

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ
МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМ. С.З. ГЖИЦЬКОГО
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему:

**«РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ УПРАВЛІННЯ РОЗУМНИМ
БУДИНКОМ»**

Виконав: здобувач групи Кн-41
спеціальності 122 «Комп'ютерні науки»

Пустовіт М. Р.
(прізвище та ініціали)

Керівник: _____ Пташник В. В.
(прізвище та ініціали)

ДУБЛЯНИ-2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Перший (бакалаврський) рівень вищої освіти
Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри

(підпис)

д.т.н., професор, Тригуба А. М.

(вч. звання, прізвище, ініціали)

“ ” _____ 202 року

З А В Д А Н Н Я НА КВАЛІФІКАЦІЙНУ РОБОТУ

Пустовіту Максиму Романовичу

(прізвище, ім'я, по батькові)

1. Тема роботи «Розробка веб-додатку для управління розумним будинком»

керівник роботи к. т. н., доцент, Пташник В. В.

(наук.ступінь, вч. звання, прізвище, ініціали)

затверджені наказом Львівського НУП від 25.02.2025 року № 123/к-с.

2. Строк подання студентом роботи 10 червня 2025 року

3. Вихідні дані до роботи: характеристика систем автоматизованого моніторингу параметрів мікроклімату в приміщеннях; технічна документація до сенсорних та виконавчих IoT-пристроїв (ESP32, DHT22, MQ-135, SDS011 тощо); технічні паспорти хмарних платформ і сервісів для передачі даних (Firebase, MQTT-брокери); науково-технічна та довідкова література з архітектури веб-додатків, принципів побудови інформаційних систем та захисту даних; технічні вимоги до систем віддаленого контролю якості повітря та автоматизованого керування мікрокліматичним обладнанням у побутових умовах..

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Вступ

1. Теоретичні основи розумного будинку у структурі розумного міста

2. Етапи проектування системи

3. Реалізація проектного рішення

4. Охорона праці

Висновки

Список використаних джерел

5. Перелік графічного матеріалу

Графічний матеріал подається у вигляді презентації

6. Консультанти розділів

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата		Відмітка про виконання
		завдання видав	завдання прийняв	
1, 2, 3	Пташник В. В., к.т.н., доцент			
4	Городецький І. М., к.т.н., доцент			

7. Дата видачі завдання 28 листопада 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Відмітка про виконання
1	Складання інженерної характеристики об'єкту проектування	28.11.2024 – 31.12.2024	
2	Вибір принципів, методів та засобів розробки веб-додатку для управління розумним будинком	01.01.2025 – 28.02.2025	
3	Проектування веб-додатку для управління розумним будинком	01.03.2025 – 30.04.2025	
4	Розгляд питань з охорони праці та безпеки у надзвичайних ситуаціях	01.05.2025 – 14.05.2025	
5	Завершення оформлення розрахунково-пояснювальної записки та презентаційного матеріалу	15.05.2025 – 31.05.2025	
6	Завершення роботи в цілому. Підготовка до захисту кваліфікаційної роботи	01.06.2025 – 10.06.2025	

Здобувач

(підпис) Пустовіт М.Р.
(прізвище та ініціали)

Керівник роботи

(підпис) Пташник В. В.
(прізвище та ініціали)

Розробка веб-додатку для управління розумним будинком.
Пустовіт М. Р. Кафедра інформаційних технологій – Дубляни,
Львівський НУВМБ ім. С.З. Гжицького, 2025.

Кваліфікаційна робота: 55 сторінок текстової частини, 35 рисунків, 3
таблиці, 21 джерело літератури.

Мета кваліфікаційної роботи полягає у створенні веб-додатку для управління розумним будинком, що забезпечує інтерактивний контроль параметрів внутрішнього середовища та підвищує ефективність керування побутовими системами за допомогою технологій Інтернету речей.

Об'єктом дослідження є інформаційна система керування параметрами середовища в інтелектуалізованих житлових просторах, яка забезпечує інтеграцію сенсорних пристроїв, виконавчих механізмів та користувацького веб-інтерфейсу.

Предмет дослідження охоплює принципи побудови веб-застосунків, методи візуалізації сенсорних даних, моделі доступу користувачів та протоколи обміну даними в системах управління розумним середовищем.

У кваліфікаційній роботі здійснено аналіз сучасних архітектур веб-додатків для IoT-систем, розроблено трирівневу модель з використанням React, Node.js і MySQL, реалізовано механізми реєстрації, автентифікації, ролей користувачів та аналітики сенсорних даних у реальному часі. Запропоновану систему протестовано на узагальненій структурі Smart Home, що демонструє її масштабованість, функціональність та готовність до впровадження в побутовому або комерційному середовищі.

Ключові слова: веб-додаток, розумний будинок, IoT, сенсори, керування середовищем.

ЗМІСТ

ВСТУП.....	6
РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ РОЗУМНОГО БУДИНКУ У СТРУКТУРІ РОЗУМНОГО МІСТА.....	7
1.1 Структура та принципи функціонування інформаційних систем	7
1.2 Архітектура веб-додатка для керування розумним середовищем	12
1.3 Виклики, перспективи та значення впровадження Smart-рішень.....	15
РОЗДІЛ 2 ЕТАПИ ПРОЕКТУВАННЯ СИСТЕМИ.....	18
2.1 Формування архітектурної структури веб-інтерфейсу	18
2.2 Моделювання бази даних для веб-інтерфейсу.....	21
2.3 Побудова діаграм прецедентів для визначення ролей користувачів.....	23
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОЕКТНОГО РІШЕННЯ	29
3.1 Програмна реалізація веб-інтерфейсу	29
3.2 Створення та наповнення бази даних.....	35
3.3 Впровадження механізмів реєстрації та автентифікації користувачів.....	41
РОЗДІЛ 4 ОХОРОНА ПРАЦІ.....	46
4.1 Вимоги безпеки праці під час експлуатації систем вентиляції, опалення та кондиціонування повітря	46
4.2 Розробка захисту від пожеж та вибухів в системах опалення, вентиляції, освітлення та кондиціонування повітря.....	49
ВИСНОВКИ	53
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	54

ВСТУП

У контексті стрімкої цифровізації та урбанізації значно зростає потреба в автоматизованих рішеннях для забезпечення комфортного, безпечного та екологічного середовища в житлових і комерційних просторах. Концепція «розумного будинку» є важливим складником більш широкої ідеї «розумного міста» та передбачає використання сучасних інформаційно-комунікаційних технологій для моніторингу та керування параметрами внутрішнього середовища. Зокрема, якість повітря, рівень вологості, температура та наявність шкідливих домішок є критично важливими для здоров'я й добробуту мешканців. У зв'язку з цим розробка ефективних веб-додатків для моніторингу та управління такими параметрами набуває особливої актуальності.

Метою кваліфікаційної роботи є створення інноваційного веб-додатку для управління розумним будинком, що забезпечує зручну взаємодію користувача із системою моніторингу внутрішнього середовища, зокрема якості повітря, на основі технологій Інтернету речей (IoT) та сучасних веб-фреймворків.

Для досягнення поставленої мети було визначено такі завдання:

- проаналізувати поточний стан інформаційних систем «розумного міста» та виділити специфіку роботи в розумному будинку;
- розробити архітектурну структуру веб-інтерфейсу на основі технологій React, Node.js та MySQL;
- спроектувати й реалізувати базу даних для зберігання сенсорних та користувацьких даних;
- побудувати діаграми прецедентів для визначення користувацьких ролей і прав доступу;
- реалізувати функціонал реєстрації, автентифікації та керування системними параметрами відповідно до призначеної ролі;
- забезпечити візуалізацію й аналіз даних з датчиків у реальному часі;
- оцінити ефективність розробленого рішення та його можливості для подальшого впровадження.

РОЗДІЛ 1

ТЕОРЕТИЧНІ ОСНОВИ РОЗУМНОГО БУДИНКУ У СТРУКТУРІ РОЗУМНОГО МІСТА

1.1 Структура та принципи функціонування інформаційних систем

У ХХІ столітті світ зіткнувся з глобальними викликами, зокрема зростанням урбанізації, перенасиченістю мегаполісів, забрудненням навколишнього середовища та нестачею енергетичних ресурсів. У цьому контексті зростає потреба у впровадженні нових технологічних рішень, що дозволяють забезпечити стале функціонування міської інфраструктури та підвищити якість життя мешканців. Однією з таких інноваційних концепцій є «розумне місто» (Smart City), що передбачає інтеграцію сучасних інформаційно-комунікаційних технологій у всі аспекти функціонування урбаністичного середовища.

Концепція розумного міста передбачає формування єдиної інтелектуальної екосистеми, яка охоплює різноманітні підсистеми – транспорт, енергетику, водопостачання, безпеку, екологію, управління будівлями.

Основою цієї екосистеми є інформаційні системи, побудовані на базі сенсорних мереж, модулів збору та обробки даних, хмарних сервісів, алгоритмів штучного інтелекту та зручних засобів візуалізації інформації (табл. 1.1).

Ключову роль у такій системі відіграє збір достовірної та актуальної інформації про стан середовища. Для цього використовуються численні сенсори, які розміщуються в різних точках міста, будівлях та об'єктах інфраструктури (таблиця 1.2).

Таблиця 1.1 – Компоненти інфраструктури «розумного міста»

Компонент	Опис функціональності	Приклади реалізації
Розумний транспорт	Управління дорожнім рухом, моніторинг громадського транспорту	Світлофори з адаптивним регулюванням, GPS-розклад
Енергетика	Енергоефективність, «розумні» лічильники, ВДЕ	Smart Grid, сонячні панелі, смарт-мережі
Водопостачання	Контроль витрат, виявлення витоків, автоматичне регулювання	Сенсори потоку, автоматичні клапани
Безпека	Відеоспостереження, системи сповіщення про НС	ІР-камери, тривожні кнопки, аналіз поведінки
Екологічний моніторинг	Вимірювання рівня CO ₂ , шуму, забруднення повітря	Станції моніторингу повітря, IoT-сенсори
Управління будівлями	Автоматизація мікроклімату, освітлення, доступу	Системи BMS, «розумні будинки»
Здоров'я	Телемедицина, контроль стану пацієнтів	IoT-пристрої для моніторингу пульсу/тиску

Сенсори можуть фіксувати параметри температури, вологості, рівня шуму, концентрації шкідливих речовин у повітрі, руху транспортних засобів або людей тощо. Отримані з них дані передаються до центральної платформи обробки за допомогою різних комунікаційних технологій – Wi-Fi, Zigbee, LoRaWAN, NB-IoT або дротових інтерфейсів.

Таблиця 1.2 – Основні типи сенсорів для моніторингу параметрів розумного середовища

Тип сенсора	Вимірюваний параметр	Приклад моделі	Інтерфейс підключення	Особливості використання
Температурний	Температура повітря	DHT22, DS18B20	1-Wire, цифровий	Використовується для HVAC-систем
Газовий	CO ₂ , CO, метан	MQ-135, MQ-2	Аналоговий, цифровий	Моніторинг повітря, дим, витоки газу
Вологість	Відносна вологість	DHT22, SHT30	Цифровий (I ² C)	Підтримка мікроклімату
Світловий	Освітленість (люкси)	BH1750, LDR	Аналоговий, I ² C	Керування освітленням, жалюзі
Частинок пилу	PM2.5, PM10	SDS011, GP2Y1010AU0F	UART, аналоговий	Оцінка якості повітря
Руху	Наявність об'єкта	HC-SR501	Цифровий	Автоматичне вмикання освітлення

На наступному етапі інформація, отримана із сенсорів, обробляється за допомогою серверних потужностей або хмарних обчислень. Центральна обчислювальна платформа системи виконує первинну фільтрацію, нормалізацію, аналіз та зберігання даних у базах. Для обробки великих масивів інформації застосовуються методи аналітики даних (data mining), інструменти статистичного аналізу, алгоритми машинного навчання та прогнозування. Результати цього аналізу слугують підґрунтям для прийняття рішень – як автоматичних, так і керованих користувачем.

Зібрані сенсорами дані передаються до вузлів обробки за допомогою комунікаційних модулів. Залежно від специфіки задачі застосовуються як бездротові протоколи (Wi-Fi, Bluetooth, Zigbee, LoRa), так і дротові рішення (Ethernet, RS-485), їх опис наведено у табл. 1.3. Зокрема, для середовищ із низьким енергоспоживанням або великою кількістю вузлів доцільно використовувати LPWAN-технології, наприклад LoRaWAN або NB-IoT. Комунікаційний рівень відповідає не лише за передачу інформації, а й за її тимчасове буферизування, шифрування та маршрутизацію.

