КОЛЕСНІКОВ**
(підпис)