Таблиця 1.3 – Характеристики мережевих технологій для IoT-систем розумного середовища

Технологія	Тип з'єднання	Дальність дії	Швидкість передачі	Споживання енергії	Приклади застосування
Wi-Fi	Бездротове	до 100 м	до 100 Мбіт/с	Високе	Передача великих обсягів даних у мережі
LoRa	Бездротове (LPWAN)	до 10 км	до 50 кбіт/с	Низьке	Розподілені сенсорні мережі
Zigbee	Бездротове	до 100 м	до 250 кбіт/с	Дуже низьке	Автоматизація будинку, датчики
Ethernet	Дротове	до 100 м	до 1 Гбіт/с	Високе	Надійне з'єднання з локальним сервером
Bluetooth LE	Бездротове	до 50 м	до 2 Мбіт/с	Низьке	Мобільні пристрої, короткий зв'язок
NB-IoT	Стільникове	до 15 км	до 250 кбіт/с	Низьке	Віддалений моніторинг, міська інфраструктура

Одним із найбільш важливих елементів таких систем є веб-інтерфейс, який забезпечує взаємодію користувача з інформаційною системою. Через веб-інтерфейс користувач отримує доступ до візуалізованих даних у реальному часі, має змогу керувати підсистемами, задавати сценарії роботи пристроїв або переглядати історію змін параметрів середовища. У розумному місті веб-інтерфейси використовуються не лише адміністраторами чи фахівцями, а й пересічними громадянами для інформування, участі в міському управлінні або прийнятті індивідуальних рішень.

Інформаційна система розумного середовища зазвичай складається з кількох функціональних рівнів: рівня збору даних, комунікаційного рівня, рівня обробки інформації та рівня взаємодії з користувачем. На першому рівні працюють численні сенсори, розміщені у приміщеннях, на вулицях або в обладнанні. Їхнім завданням є безперервний моніторинг фізичних параметрів, наприклад температури, вологості, концентрації вуглекислого газу, рівня освітлення чи наявності руху. Компоненти інформаційної системи розумного міста показано на рисунку 1.1.



Рисунок 1.1 – Компоненти інформаційної системи розумного міста

Далі дані надходять до обчислювального рівня – це може бути локальний мікрокомп'ютер (наприклад, Raspberry Pi) або сервер у хмарі. Тут інформація структурується, записується до бази даних, і при потребі проходить через алгоритми аналітики. Сучасні системи часто використовують NoSQL-бази (MongoDB, InfluxDB), якщо дані є слабо структурованими, або традиційні реляційні СКБД (MySQL, PostgreSQL), якщо потрібно підтримувати чітку логіку зв'язків між таблицями. Обробка даних включає як просту статистику (наприклад, обчислення середніх значень, виявлення пікових навантажень), так і складніший аналіз із використанням моделей машинного навчання – класифікацію станів, прогнозування, виявлення аномалій. Узагальнену архітектуру системи збору та обробки даних у розумному середовищі наведено на рисунку 1.2.

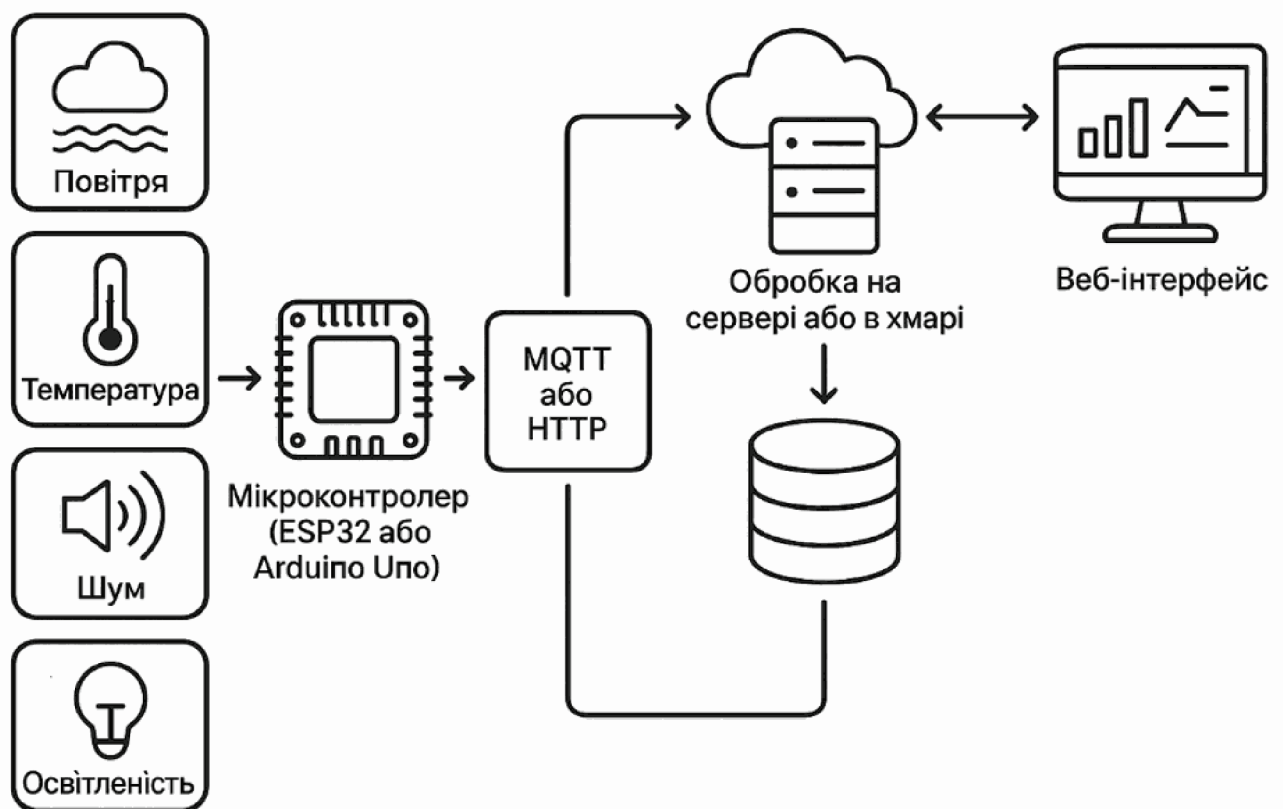


Рисунок 1.2 – Архітектура системи збору та обробки даних у розумному середовищі

1.2 Архітектура веб-додатка для керування розумним середовищем

На рівні взаємодії з користувачем ключовим є веб-інтерфейс. Це може бути адаптивний веб-додаток, створений за допомогою сучасних технологій, таких як HTML5, CSS3, JavaScript-фреймворки (React, Vue, Angular). Його завдання – надати користувачеві зручний доступ до інформації про стан системи, дозволити візуалізувати дані у вигляді графіків, таблиць або інтерактивних панелей, а також керувати параметрами роботи пристроїв. Наприклад, користувач може переглянути рівень CO₂ у приміщенні, отримати повідомлення про перевищення допустимого порогу та увімкнути вентиляцію.

У системах типу «розумний дім», що є складовою розумного міста, взаємодія відбувається через контролери – наприклад, ESP32, Arduino Uno або Raspberry Pi. Ці пристрої слугують шлюзом між сенсорами, виконавчими пристроями (реле, двигунами, вентиляторами) та веб-додатком. Вони забезпечують первинну обробку даних, виконання запрограмованих сценаріїв (наприклад, автоматичне включення освітлення при зниженні рівня освітленості), а також передають інформацію на сервер або хмарну платформу.

Для комунікації між пристроями та веб-додатком широко застосовується протокол MQTT – легкий брокерно-орієнтований протокол обміну повідомленнями, ідеальний для IoT-середовища. Він дозволяє організовувати обмін даними у форматі «публікація-підписка» (publish-subscribe), що значно спрощує масштабування системи. Альтернативно можуть використовуватися HTTP-запити або WebSocket-підключення, що забезпечують двосторонній обмін інформацією в реальному часі між клієнтом і сервером.

Архітектура веб-додатка, що використовується для керування інформаційною системою розумного середовища, зазвичай має трирівневу структуру: клієнтський рівень (frontend), серверний рівень (backend) і рівень бази даних. На клієнтському рівні реалізується інтерфейс користувача, який відповідає за візуалізацію даних, зручність навігації, адаптивність до різних типів пристроїв (комп'ютер, планшет, смартфон). У цьому шарі можуть

застосовуватися такі фреймворки, як React.js або Vue.js, що забезпечують створення інтерактивних односторінкових застосунків (SPA) із високою продуктивністю. Типова структура веб-додатка показана на рисунку 1.3.

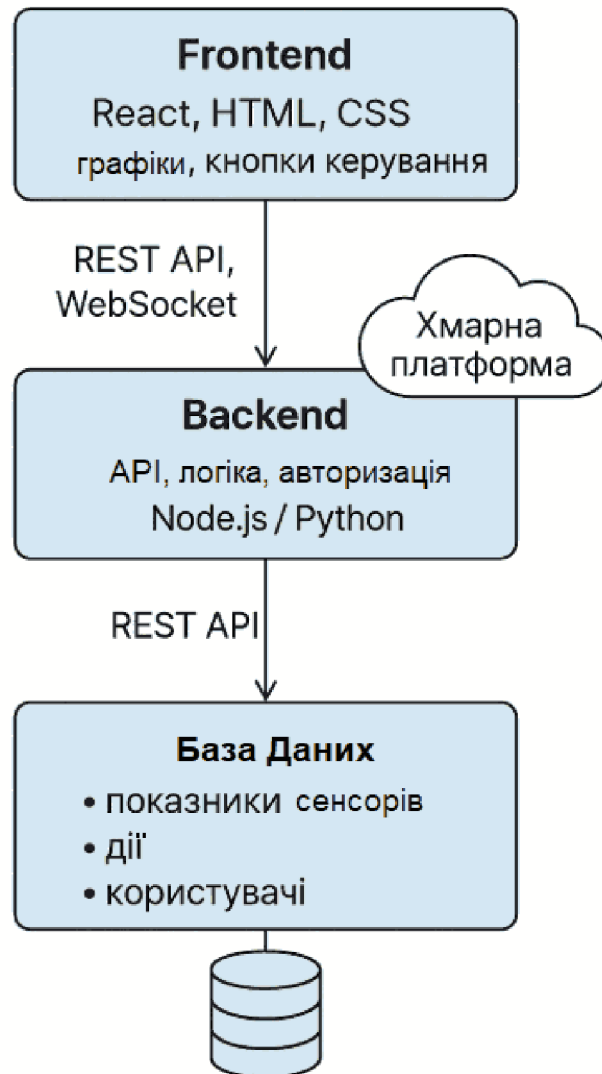


Рисунок 1.3 – Структура веб-додатка для керування системою розумного середовища

На серверному рівні (backend) обробляються запити від користувача, здійснюється доступ до бази даних, реалізуються логіка керування пристроями та алгоритми авторизації. У цьому шарі доцільно використовувати такі середовища, як Node.js (із фреймворком Express), Python (Flask, Django) або PHP, залежно від вимог до швидкодії, розширюваності та простоти розробки. Для забезпечення безпечного доступу до системи впроваджуються механізми

автентифікації (JWT, OAuth2), розмежування прав доступу та шифрування з'єднання (HTTPS).

API (інтерфейс прикладного програмування) слугує посередником між frontend і backend, дозволяючи отримувати або передавати дані у форматі JSON або XML. Наприклад, при зверненні до API можна отримати список доступних пристроїв, значення сенсорів за останні 24 години або змінити параметри роботи конкретного пристрою. Така архітектура дозволяє масштабувати систему, підключати нові модулі та створювати мобільні застосунки, що користуються тим самим API.

Інформаційні системи цього типу повинні бути не лише функціональними, а й інтуїтивно зрозумілими для кінцевого користувача. UX/UI-дизайн веб-інтерфейсу відіграє важливу роль у забезпеченні зручності взаємодії з системою, зменшенні кількості помилок, підвищенні швидкості прийняття рішень. При проєктуванні інтерфейсу необхідно враховувати типи користувачів: оператори, адміністратори, мешканці, технічний персонал. Наприклад, для мешканця достатньо простого дашборду з основними показниками та кнопками для активації пристроїв (рис. 1.4), тоді як адміністратор має доступ до логів, налаштувань сценаріїв та моніторингу збоїв.

Окрім візуального представлення даних, веб-інтерфейси можуть виконувати функцію активного реагування на події. Система може генерувати повідомлення про перевищення порогових значень (температура, вологість, рівень CO₂) та надсилати їх у вигляді рор-уп-сповіщень або повідомлень на електронну пошту, Telegram чи інші сервіси. Така функціональність реалізується за допомогою подієво-орієнтованої логіки, таймерів або асинхронних тригерів, що працюють у фоновому режимі на сервері.

Сучасні веб-інтерфейси також інтегруються з картографічними сервісами (наприклад, Google Maps, Leaflet) для просторової візуалізації сенсорних точок, що особливо актуально для міських мереж. Користувач може переглядати значення параметрів безпосередньо на карті, обирати окремі райони, застосовувати фільтри за типами пристроїв або за періодами часу. Така інтеграція значно підвищує наочність і зручність керування системою.

1.3 Виклики, перспективи та значення впровадження Smart-рішень

Впровадження інформаційних систем у розумному місті передбачає не лише технічне оснащення та розробку програмного забезпечення, але й вирішення ряду організаційних, правових та етичних питань. Серед основних переваг таких систем варто виділити підвищення прозорості в управлінні ресурсами, зменшення витрат на енергію, оперативне реагування на надзвичайні ситуації та покращення екологічної ситуації завдяки постійному моніторингу якості повітря, рівня шуму та інших факторів.

Наприклад, у системах автоматизованого керування мікрокліматом приміщень або вентиляцією, аналіз даних із сенсорів дозволяє не лише підтримувати комфортні умови, а й зменшити споживання електроенергії завдяки інтелектуальному увімкненню та вимкненню обладнання. Аналогічно, у системах моніторингу повітря зовнішнього середовища такі рішення дозволяють своєчасно попереджати населення про небезпечні концентрації шкідливих речовин.

Проте розробка та масштабування таких систем супроводжується низкою викликів. Однією з ключових проблем є забезпечення інформаційної безпеки. В умовах, коли система обробляє персональні дані користувачів, керує обладнанням і має доступ до інфраструктури, вкрай важливо впроваджувати захищені протоколи зв'язку, системи аутентифікації та контроль прав доступу (рис. 1.4). Зокрема, необхідно враховувати потенційні ризики витоку даних, атак типу «man-in-the-middle», підміни команд тощо.

Іншою важливою проблемою є масштабованість. Система, яка спочатку працює на рівні одного будинку або району, з часом може розширюватися до рівня міста. Це потребує відповідного проєктування архітектури – як програмної (розділення сервісів, мікросервісна структура), так і апаратної (використання потужніших серверів, хмарних обчислень, балансування навантаження). Використання хмарних платформ, таких як AWS IoT, Microsoft

Azure IoT Hub, Google Cloud IoT, дозволяє забезпечити гнучкість і стійкість до збоїв при розширенні системи.

Окремо слід згадати про інтероперабельність – здатність системи інтегруватися з іншими службами, сервісами або платформами. У сучасному місті не існує єдиної системи, яка б охоплювала всі аспекти життя: окремо функціонують системи відеоспостереження, енергоменеджменту, громадського транспорту, охорони здоров'я. Тому необхідно забезпечити можливість взаємодії між ними через стандартизовані API, обмін повідомленнями або спільну базу даних. Це дозволить створити єдину інформаційну екосистему міста, в якій різні служби доповнюють одна одну.

З технічної точки зору, важливою перевагою подібних систем є можливість реалізації сценарного управління. Наприклад, при фіксації перевищення допустимого рівня CO₂ система може автоматично відкрити вікна, увімкнути вентиляцію та надіслати повідомлення користувачу. Такі сценарії можуть бути запрограмовані користувачем через веб-інтерфейс, з використанням умов, таймерів і тригерів. Це надає системі гнучкості та дозволяє враховувати індивідуальні потреби кожного користувача.

Таким чином, побудова інформаційних систем для розумного середовища є багаторівневим і міждисциплінарним процесом, який об'єднує апаратні рішення (сенсори, контролери, мережеве обладнання), програмну інфраструктуру (веб-інтерфейси, бази даних, хмарні платформи) та інтелектуальні алгоритми аналізу даних. Кожен із компонентів відіграє критичну роль у забезпеченні безперервного функціонування системи, її стійкості до збоїв та здатності адаптуватися до змін середовища.

Одним із ключових факторів успішного впровадження таких рішень є орієнтація на користувача. Веб-інтерфейси повинні бути не лише інформативними, а й зручними, інтуїтивними, з адаптацією до різних сценаріїв використання. Крім того, система має підтримувати багаторівневий доступ, що дозволяє різним категоріям користувачів (мешканцям, технічному персоналу, адміністраторам) взаємодіяти з нею в межах визначених повноважень.

Розширення подібних систем на рівень міської інфраструктури відкриває можливості для створення цілісної екосистеми розумного міста. Такі екосистеми включають не лише побутові аспекти (комфорт, енергозбереження), а й глобальні цілі – зниження шкідливих викидів, формування політик сталого розвитку, підвищення прозорості роботи органів місцевого самоврядування. У результаті мешканці отримують покращену якість життя, а міська влада – ефективні інструменти управління. Загальна схема архітектури інформаційної системи розумного середовища наведена на рисунку 1.4.



Рисунок 1.4 – Загальна архітектура інформаційної системи розумного середовища

РОЗДІЛ 2

ЕТАПИ ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Формування архітектурної структури веб-інтерфейсу

Існуючі системи моніторингу внутрішнього середовища мають низку суттєвих недоліків: високу вартість, складність налаштування та інформаційну несумісність. Їх функціонал часто обмежується загальними параметрами середовища без детального аналізу якості повітря, а результати вимірювань рідко проходять верифікацію [12]. Крім того, більшість IoT-систем розумного будинку орієнтовані на безпекові сповіщення, а не повноцінний моніторинг повітря [10]. Питання калібрування сенсорів, управління даними та простоти впровадження залишаються малодослідженими [11].

У цій роботі запропоновано систему моніторингу якості повітря для розумного будинку, що усуває вказані проблеми. Її ключовою особливістю є управління виключно уповноваженим персоналом з чітко визначеними ролями доступу. Через веб-інтерфейс користувачі отримують інформацію про рівні забруднення відповідно до своїх повноважень.

Система побудована на базі технології Інтернету речей (IoT), що дає змогу ідентифікувати основні забруднювачі повітря (чадний газ, CO₂, SO₂ тощо), забезпечує віддалений доступ до даних і зберігання інформації у базі для подальшої обробки. Візуалізація даних реалізована у вигляді наочних діаграм.

На стороні клієнта веб-інтерфейс реалізовано з використанням React і Redux. React – популярна бібліотека для створення динамічних та інтерактивних інтерфейсів, де кожен компонент відповідає за відображення окремої частини й може оновлюватися в режимі реального часу. Redux застосовується для централізованого керування станом, що забезпечує узгодженість даних між компонентами – особливо важливу, коли зміни в одній частині впливають на інші.

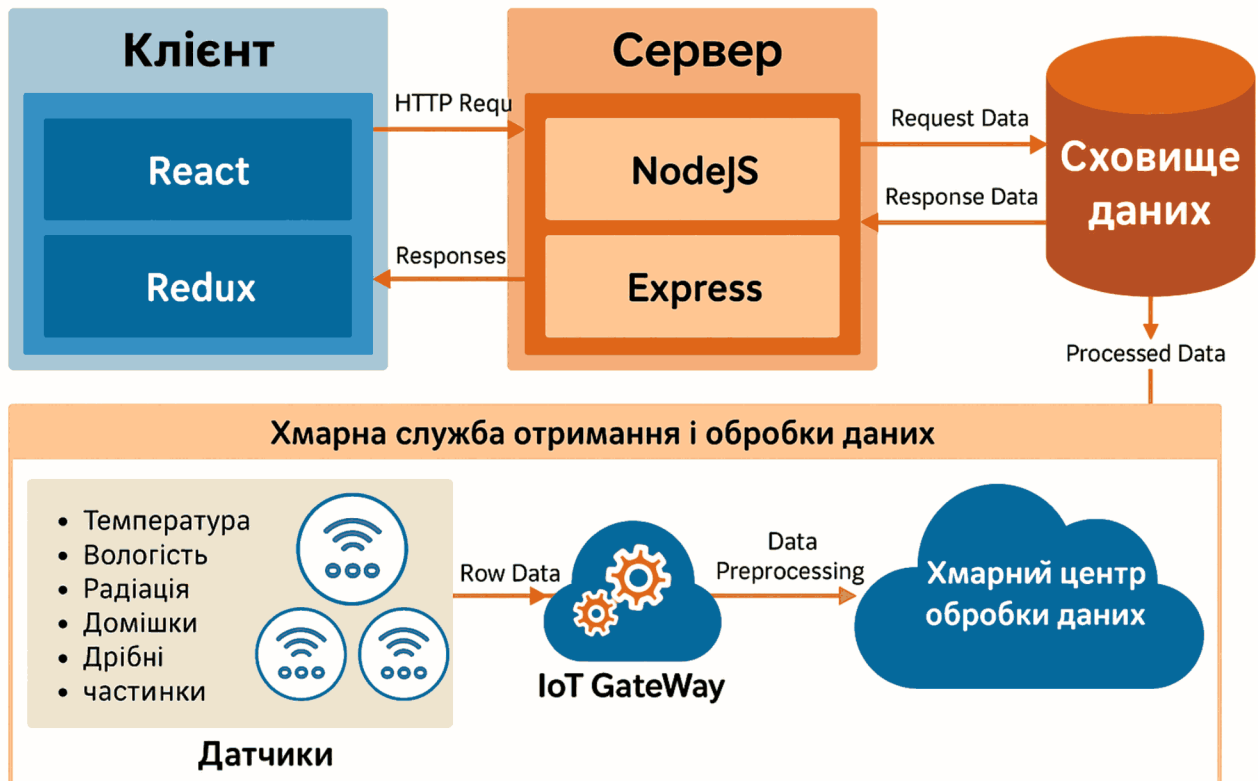


Рисунок 2.1 – Архітектурна схема веб-інтерфейсу для системи моніторингу мікроклімату приміщень

Клієнтська частина обмінюється даними із сервером через HTTP-запити. Наприклад, при перегляді інформації про якість повітря або зміні параметрів пристрою надсилається відповідний запит. Серверна частина реалізована на Node.js з використанням Express. Node.js забезпечує асинхронну обробку запитів, дозволяючи ефективно працювати з великою кількістю з'єднань. Express спрощує маршрутизацію й роботу з middleware. Сервер обробляє запити клієнтів і взаємодіє з базою даних для отримання або збереження інформації.

У системі використовується реляційна база даних MySQL, яка підтримує складні SQL-запити, транзакції та забезпечує надійне зберігання. Сервер формує запити для доступу до бази, а результати повертаються клієнту. Датчики підключені до IoT-шлюзу (Gateway), що виконує попередню обробку даних – фільтрацію, нормалізацію та агрегацію. Це зменшує обсяг переданої інформації й підвищує ефективність обробки. Алгоритм роботи хмарного сервісу збору й аналізу даних показано на рисунку 2.2.



Рисунок 2.2 – Блок-схема хмарного сервісу збору та обробки сенсорних даних

Після попередньої обробки дані надсилаються з IoT-шлюзу до хмарного центру обробки. Він виконує кілька важливих функцій: перевіряє дані на правильність і повноту, відкидаючи некоректні записи, а також здійснює обчислення, статистичний аналіз і виявлення аномалій. Отримані результати можуть використовуватися для звітів, сповіщень чи запуску дій.

Після аналізу оброблені дані зберігаються в базі даних, звідки серверна частина отримує необхідну інформацію. Це дозволяє користувачам переглядати актуальні та історичні дані про якість повітря, а також змінювати параметри пристроїв у реальному часі.

Інтерфейс користувача створено за допомогою React і оформлено через Tailwind CSS – утилітарний фреймворк, який спрощує стилізацію компонентів та забезпечує швидку й узгоджену розробку.

2.2 Моделювання бази даних для веб-інтерфейсу

Ця база даних створена для підтримки веб-інтерфейсу інформаційної системи «Розумного будинку» і використовується для обробки даних, зібраних сенсорами в межах моніторингу якості повітря. Вона містить інформацію про користувачів, їх ролі, розташування сенсорів, типи платформ підключення, а також показники з датчиків і їх історію. Це дозволяє контролювати якість повітря в реальному часі й аналізувати історичні дані для виявлення тенденцій і відхилень.

База складається з кількох основних таблиць, кожна з яких відповідає за окремий аспект роботи системи. Схему структури подано на рисунку 2.3. Таблиця `user` зберігає дані про користувачів: унікальний ID (`userID`), ім'я користувача, пароль, email та роль (`roleID`). Роль визначається зовнішнім ключем, що посилається на таблицю `roles`, де зберігається ідентифікатор і назва кожної ролі.

У таблиці `Location` зберігаються дані про місце розташування датчиків і пристроїв: унікальний ID (`locationID`), країна, місто й адреса. Це дозволяє точно визначити місцезнаходження обладнання.

Таблиця `Platform_Types` містить відомості про типи платформ, до яких підключаються пристрої. Вона включає `platformID`, назву типу (`typeName`), модель (`modelName`) і технічні властивості у форматі JSON.

Таблиця `IoTDevice` зберігає інформацію про IoT-пристрої: `deviceID`, тип, користувача (`userID`), підключений датчик (`sensorID`), привід (`actuatorID`), розташування (`locationID`) і платформу (`platformID`), а також статус пристрою. Усі зв'язки реалізовані через зовнішні ключі.

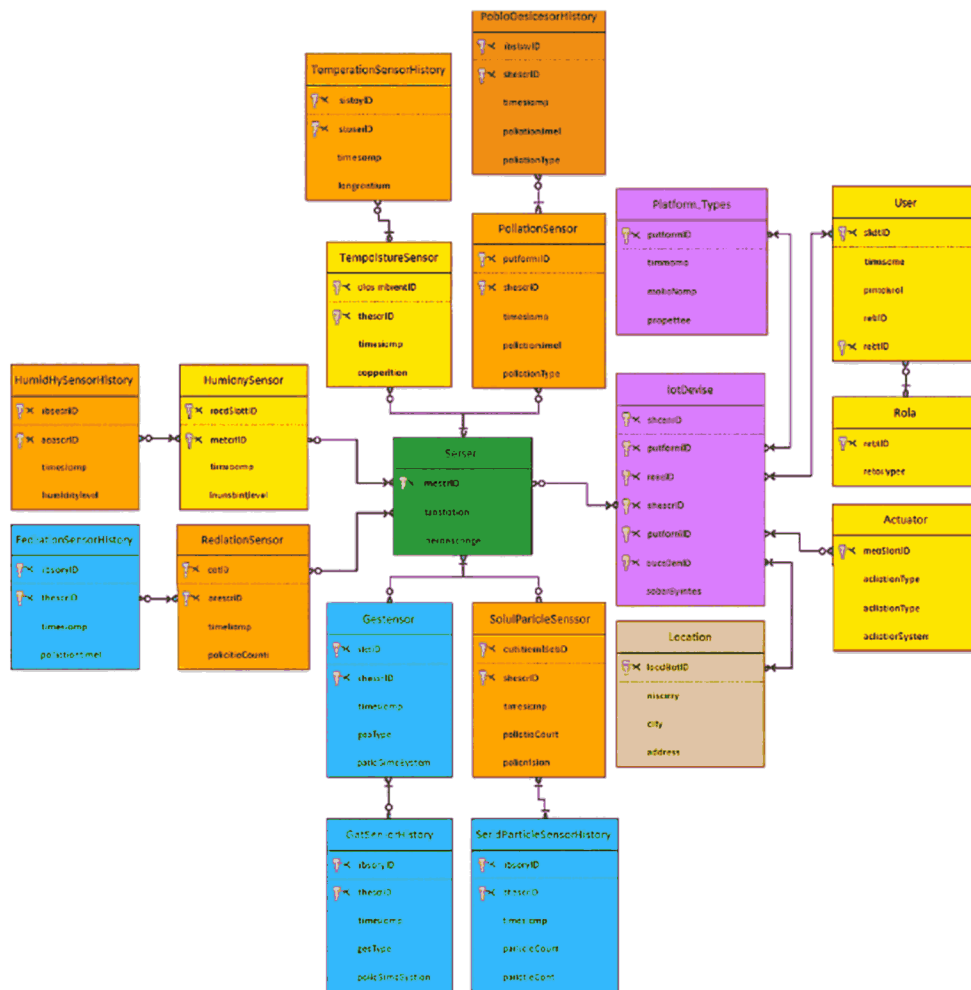


Рисунок 2.3 – Структурна модель бази даних інформаційної системи

У таблиці Actuators фіксуються дані про приводи: actuatorID, тип (actuatorType) і стан (actuator_status), що дає змогу контролювати їх доступність і працездатність.

Таблиця Sensors містить інформацію про сенсори: sensorID, тип (sensorType) і статус (sensor_status). Для кожного типу сенсора є окрема таблиця характеристик.

Окрім основних таблиць, база даних містить історичні таблиці, що зберігають вимірювання у різні моменти часу. Наприклад, Humidity_History містить historyID, sensorID, timestamp і humidityLevel. Для кожного типу сенсора існує відповідна історична таблиця, що дозволяє виконувати аналіз у часовому розрізі.

Зв'язки між сутностями реалізовані через зовнішні ключі. Наприклад, між таблицями User і Role існує зв'язок «багато-до-багатьох», як і між IoTDevice та

User, де пристрій може бути пов'язаний із кількома користувачами. Кожен пристрій також може мати кілька датчиків і один виконавчий механізм, при цьому всі компоненти зазвичай підключені до однієї платформи.

2.3 Побудова діаграм прецедентів для визначення ролей користувачів

Діаграма варіантів використання у веб-інтерфейсі моніторингу якості повітря показує функції, доступні користувачам залежно від їхніх ролей. Кожна роль має власні повноваження, що визначають доступ до функціоналу системи. Серед підтримуваних ролей: гість, користувач, адміністратор, модератор і власник. Кожна з них має унікальні можливості, детально описані нижче.

Ролі мають ієрархічну структуру (див. рис. 2.4): гість – базова роль, яку можуть розширювати інші. Роль користувача є основною для розширених – адміністратора, техніка й власника, які успадковують її функції та мають додаткові дозволи.

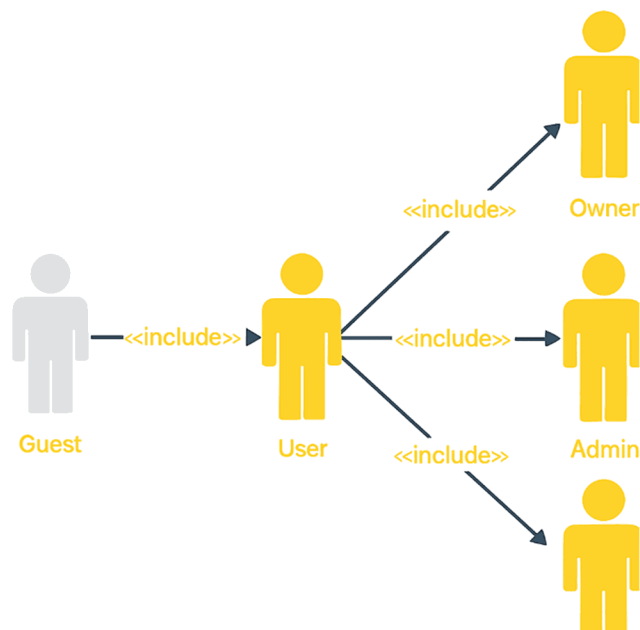


Рисунок 2.4 – Ієрархія ролей користувачів у системі

Систему розроблено таким чином, що кожна наступна роль успадковує дозволи попередньої. Почнемо з базової ролі – гостя. Роль гостя є відправною точкою для всіх користувачів системи. Вона забезпечує базове ознайомлення з функціональністю системи та доступними даними (рис. 2.5).



Рисунок 2.5 – Діаграма прецедентів взаємодії ролі «Гість» із системою моніторингу середовища

Має найменші дозволи та привілеї в системі. Гість може переглядати інформацію веб-інтерфейсу через головне меню, зокрема список загальнодоступних місць, таких як лекційні аудиторії, громадські простори чи інші локації. Цей список надає можливість перегляду даних про конкретне місце, включаючи характеристики повітряного середовища на основі показників датчиків, а також історичні дані якості повітря, що дозволяє відстежувати зміни параметрів і переглядати інформацію про встановлені сенсори. Таким чином, користувач отримує доступ до переліку встановлених датчиків і відповідних параметрів.

Наступною роллю в ієрархії є користувач. У порівнянні з гостем, користувач має розширені права, що дозволяє йому активніше взаємодіяти із системою. Він може не лише переглядати дані, а й отримувати доступ до більш детальної інформації та виконувати певні дії, які недоступні для гостя (рис. 2.6).

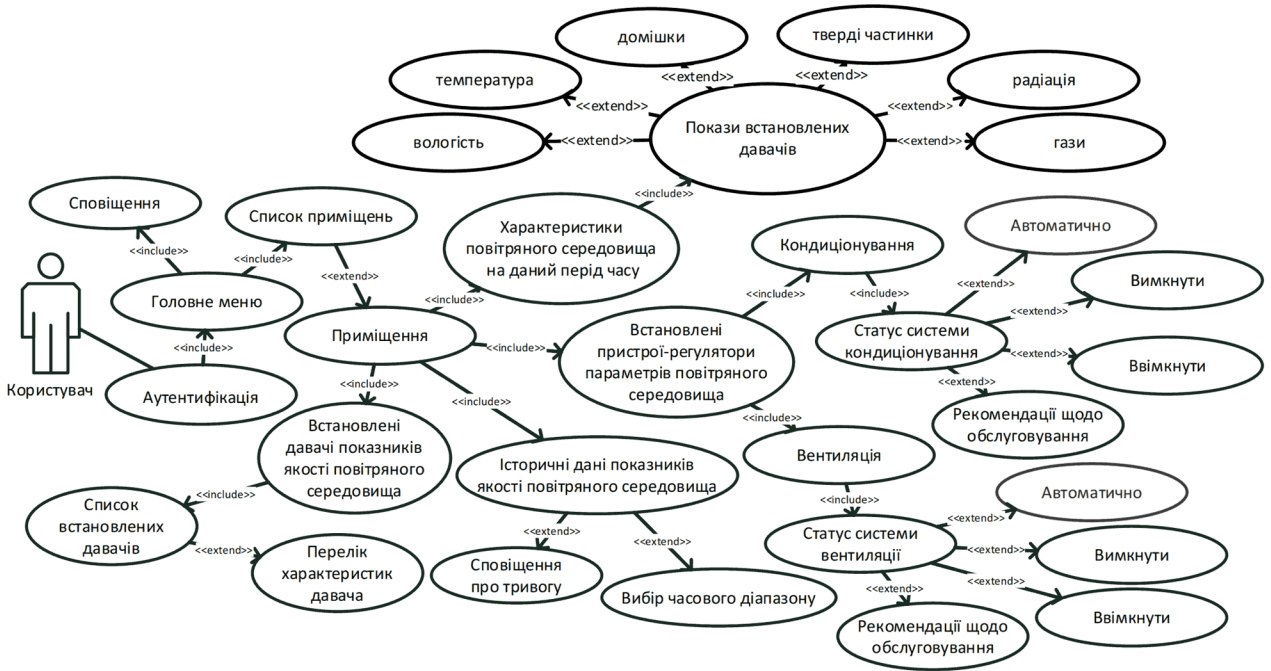


Рисунок 2.6 – Діаграма прецедентів взаємодії ролі «Користувач» із системою моніторингу середовища

Користувачі інформаційної системи моніторингу та регулювання показників якості навколишнього середовища в «розумних місцях» мають більше можливостей, ніж гості. Вони можуть виконувати всі дії, доступні для гостей, а після проходження аутентифікації отримують доступ до списку сповіщень та головного меню. Головне меню містить інформацію про веб-інтерфейс інформаційної системи та перелік місць, доступних для моніторингу.

Перелік місць включає характеристики атмосферного середовища, пристрої для регулювання параметрів повітря, встановлені в кожному місці, а також історичні дані щодо якості повітря. Передбачено можливість переходу до історичних даних датчиків, що дозволяє відображати сповіщення про перевищення порогових значень або вибирати потрібний часовий діапазон. Також у списку міститься інформація про встановлені датчики якості повітря – зокрема перелік сенсорів та характеристики кожного з них.

Під час перегляду характеристик повітряного середовища користувач може окремо аналізувати показання встановлених датчиків, зокрема вологості, температури, домішок, твердих частинок, радіації та газів, що можуть бути присутні у повітрі.

Окрім цього, користувач має доступ до інформації про систему вентиляції та кондиціонування повітря, яка використовується для регулювання мікроклімату «розумного приміщення». За допомогою інтерфейсу можна переглядати її поточний стан, а також перемикає між автоматичним і ручним режимами керування. Система вентиляції має аналогічний набір функцій: автоматичне або ручне відкриття й закриття, а також рекомендації щодо технічного обслуговування.

Цю систему також можна використовувати як рекомендаційну: на основі сигналів і показників систем кондиціонування або вентиляції вона може генерувати повідомлення з порадами звернутися до сервісного центру для подальшого обслуговування.

Наступним рівнем в ієрархії є адміністратор, який має всі права користувача, а також додаткові адміністративні функції (рис. 2.7).



Рисунок 2.7 – Діаграма прецедентів взаємодії ролі «Адміністратор» із системою моніторингу середовища

Адміністратор інформаційної системи, що забезпечує функціонування внутрішнього середовища «розумного місця», має найвищі повноваження в системі. Він може виконувати всі дії, доступні користувачам, а також, після проходження аутентифікації, отримує доступ до головного меню для перегляду інформації веб-інтерфейсу, списку місць, сповіщень і до використання панелі адміністрування.

Панель адміністрування дає змогу переглядати статистику використання веб-інтерфейсу, керувати обліковими записами користувачів, призначати їм ролі, додавати або блокувати користувачів, а також змінювати або видаляти наявні ролі. Адміністратор має змогу налаштовувати параметри датчиків та виконавчих механізмів, контролювати мережеві налаштування, продуктивність системи та загальний стан її функціонування.

Наступною роллю є модератор, який поєднує в собі частину адміністративних і технічних функцій (рис. 2.8).



Рисунок 2.8 – Діаграма прецедентів взаємодії ролі «Модератор» із системою моніторингу середовища

Адміністратор має як адміністративні повноваження, так і технічні обов'язки. Він може виконувати всі операції, доступні звичайним користувачам, а також додавати або блокувати облікові записи, переглядати статистику використання системи, налаштовувати датчики та виконавчі механізми, надавати технічну підтримку, контролювати технічний стан системи, виконувати технічне обслуговування пристроїв і здійснювати налагодження системи.

Наступною є роль власника пристрою. Вона відповідає виключно за керування тими пристроями, які належать цьому користувачеві (рис. 2.9).



Рисунок 2.9 – Діаграма повноважень ролі «Модератор» у системі моніторингу середовища

Власники пристроїв мають функціональні можливості, орієнтовані на керування власним обладнанням. Вони можуть виконувати всі дії, доступні звичайним користувачам, а також додавати або блокувати користувачів для своїх пристроїв, переглядати статистику використання виключно щодо власного обладнання, налаштовувати датчики та виконавчі механізми, керувати пристроями (додавати, налаштовувати та видаляти), а також призначати ролі користувачам своїх пристроїв.

Таким чином, система забезпечує широкий спектр функцій для різних категорій користувачів, дозволяючи ефективно керувати пристроями та контролювати якість повітря в приміщеннях. Ролі користувачів визначено таким чином, щоб кожен мав доступ до необхідних інструментів і даних відповідно до своїх повноважень і обов'язків.

РОЗДІЛ 3

РЕАЛІЗАЦІЯ ПРОЕКТНОГО РІШЕННЯ

3.1 Програмна реалізація веб-інтерфейсу

Панель користувача є важливою складовою будь-якої інформаційної системи й може використовуватися для керування користувачами та їхніми ролями. Наприклад панель адміністратора містить такі дані:

- ім'я користувача;
- дату створення облікового запису;
- роль користувача;
- дату останнього оновлення даних користувача;
- можливість змінити роль користувача.

Панель адміністрування виконує низку ключових функцій, що забезпечують ефективне керування користувачами системи. Одним із її основних завдань є зручне відображення списку користувачів. У цьому списку подаються імена, дата створення облікового запису, поточна роль та дата останнього оновлення, що дає змогу адміністраторам швидко орієнтуватися в структурі користувачів і приймати обґрунтовані рішення.

Важливою функцією панелі є редагування ролей користувачів. Адміністратори мають змогу змінювати ролі залежно від функціональних обов'язків і рівня доступу кожного користувача, що забезпечує гнучкість у керуванні правами доступу до різних елементів системи.

Щоб полегшити роботу з великою кількістю облікових записів, панель повинна підтримувати функції фільтрації та пошуку. Це дозволяє знаходити потрібного користувача за іменем або іншими критеріями. Крім того панель повинна бути надійно захищена від несанкціонованого доступу, а всі дії, які в ній виконуються, мають реєструватися у відповідних журналах для подальшого аудиту та контролю.

Панель адміністрування відображає таблицю з даними користувачів. Кожен рядок таблиці містить інформацію про окремого користувача: ім'я, дату створення облікового запису, роль, дату останнього оновлення та розкривний список для зміни ролі. Адміністратор може змінити роль користувача, вибравши нове значення зі спадного списку. Після вибору нової ролі на сервер надсилається запит на оновлення даних. У разі успішного оновлення нова роль одразу відображається в таблиці.

На рисунках 3.1 – 3.4 зображено елементи інтерфейсу, призначені для реалізації інтерактивного графіка даних із сенсорів. Користувачі можуть переглядати дані з різних датчиків, до яких вони мають доступ. Ці датчики вимірюють такі параметри як температура, вологість, фонове випромінювання, концентрація дрібнодисперсних частинок і різні гази.



Рисунок 3.1 – Віджет інтерактивного графіка чистоти повітря

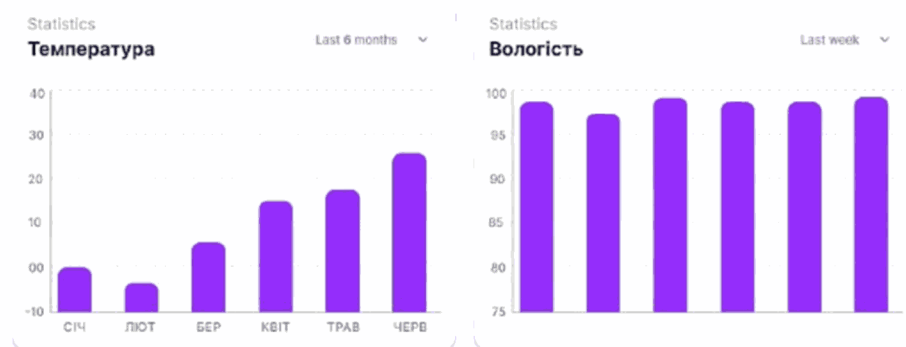


Рисунок 3.2 – Віджети інтерактивного графіка даних температури та вологості

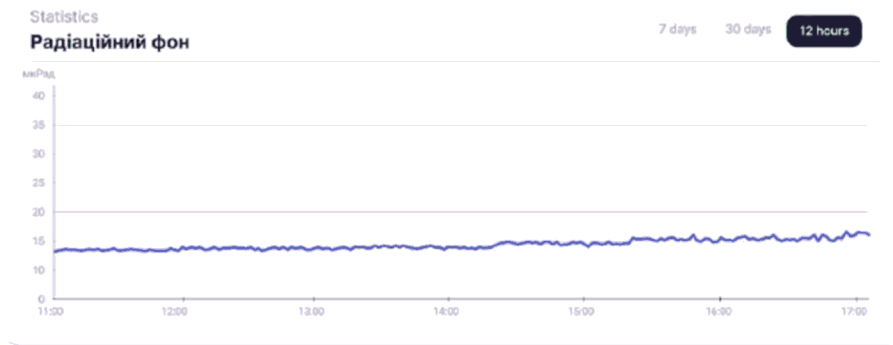


Рисунок 3.3 – Віджет інтерактивного графіка даних радіаційного фону

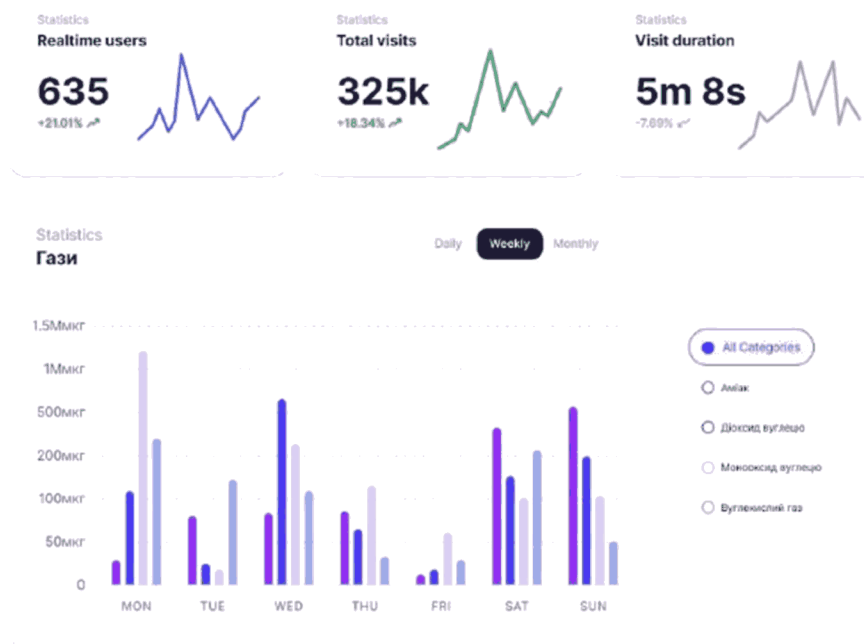


Рисунок 3.4 – Віджет з інтерактивними графіками показників концентрації дрібнодисперсних частинок і різних газів

Користувачі можуть сортувати історичні дані та змінювати інтервал часу для перегляду. Кожен параметр якості повітря відображається на окремій діаграмі. Діаграми є інтерактивними: користувачі можуть отримати більше інформації, навівши курсор на певні точки графіка. Це забезпечує зручний спосіб аналізу даних у режимі реального часу.

Користувачі можуть обирати різні часові інтервали для перегляду даних: щоденно, щотижнево, щомісячно або щорічно. Така можливість дозволяє детально аналізувати тенденції та зміни параметрів якості повітря протягом різних періодів.

Додаткові функції фільтрації даних дають змогу користувачам обирати окремі параметри для перегляду. Наприклад, можна фільтрувати дані за типом газу або переглядати лише обрані категорії даних.

Кожен компонент на сторінці відповідає за відображення інформації з датчиків системи моніторингу внутрішнього середовища. Компонент `AirQualityDashboard` є основним контейнером для всіх діаграм (рис. 3.5). Він відповідає за завантаження даних з бази та передачу їх до окремих графічних компонентів. Компонент використовує хук `useEffect` для виклику функції `fetchAirQualityData` під час першого завантаження сторінки. Виклик функції здійснюється через `useDispatch`, а отримання даних – через `useSelector`.

```
import React, { useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { fetchAirQualityData } from './actions'; // Імпорт дії для отримання даних

// Імпорт компонентів для візуалізації окремих параметрів
import TemperatureChart from './TemperatureChart';
import HumidityChart from './HumidityChart';
import RadiationChart from './RadiationChart';
import ParticlesChart from './ParticlesChart';
import GasesChart from './GasesChart';

const AirQualityDashboard = () => {
  // ◆ Частина 1: Ініціалізація Redux-зв'язку
  const dispatch = useDispatch(); // Отримання функції dispatch
  const { data, loading } = useSelector(state => state.airQuality); // Отримання стану з Redux store

  // ◆ Частина 2: Завантаження даних при монтуванні компонента
  useEffect(() => {
    dispatch(fetchAirQualityData()); // Запуск дії отримання даних
  }, [dispatch]);

  // ◆ Частина 3: Відображення контенту
  if (loading) {
    // Якщо дані ще завантажуються – показуємо повідомлення
    return <p>Loading data...</p>;
  }

  // Після завантаження – показуємо графіки
  return (
    <div>
      <TemperatureChart data={data.temperature} />
      <HumidityChart data={data.humidity} />
      <RadiationChart data={data.radiation} />
      <ParticlesChart data={data.particles} />
      <GasesChart data={data.gases} />
    </div>
  );
};
```

Рисунок 3.5 – Ініціалізація Redux-зв'язку, завантаження даних, відображення контенту

Компонент `TemperatureChart` відповідає за візуалізацію даних про температуру. Він отримує дані у вигляді пропсів і використовує бібліотеку `Recharts` для побудови лінійної діаграми. `LineChart` задає основну структуру діаграми, `CartesianGrid` додає сітку у фоновому режимі, `XAxis` і `YAxis` відображають осі координат, `Tooltip` показує додаткову інформацію при наведенні курсора, а `Line` малює саму лінію діаграми (рис. 3.6).

```
import React from 'react';
import {
  LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, Legend, ResponsiveContainer
} from 'recharts'; // Імпорт компонентів з бібліотеки Recharts

// ♦ Компонент для відображення графіка температури
const TemperatureChart = ({ data }) => {
  return (
    // Контейнер, що адаптується до ширини батьківського елемента
    <ResponsiveContainer width="100%" height={300}>
      {/* Побудова лінійного графіка на основі переданих даних */}
      <LineChart data={data}>
        {/* Сітка з пунктирними лініями для кращого читання графіка */}
        <CartesianGrid strokeDasharray="3 3" />

        {/* Вісь X з прив'язкою до ключа 'time' у даних */}
        <XAxis dataKey="time" />

        {/* Вісь Y для значень температури */}
        <YAxis />

        {/* Підказка при наведенні на точку графіка */}
        <Tooltip />

        {/* Легенда, що пояснює, яка лінія за що відповідає */}
        <Legend />

        {/* Лінія графіка з даних по ключу 'value' і заданим кольором */}
        <Line type="monotone" dataKey="value" stroke="#8884d8" />
      </LineChart>
    </ResponsiveContainer>
  );
};

export default TemperatureChart;
```

Рисунок 3.6 – Компонент для виведення температурних показників у приміщенні

Процес створення сторінки автентифікації користувача (рисунок 3.7) охоплює кілька взаємопов'язаних етапів. Спершу формується інтерфейс, який включає стандартну форму для введення імені користувача, електронної адреси та пароля, а також кнопку для входу за допомогою стороннього сервісу, наприклад Google. Далі реалізується клієнтська логіка, що передбачає обробку подій введення даних, перевірку їхньої коректності та організацію взаємодії з сервером для підтвердження автентичності. На наступному етапі створюється серверна частина, яка відповідає за обробку запитів на реєстрацію та вхід, забезпечує збереження й перевірку даних у базі та здійснює інтеграцію з зовнішніми сервісами автентифікації через протокол OAuth.

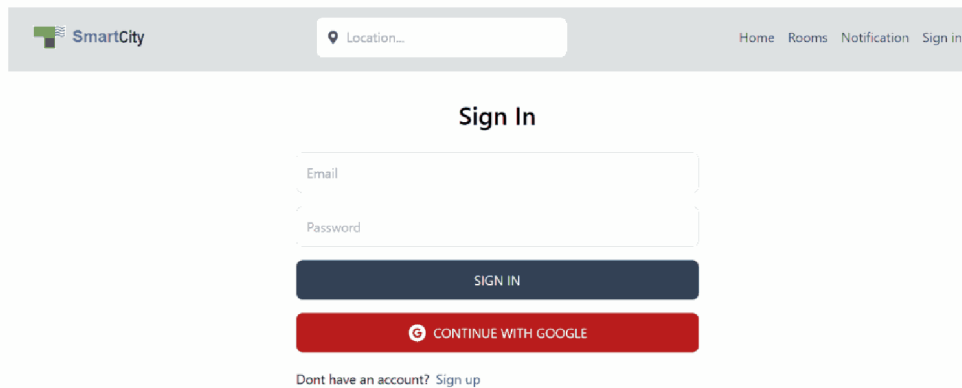


Рисунок 3.7 – Інтерфейс користувацької автентифікації

Для реалізації автентифікації через Google використовується сервіс Firebase. Після натискання кнопки Google Authentication викликається функція, яка відкриває вікно автентифікації Google. Після успішної автентифікації отримані дані надсилаються на сервер, зберігаються в базі даних, а також створюється сесія користувача. На стороні сервера використовується середовище Node.js та фреймворк Express для обробки запитів.

3.2 Створення та наповнення бази даних

База даних є важливою складовою системи, оскільки забезпечує зберігання всієї інформації, необхідної для її функціонування. У цьому проєкті використовується база даних MySQL, оскільки вона є надійною, масштабованою та підтримує широкий спектр функцій SQL.

Процес розробки бази даних розпочинається зі створення схеми, яка включає визначення всіх таблиць та зв'язків між ними. Основними таблицями є: таблиця користувачів, таблиця ролей, таблиця розташувань, таблиця типів платформ, таблиця пристроїв Інтернету речей, таблиця датчиків і таблиця виконавчих пристроїв. Після формування структури бази даних необхідно заповнити таблиці початковими даними. Наприклад, можна додати кілька датчиків і відповідні початкові значення їх параметрів (рис. 3.8, 3.9).

```
-- Додавання сенсора вологості до таблиці Sensor
INSERT INTO Sensor (sensorID, sensorType, sensor_status)
VALUES (1, 'Humidity', 'active');

-- Додавання сенсора температури до таблиці Sensor
INSERT INTO Sensor (sensorID, sensorType, sensor_status)
VALUES (2, 'Temperature', 'active');

-- Додавання мультисенсора до таблиці Sensor
INSERT INTO Sensor (sensorID, sensorType, sensor_status)
VALUES (3, 'Multi', 'active');
```

Рисунок 3.8 – Компонент для виведення температурних показників у приміщенні

Важливою складовою системи моніторингу внутрішнього середовища «розумного будинку» є процес ефективного отримання та обробки даних із бази даних. Цей процес забезпечує актуальність відображуваної інформації та дозволяє зберігати історичні дані. База даних складається з кількох таблиць, кожна з яких відповідає за зберігання інформації певного типу датчика.

```

-- Додавання показника температури від сенсора з ID = 1
INSERT INTO TemperatureSensor (sensorID, timestamp, temperature)
VALUES (1, NOW(), 23.5);

-- Додавання показника вологості від сенсора з ID = 1
INSERT INTO HumiditySensor (sensorID, timestamp, humidityLevel)
VALUES (1, NOW(), 45.2);

-- Додавання показника температури від сенсора з ID = 2
INSERT INTO TemperatureSensor (sensorID, timestamp, temperature)
VALUES (2, NOW(), 26.1);

-- Додавання показника забруднення повітря (тип PM2.5) від сенсора з ID = 2
INSERT INTO PollutionSensor (sensorID, timestamp, pollutionLevel, pollutionType)
VALUES (2, NOW(), 55, 'PM2.5');

-- Додавання показника забруднення повітря (тип PM10) від сенсора з ID = 2
INSERT INTO PollutionSensor (sensorID, timestamp, pollutionLevel, pollutionType)
VALUES (2, NOW(), 30, 'PM10');

-- Додавання показника вологості від сенсора з ID = 3
INSERT INTO HumiditySensor (sensorID, timestamp, humidityLevel)
VALUES (3, NOW(), 50.7);

-- Додавання показника концентрації газу CO2 від сенсора з ID = 3
INSERT INTO GasSensor (sensorID, timestamp, gasType, gasConcentration)
VALUES (3, NOW(), 'CO2', 400);

```

Рисунок 3.9 – Реалізація додавання кількох типів сенсорів у таблицю sensors

Щоб отримати поточні дані з датчика, необхідно звернутися до відповідної таблиці бази даних за допомогою SQL-запитів. Кожна таблиця містить дані для конкретного типу датчика, наприклад, таблиця TemperatureSensor зберігає показники температури, таблиця HumiditySensor – показники вологості тощо. Ці таблиці оновлюються щоразу, коли датчик надсилає нові дані.

Перед початком роботи з даними слід ознайомитися зі структурою головної таблиці датчиків, яка називається Sensors. Вона містить загальну інформацію про кожен датчик, зокрема його унікальний ідентифікатор (sensorID), тип (sensorType) і статус (sensor_status). Щоб отримати інформацію про датчик, необхідно виконати SQL-запит типу SELECT (рис. 3.10).

```
-- Вибірка полів: ідентифікатора сенсора, його типу та статусу
SELECT
    sensorID,          -- Унікальний ідентифікатор сенсора
    sensorType,         -- Тип сенсора (наприклад: Temperature, Humidity)
    sensor_status      -- Статус сенсора (наприклад: active, inactive)
FROM
    Sensor              -- Назва таблиці, з якої отримуються дані
WHERE
    sensorID = 1;       -- Умова: обрати лише той сенсор, у якого ID дорівнює 1
```

Рисунок 3.10 – Отримання інформації про сенсор з ID = 1

Окрім основної таблиці датчиків, кожен тип сенсора має окрему таблицю. У цих таблицях зберігаються детальні дані, отримані від кожного датчика. Наприклад, щоб отримати розширену інформацію, можна виконати операцію JOIN між основною таблицею датчиків та відповідною таблицею показників. Наведений нижче запит (рис. 3.11) витягує дані про температуру та вологість для датчика з ідентифікатором ID = 1.

```
-- Вибірка інформації про сенсор, його температуру та вологість
SELECT
    s.sensorID,          -- Унікальний ідентифікатор сенсора
    s.sensorType,         -- Тип сенсора (наприклад: Temperature, Multi)
    ts.temperature,       -- Значення температури з таблиці TemperatureSensor
    hs.humidityLevel      -- Значення вологості з таблиці HumiditySensor
FROM
    Sensor s              -- Основна таблиця сенсорів (скорочена як 's')

-- Об'єднання з таблицею температурних сенсорів за sensorID
LEFT JOIN
    TemperatureSensor ts ON s.sensorID = ts.sensorID

-- Об'єднання з таблицею сенсорів вологості за sensorID
LEFT JOIN
    HumiditySensor hs ON s.sensorID = hs.sensorID

-- Умова: обрати лише сенсор із sensorID = 1
WHERE
    s.sensorID = 1;
```

Рисунок 3.11 – Отримання температури та вологості від сенсора з ID = 1

При заміні датчика, важливо зберегти дані зі старого пристрою для подальшого аналізу минулих періодів. Цей процес передбачає збереження поточних даних датчика в таблиці історичних даних, після чого основна таблиця оновлюється інформацією про новий датчик. Перед заміною хост повинен ініціювати дію, яка записує поточні дані у відповідну таблицю. Для цього використовується оператор `INSERT INTO ... SELECT`, що дозволяє перенести актуальні показники до таблиці. На рисунку 3.12 наведено приклад, як зберегти дані про поточну температуру в таблиці «Історія датчика температури». Аналогічно, щоб записати поточні дані про вологість, подібна команда (рис. 3.13)

```
-- Додавання поточного значення температури до таблиці історії вимірювань
INSERT INTO TemperatureSensorHistory (sensorID, timestamp, temperature)
SELECT
    sensorID,          -- Ідентифікатор сенсора
    NOW(),              -- Поточна дата й час як мітка часу
    temperature         -- Поточне значення температури
FROM
    TemperatureSensor
WHERE
    sensorID = 1;      -- Умова: обрати лише сенсор з ID = 1
```

Рисунок 3.12 – Збереження актуального значення температури до таблиці історичних даних

```
-- Додавання поточного значення вологості до таблиці історії вимірювань
INSERT INTO HumiditySensorHistory (sensorID, timestamp, humidityLevel)
SELECT
    sensorID,          -- Ідентифікатор сенсора
    NOW(),              -- Поточна дата й час (мітка часу)
    humidityLevel       -- Поточне значення вологості
FROM
    HumiditySensor
WHERE
    sensorID = 1;      -- Умова: обрати лише дані для сенсора з ID = 1
```

Рисунок 3.13 – Збереження актуального значення вологості до таблиці історичних даних

Після завершення реєстрації даних старий датчик замінюється, а новий запис додається до головної таблиці датчиків, щоб відображати показання нового пристрою. Наприклад, для вставлення нових даних про температуру, вологість і радіацію можна скористатися стандартними операторами (рис. 3.14).

```
-- Додавання нового значення температури для сенсора з ID = 1
INSERT INTO TemperatureSensor (sensorID, timestamp, temperature)
VALUES (1, NOW(), 24.0);           -- 24.0 °C, поточна дата та час

-- Додавання нового значення вологості для сенсора з ID = 1
INSERT INTO HumiditySensor (sensorID, timestamp, humidityLevel)
VALUES (1, NOW(), 50.0);          -- 50.0 %, поточна дата та час

-- Додавання нового значення рівня радіації для сенсора з ID = 1
INSERT INTO RadiationSensor (sensorID, timestamp, radiationLevel)
VALUES (1, NOW(), 0.02);          -- 0.02 мкЗв/год, поточна дата та час
```

Рисунок 3.14 – Додавання нових записів після заміни сенсора

Під час аналізу даних датчика важливо перевірити наявність історичних даних (особливо якщо датчик було замінено) та включити їх до аналізу. Щоб з'ясувати, чи існують історичні дані, можна виконати запит до відповідної таблиці історії. Наприклад, щоб перевірити історичні дані температури для датчика з ID 1, можна використати такий запит (рис. 3.15).

```
-- Отримання списку усіх температурних вимірювань для сенсора з ID = 1
SELECT
    sensorID,           -- Ідентифікатор сенсора
    timestamp,          -- Час фіксації температурного значення
    temperature         -- Значення температури (°C)
FROM
    TemperatureSensorHistory
WHERE
    sensorID = 1;       -- Вибірка лише для сенсора з ID = 1
```

Рисунок 3.15 – Вибірка історичних температурних даних для сенсора з ID = 1

Якщо історичні дані знайдено, їх можна отримати та об'єднати з поточними даними датчика. Щоб отримати історичні дані температури для датчика з ідентифікатором 1, можна використати такий запит (рис. 3.16).

```
-- Вибірка історичних температурних значень для конкретного сенсора
SELECT
    sensorID,      -- Унікальний ідентифікатор сенсора, з якого отримано дані
    timestamp,     -- Час фіксації кожного вимірювання температури
    temperature    -- Виміряне значення температури (у °C)
FROM
    TemperatureSensorHistory -- Таблиця з історією температурних показників
WHERE
    sensorID = 1; -- Умова: вибрати тільки записи, що належать сенсору з ID = 1
```

Рисунок 3.16 – Отримання історичних даних для сенсора з ідентифікатором 1

Щоб об'єднати поточні дані з історичними, можна скористатися операцією UNION. Такий підхід гарантує, що всі точки даних – як поточні, так і історичні – буде включено до вибірки. Наступний запит (рис. 3.17) об'єднує поточні та історичні дані температури для датчика з ідентифікатором 1.

```
-- Вибірка всіх (поточних і архівних) температурних значень для сенсора з ID = 1
SELECT
    sensorID,      -- Ідентифікатор сенсора
    timestamp,     -- Час фіксації температурного значення
    temperature    -- Значення температури (у °C)
FROM
    TemperatureSensor      -- Поточна таблиця з останніми показниками
WHERE
    sensorID = 1          -- Фільтр по ID сенсора

-- Об'єднання з історичними даними без видалення повторів
UNION ALL

SELECT
    sensorID,      -- Ідентифікатор сенсора
    timestamp,     -- Час фіксації історичного значення
    temperature    -- Історичне значення температури
FROM
    TemperatureSensorHistory -- Таблиця історії температур
WHERE
    sensorID = 1;          -- Фільтр по ID сенсора
```

Рисунок 3.17 – Об'єднання поточних і архівних даних температури

Безперервний запис і збереження даних у поточних та історичних таблицях, надання точних позначок часу дозволяють ефективно керувати базою даних датчиків. Наступний запит (рис. 3.18) отримує всі дані для датчика з ідентифікатором 1 шляхом об'єднання інформації з кількох таблиць.

```
-- Отримання поточних значень з трьох сенсорних таблиць для одного сенсора
SELECT
    s.sensorID,                -- Унікальний ідентифікатор сенсора
    s.sensorType,              -- Тип сенсора (наприклад, 'Multi', 'Temperature', тощо)

    ts.timestamp AS tempTimestamp, -- Час останнього вимірювання температури
    ts.temperature,             -- Поточне значення температури (°C)

    hs.timestamp AS humTimestamp, -- Час останнього вимірювання вологості
    hs.humidityLevel,           -- Поточне значення вологості (%)

    rs.timestamp AS radTimestamp, -- Час останнього вимірювання радіації
    rs.radiationLevel           -- Поточне значення рівня радіації (наприклад, мкЗв/год)
FROM
    Sensor s                    -- Основна таблиця з інформацією про сенсори

-- Приєднання таблиці з температурними даними
LEFT JOIN
    TemperatureSensor ts ON s.sensorID = ts.sensorID

-- Приєднання таблиці з даними вологості
LEFT JOIN
    HumiditySensor hs ON s.sensorID = hs.sensorID

-- Приєднання таблиці з даними радіації
LEFT JOIN
    RadiationSensor rs ON s.sensorID = rs.sensorID

-- Фільтрація: обрати тільки сенсор з ID = 1
WHERE
    s.sensorID = 1;
```

Рисунок 3.18 – Отримання всіх актуальних даних сенсора з ідентифікатором 1

3.3 Впровадження механізмів реєстрації та автентифікації користувачів

Автентифікація користувачів є важливим аспектом веб-інтерфейсів, що гарантує доступ лише автентифікованим користувачам до певних ресурсів і можливості виконання визначених дій. Інформаційна система моніторингу внутрішнього середовища «розумних будівель» використовує метод

двофакторної автентифікації. Це включає традиційне введення даних – електронної пошти та пароля – а також автентифікацію за допомогою OAuth через сервіс Google.

Традиційний процес автентифікації починається зі створення елементів інтерфейсу користувача (UI) для сторінок реєстрації та входу. Ці сторінки інтуїтивно зрозумілі, прості у використанні та містять поля для введення імені користувача, електронної пошти й пароля. Форма також має кнопку надсилання для запуску процесу автентифікації (рис. 3.19). Крім того, для забезпечення гнучкості інтерфейс містить опцію входу через OAuth, що дозволяє користувачам автентифікуватися за допомогою облікового запису Google. Така інтеграція не лише спрощує вхід, а й підвищує загальний рівень безпеки.

The image shows a web form for user registration. At the top, there is a location input field with a pin icon and the placeholder text 'Місто/Локація/Регіон...'. To the right of this field is a 'Sign in' button. Below the location field is the title 'SignUp'. The form contains three input fields: 'Username', 'Email', and 'Password'. Below these fields are two buttons: a dark blue button labeled 'SIGN UP' and a red button labeled 'CONTINUE WITH GOOGLE' with the Google logo. At the bottom of the form, there is a link that says 'Have an account? Sign in'. The footer of the page has three icons: a house icon for 'Home', a list icon for 'Rooms', and a bell icon for 'Notification'.

Рисунок 3.19 – Форма входу до системи з перевіркою даних користувача

Обробка користувацького введення в React є важливою складовою динамічного управління формами. Використовуючи хук `useState` у React, ми можемо зручно керувати станом даних форми (рис. 3.20). Обробники подій дозволяють фіксувати зміни в полях введення, що дає змогу оновлювати дані в реальному часі та здійснювати їх перевірку перед надсиланням.

```
// Імпорт хука useState з React
import { useState } from 'react';

// Ініціалізація стану formData як порожнього об'єкта
const [formData, setFormData] = useState({});

// Обробник зміни значення в будь-якому input-полі форми
const handleChange = (e) => {
  setFormData({
    ...formData, // Зберігаємо всі наявні поля
    [e.target.id]: e.target.value // Оновлюємо поле з id, що відповідає введеному значенню
  });
};
```

Рисунок 3.20 – Керування станом полів вводу за допомогою useState у React

Коли користувач надсилає реєстраційну форму, програма відправляє POST-запит на сервер із даними користувача (рис. 3.21).

```
// Асинхронна функція для обробки відправлення форми
const handleSubmit = async (e) => {
  e.preventDefault(); // Забороняємо стандартну поведінку форми (перезавантаження сторінки)

  try {
    setLoading(true); // Увімкнення індикатора завантаження

    // Відправлення POST-запиту до API для реєстрації
    const res = await fetch("/api/auth/signup", {
      method: "POST", // HTTP-метод
      headers: {
        "Content-Type": "application/json", // Вказуємо, що відправляємо JSON
      },
      body: JSON.stringify(formData), // Сериалізуємо об'єкт formData у JSON
    });

    const data = await res.json(); // Розбираємо відповідь API у форматі JSON

    // Якщо API повертає success: false – показуємо помилку
    if (data.success === false) {
      setLoading(false); // Вимикаємо завантаження
      setError(data.message); // Виводимо повідомлення про помилку
      return; // Завершуємо виконання
    }

    // Якщо реєстрація успішна:
    setLoading(false); // Вимикаємо завантаження
    setError(null); // Очищаємо попередню помилку
    navigate("/sign-in"); // Переходимо на сторінку входу
  } catch (error) {
    // Обробка помилки, якщо запит не вдався (наприклад, через мережу)
    setLoading(false); // Вимикаємо завантаження
    setError(error.message); // Виводимо текст помилки
  }
};
```

Рисунок 3.21 – Форма реєстрації з використанням асинхронної функції

Ця операція передбачає встановлення стану завантаження, щоб надати користувачеві зворотний зв'язок, а також коректну обробку можливих помилок. Процес надсилання форми інкапсульовано в асинхронну функцію, яка використовує `fetch` для зв'язку з сервером.

На стороні сервера контролер автентифікації відповідає за обробку запитів на реєстрацію та вхід. Під час процесу реєстрації контролер хешує пароль користувача за допомогою `bcryptjs` перед збереженням у базі даних. Цей крок є критично важливим, оскільки хешування паролів забезпечує додатковий рівень безпеки: навіть якщо базу даних буде зламано, справжній пароль залишиться захищеним. Дані користувача зберігаються в базі даних, після чого клієнту повертається повідомлення про успішне виконання операції (рис. 3.22).

```
// Контролер входу користувача
export const signin = async (req, res, next) => {
  const { email, password } = req.body;

  try {
    // Шукаємо користувача за email
    const validUser = await User.findOne({ email });

    // Якщо користувача не знайдено – кидаємо 404
    if (!validUser) return next(errorHandler(404, "User not found!"));

    // Перевіряємо правильність пароля
    const validPassword = bcryptjs.compareSync(password, validUser.password);
    if (!validPassword) return next(errorHandler(401, "Wrong credentials!"));

    // Генеруємо JWT-токен з ID користувача
    const token = jwt.sign(
      { id: validUser._id },
      process.env.JWT_SECRET // Секрет з .env
    );

    // Вилучаємо пароль зі структури перед відправкою клієнту
    const { password: pass, ...rest } = validUser._doc;

    // Встановлюємо токен у HTTP-only cookie та повертаємо дані користувача
    res
      .cookie("access_token", token, { httpOnly: true }) // Токен недоступний JavaScript
      .status(200)
      .json(rest); // Повертаємо інші дані користувача (без пароля)
  } catch (error) {
    next(error); // Обробка винятків
  }
};
```

Рисунок 3.22 – Логіка автентифікації та обробка запитів на сервері

Щоб увійти до системи, сервер перевіряє облікові дані, надані користувачем, зі збереженими у базі даних. Якщо дані є дійсними, сервер генерує JSON Web Token (JWT) і надсилає його клієнту. JWT використовується для підтримки сеансу без збереження стану, що дозволяє клієнту безпечно автентифікувати наступні запити. Токен встановлюється у вигляді cookie з прапорцем `httpOnly`, що підвищує безпеку, запобігаючи доступу до нього з клієнтських скриптів.

Потік автентифікації через OAuth, зокрема за допомогою Google, забезпечує простішу та безпечнішу альтернативу традиційному входу. Процес починається зі створення компонента OAuth в інтерфейсі. Цей компонент використовує Firebase для керування автентифікацією через Google. Коли користувач натискає кнопку входу через Google, відкривається спливаюче вікно для авторизації через обліковий запис Google.

Інтеграція з Firebase потребує імпорту необхідних модулів, таких як `GoogleAuthProvider`, `getAuth` і `signInWithPopup`. Після успішного входу за допомогою Google, Firebase повертає об'єкт результату, який містить інформацію про користувача.

На стороні сервера контролер OAuth перевіряє, чи існує користувач у базі даних. Якщо користувач уже зареєстрований, сервер генерує JWT і повертає його клієнту, як і у традиційному потоці входу. Якщо ж користувач відсутній, сервер створює новий запис. Для підвищення безпеки модуль `bcryptjs` генерує випадковий пароль і хешує його перед збереженням. Дані нового користувача – зокрема ім'я, електронна пошта та зображення профілю, надані Google – зберігаються в базі даних. Після цього сервер створює JWT для нового користувача і повертає його клієнту.

Цей процес автентифікації через OAuth не лише спрощує вхід для користувача, а й повністю використовує потужну інфраструктуру безпеки. Впровадження автентифікації користувачів у нашої інформаційній системі «розумних будівель» дозволяє ефективно поєднати безпеку та зручність.

РОЗДІЛ 4

ОХОРОНА ПРАЦІ

4.1 Вимоги безпеки праці під час експлуатації систем вентиляції, опалення та кондиціонування повітря

Згідно ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування» в проектах опалення, вентиляції та кондиціонування слід передбачати технічні рішення, що забезпечують різні параметри, ось деякі з них:

- нормовані параметри мікроклімату та концентрацію шкідливих речовин у повітрі зони обслуговування приміщень житлових будинків, громадських будівель і споруд, будівель адміністративного та побутового призначення згідно з санітарно-епідеміологічними вимогами;
- нормовані параметри мікроклімату та концентрацію шкідливих речовин у повітрі робочої зони виробничих, лабораторних та складських (далі - виробничих) приміщень у будівлях будь-якого призначення згідно з ГОСТ 12.1.005 і санітарно-гігієнічними вимогами до мікроклімату виробничих приміщень згідно з ДСН 3.3.6.042 та відповідно до положень розділу 5 "Параметри внутрішнього та зовнішнього повітря" цих Норм;
- охорону атмосферного повітря від вентиляційних викидів шкідливих речовин;
- механічну безпеку, електробезпеку, виконання вимог охорони праці під час монтажу, налагодження, випробувань та експлуатації опалювально-вентиляційного обладнання;
- вибухопожежобезпечність опалювально-вентиляційного обладнання.

У створенні сприятливих умов праці ефективним засобом є вентиляція – сукупність заходів та засобів призначених для забезпечення на постійних робочих місцях та зонах обслуговування виробничих приміщень метеорологічних умов та чистоти повітряного середовища, що відповідають

гігієнічним та технічним вимогам. Залежно від способу переміщення повітря вентиляція буває природна і механічна (штучна) [38].

Природна вентиляція здійснюється за рахунок сили вітру і природними (гравітаційними) силами. Вітер, обдуваючи споруду, попереду неї створює зону підвищеного тиску, а з протилежного боку виникає зона певного розрідження. Під дією напору вітер через фрамуги, кватирки, створи й інші отвори проникає у приміщення, а під дією розрідження забруднене повітря виходить назовні. Перевагою природної вентиляції є її дешевизна та простота експлуатації. Основний її недолік у тому, що повітря надходить у приміщення без попереднього очищення, а виділене відпрацьоване повітря також не очищається і забруднює довкілля.

Механічна вентиляція – це примусове видалення з приміщень забрудненого повітря і заміна його на свіже за допомогою вентиляційних агрегатів. Сукупність вентиляційного агрегату, повітроводів, регулювальних, пускових та інших пристроїв складає вентиляційну систему для конкретного виробничого приміщення. Штучна вентиляція може бути загальнообмінною, місцевою та комбінованою.

Природна та штучна вентиляції повинні відповідати наступним санітарногігієнічним вимогам: створювати в робочій зоні приміщень нормовані параметри повітряного середовища; не вносити в приміщення забруднене повітря ззовні або шляхом засмоктування забрудненого повітря з суміжних приміщень; не створювати на робочих місцях протягів чи різкого охолодження; бути доступними для управління та ремонту під час експлуатації; не створювати під час експлуатації додаткових незручностей, бути економічними, вибухопожежобезпечними, не заважати використовувати технологічні операції, не створювати перешкоди внутріцеховому транспорту, не впливати на якість продукції.

При роботі припливно-витяжної вентиляції необхідно, щоб кількість повітря, що надходить ззовні, не перевищувала або була на 10-15% меншою від кількості повітря, що видаляється витяжними пристроями.

Потребу в чистому повітрі на одного працівника на годину можна визначити за формулою:

$$a = k / p - g, \quad (4.1)$$

де: a – необхідний вентиляційний об'єм повітря;

k – кількість літрів вуглекислого газу, що виділяє людина за 1 годину;

p – допустима концентрація вмісту вуглекислого газу в приміщенні (0,1 %) – 1,0 л в 1 м³ повітря;

g – вміст вуглекислого газу в повітрі (0,03%) – 0,03 л в 1 м³ повітря.

Швидкість руху повітря має становити для видалення газів з холодних приміщень – 0,5-1 м/с, з теплих – 1,0-1,5 м/с, а для видалення пилу – відповідно 0,8-1,5 м/с і 1,5-2,5 м/с.

Місцева вентиляція забезпечує нормалізацію повітряного середовища на робочих місцях. Вона може бути припливною (повітряні душі, повітряні та повітряно-теплові завіси) і витяжною (вловлювання шкідливих речовин безпосередньо біля місць їх утворення).

Для створення та автоматичного підтримування в приміщенні заданих або таких, що змінюються за певною програмою умов мікроклімату використовують кондиціонування.

Кондиціонування повітря – це створення автоматичного підтримування в приміщенні, незалежно від зовнішніх умов (постійних чи таких, що змінюються), по визначеній програмі температури, вологості, чистоти і швидкості руху повітря. У відповідності з вимогами для конкретних приміщень повітря нагрівають або охолоджують, зволожують або висушують, очищають від забруднюючих речовин або піддають дезінфекції, дезодорації, озонуванню.

Для кондиціонування повітря у виробничих приміщеннях використовують такі кондиціонери:

- центральні, що встановлюються за межами робочих приміщень;
- місцеві, розміщені безпосередньо у приміщенні.

Будова, склад і експлуатація систем опалення мають відповідати вимогам ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування».

Для нагрівання окремих приміщень дозволяється використовувати електричні прилади з закритими спіралями і з такою потужністю споживання, яка б не призводила до підвищення сили струму понад допустиму для даної електромережі.

Опалювальні прилади розмішують у місцях, доступних для огляду, ремонту, очищення на відстані 0,1 м від поверхні стін. Не дозволяється розміщувати опалювальні прилади в нішах стін. Опалювальні прилади на сходових клітках розміщуються в нижніх поверхах, а також у відсіках тамбурів, які не мають зовнішніх дверей.

4.2 Розробка захисту від пожеж та вибухів в системах опалення, вентиляції, освітлення та кондиціонування повітря

Перед початком опалювального сезону теплові мережі, які розміщені у приміщеннях, калориферні й теплогенераторні установки, котельні, печі та інші опалювальні прилади повинні бути перевірені й відремонтовані. Несправні опалювальні прилади не повинні допускатися до експлуатації.

Гарячі поверхні тепломереж, що розташовані у приміщеннях, у яких вони можуть створити небезпеку спалахування парів, газів, пилу або аерозолів, потрібно ізолювати таким способом, щоб температура на поверхні теплоізованої конструкції була не менш ніж на 20% нижчою за температуру самоспалахування речовин [39].

Усі гарячі ділянки поверхонь трубопроводів і обладнання, яке розташоване в зоні можливого потрапляння на них вибухонебезпечних, горючих або легкозаймистих речовин, необхідно покрити металевою обшивкою. Не дозволяється експлуатація теплових мереж з просоченою вибухонебезпечними, горючими або легкозаймистими речовинами теплоізоляцією. Очищення печей та димоходів від сажі необхідно проводити перед початком, а також впродовж усього опалювального сезону, а саме [40]:

- кухонних кип'ятильників та плит – один раз на місяць;
- печей безперервної дії – не рідше одного разу на два місяці;
- опалювальних печей періодичної дії на рідкому та твердому паливі – не рідше одного разу на три місяці.

У приміщеннях котелень та інших теплогенеруючих установок населених пунктів і підприємств забороняється:

- сушити взуття, спецодяг та інші матеріали на паропроводах та котлах;
- працювати при відключених або зіпсованих приладах регулювання і контролю, а також за їх відсутності;
- розпалювати установки без їх попередньої продувки; подавати паливо, коли газові пальники або форсунки згасли;
- експлуатувати установки у випадку витікання газу із системи паливоподачі або підтікання рідкого палива;
- допускати до роботи працівників, які не пройшли навчання з пожежнотехнічного мінімуму, не отримали відповідних посвідчень кваліфікації.

Забороняється вносити зміни до елементів системи кондиціонування, вентиляції повітря і опалення, які перешкоджають поширенню пожежі. Не дозволяється робота технологічного обладнання у пожежонебезпечних та вибухопожежонебезпечних приміщеннях при відключених або несправних сухих фільтрах, гідрофільтрах, пиловловлювальних, пиловсмоктувальних та інших приладах вентиляційних систем.

При розміщенні вибухозахищених вентиляторів за межами приміщень для них необхідно влаштовувати спеціальне укриття з негорючих матеріалів. Під час експлуатації вентиляційних систем забороняється [40]:

- експлуатувати переповнені циклони;
- видаляти за допомогою однієї системи відсосів пил, пару, різні гази та інші речовини, які при змішуванні можуть викликати вибух, горіння або спалахи;

- складувати впритул (на відстані менше половини метра) до устаткування і повітроводів негорючі матеріали в горючій упаковці або горючі матеріали;
- застосовувати припливно-витяжні повітроводи й канали для відведення газів від газових колонок, приладів опалення та інших нагрівальних приладів;
- залишати двері вентиляційних камер відчиненими, зберігати в камерах різне устаткування та матеріали;
- закривати решітки, отвори і витяжні канали.

Автономні моноблочні кондиціонери, а також автономні кондиціонери роздільного типу можна розміщувати у будівлях та приміщеннях різного призначення, крім приміщень, у яких не допускається рециркуляція. Зовнішні блоки автономних кондиціонерів роздільного типу потужністю по холоду до 12 кВт допускається розміщувати у критих переходах, відкритих сходових клітках, на незасклених лоджіях.

Холодильне обладнання з аміаковмісним холодоагентом можна використовувати при реконструкції для холодопостачання систем кондиціонування виробничих приміщень, розміщуючи обладнання в окремих прибудовах, будинках або окремих приміщеннях одноповерхових виробничих будинків.

Під час експлуатації калориферів потрібно дотримуватися таких вимог [40]:

- слідкувати за тим, щоб транзитні канали, через які подається нагріте в калорифері повітря, не мали отворів, крім каналів, призначених для подавання повітря у приміщення;
- систематично проводити гідравлічним або пневматичним способом очищення калориферів від забруднень;
- не допускати виникнення зазорів між калориферами, а також між будівельними і калориферними конструкціями камер, а виявлені зазори зашпаровувати негорючими матеріалами;
- тримати в справному стані контрольно-вимірювальні прилади;

– відстань між конструкціями з важкогорючих та горючих матеріалів і калориферами повинна бути не меншою за півтора метра за наявності електричного або вогневого підігріву і не меншою за 0,1 метр, коли теплоносієм є пара або вода.

Прокладання, підключення, монтаж мереж, влаштування електричного захисту на лініях, які живлять побутові кондиціонери, повинні виконуватись відповідно до вимог інструкції виробника. Лінії живлення до кожного побутового кондиціонера чи групи кондиціонерів потрібно забезпечувати автономним пристроєм електричного захисту незалежно від наявності захисту на загальній лінії, яка відповідає за живлення групи кондиціонерів.

Під час експлуатації побутових кондиціонерів заборонено [40]:

- перетинати протипожежні перешкоди інженерними системами кондиціонера без влаштування проходок, які відповідають нормованій межі вогнестійкості протипожежної перешкоди;
- замінювати наявні триполюсні штепсельні роз'єднувачі на двополюсні;
- вносити в конструкцію кондиціонерів зміни, не передбачені заводом виробником.

Було розглянуто вимоги безпеки праці під час експлуатації систем вентиляції, опалення та кондиціонування повітря, а також розробку заходів захисту від пожеж та вибухів в цих системах. Оскільки в контексті виконання кваліфікаційної роботи, присвяченої розробці веб-інтерфейсу інформаційної системи спостереження внутрішнього середовища «розумних приміщень», це важливо тому, що забезпечення безпечних умов праці є критичним для ефективного та безперебійного функціонування таких систем.

ВИСНОВКИ

У межах кваліфікаційної роботи реалізовано повноцінний цикл розробки веб-додатку для моніторингу та керування параметрами внутрішнього середовища в розумному будинку на основі технологій Інтернету речей (IoT). Проведено глибокий аналіз сучасних архітектур IoT-систем та веб-інтерфейсів, на основі чого обґрунтовано вибір технологічного стека: React для клієнтської частини, Node.js з Express для серверної логіки, та MySQL як системи керування базами даних.

У роботі реалізовано трирівневу структуру веб-застосунку з підтримкою автентифікації користувачів, ролей, персоналізованого доступу до функціоналу та візуалізації сенсорних даних у режимі реального часу. Побудована база даних підтримує зберігання як поточних, так і історичних даних, що дає змогу аналізувати зміни параметрів середовища та формувати рекомендації для керування системами вентиляції чи кондиціонування. Запропонована структура ролей (гостьовий режим, користувач, адміністратор, модератор, власник пристроїв) забезпечує гнучке керування доступом та налаштуванням пристроїв.

Система протестована на узагальненій моделі Smart Home, що підтвердило її функціональність, масштабованість і придатність до впровадження у побутовому або комерційному середовищі. Проєкт демонструє можливість створення ефективного, зручного та безпечного засобу для керування параметрами середовища в рамках концепції «розумного будинку» та «розумного міста».

Отримані результати мають як прикладне значення — для подальшої реалізації в реальних умовах, так і наукову цінність — як основа для розвитку розподілених систем моніторингу та управління на основі IoT.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Persaud T., et al. Smart city innovations to improve quality of life in urban settings : матеріали IEEE Global Humanitarian Technology Conference (GHTC). – IEEE, 2020.
2. Včelák J., et al. Smart building monitoring from structure to indoor environment : матеріали Smart City Symposium Prague (SCSP). – IEEE, 2017.
3. Hsu H.-H., Chang C.-Y., Hsu C.-H., ред. Big data analytics for sensor-network collected intelligence. – Morgan Kaufmann, 2017.
4. Okonta D. E., Vukovic V. Smart Cities Software Applications for Sustainability and Resilience // Heliyon. – 2024.
5. Sethi P., Sarangi S. R. Internet of things: architectures, protocols, and applications // Journal of Electrical and Computer Engineering. – 2017. – № 1. – Article ID 9324035.
6. Marques G., et al. Indoor air quality monitoring systems for enhanced living environments: A review toward sustainable smart cities // Sustainability. – 2020. – T. 12, № 10. – C. 4024.
7. Narayana T. L., et al. Advances in real time smart monitoring of environmental parameters using IoT and sensors // Heliyon. – 2024. – T. 10, № 7.
8. Gracias J. S., et al. Smart Cities–A Structured Literature Review // Smart Cities. – 2023. – T. 6, № 4. – C. 1719–1743.
9. Addas A. The concept of smart cities: a sustainability aspect for future urban development based on different cities // Frontiers in Environmental Science. – 2023.
10. Mohanty R., Kumar B. P. Urbanization and smart cities // Solving Urban Infrastructure Problems Using Smart City Technologies. – Elsevier, 2021. – C. 143–158.
11. Rai H. M., et al. Use of Internet of Things in the context of execution of smart city applications: a review // Discover Internet of Things. – 2023. – T. 3, № 1. – Стаття 8.

12. Shin Hwang. What is a Smart City? Definition and 6 Key Features [Електронний ресурс]. – 2024. – Режим доступу: <https://letmewp.com/what-is-a-smart-city-definition-and-6-key-features> (дата звернення: 9.06.2024).
13. Hopkins T., et al. User interfaces in smart cities // Handbook of Smart Cities. – 2020. – С. 1–33.
14. Wang C., et al. Economic and environmental impacts of energy efficiency over smart cities and regulatory measures using a smart technological solution // Sustainable Energy Technologies and Assessments. – 2021. – Т. 47. – Article ID 101422.
15. Hernaningsih T., et al. Application of the concept of smart city and smart water management for the new capital city // IOP Conference Series: Earth and Environmental Science. – 2023. – Т. 1201, № 1.
16. Стручок В. С., Стручок О. С., Мудра Д. В. Навчальний посібник до написання розділу дипломного проекту та дипломної роботи «Безпека в надзвичайних ситуаціях» для студентів всіх спец. денної, заочної (дистанційної) та екстернатної форм навчання. – Тернопіль, 2017. – 112 с.
17. Стручок В. С. Безпека в надзвичайних ситуаціях: методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання. – Тернопіль : ФОП Паляниця В. А., 2022. – 156 с.
18. Бедрій Я. І. Безпека життєдіяльності : навч. посіб. – Київ : Кондор, 2009. – 284 с.
19. ДБН В.2.5-67:2013. Опалення, вентиляція та кондиціонування / Інститут «УкрНДІспецбуд». – 2013. – [Електронний ресурс]. – URL: <https://dbn.co.ua/load/normativy/dbn/1-1-0-1018> (дата звернення: 10.06.2024).
20. Желібо Є. П., Заверуха Н. М., Зацарний В. В. Безпека життєдіяльності : навчальний посібник. – Київ : Каравела, 2004. – 328 с.
21. Наказ КМУ про затвердження Правил пожежної безпеки в Україні № 1417 від 30.12.2014.