Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____*Комп'ютерних наук*_____
(повна назва)
Кафедра_____*Системотехніки*_____

(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

Застосування технологій Інтернету речей для управління показниками середовища в приміщеннях
(тема)

Виконав:
студент _2__ курсу, групи _*СПРм-22-1*
_____*Хоменко І. С.*_____
(прізвище, ініціали)

Спеціальність *122 Комп'ютерні науки* __
(код і повна назва спеціальності)
Тип програми __освітньо-наукова_____
Освітня програма _«*Системне проєктування* _____»
_____
(повна назва освітньої програми)

Керівник _____*проф. Міщеряков Ю. Р.*_____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ __*Гребеннік І.В.*__
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*

Кафедра *Системотехніки*

Рівень вищої освіти другий (магістерський)

Спеціальність *122 Комп'ютерні науки*
(код і повна назва)

Тип програми освітньо-наукова

Освітня програма «*Системне проєктування»*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«___»_____20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Хоменко Іллі Сергійовичу*

(прізвище, ім'я, по батькові)

1. Тема роботи: *Застосування технологій Інтернету речей для управління показниками середовища в приміщеннях.*
затверджена наказом по університету від *01.04.* 2024 р. № *259Ст*
2. Термін подання студентом роботи до екзаменаційної комісії: 22.06.2024 р
3.Вихідні дані до роботи: *Розробити компоненти системи Інтернету речей для відстеження показників в приміщеннях. Описати і впровадити алгоритм управління показниками для створення сприятливих умов в приміщеннях.*

4. Перелік питань, що потрібно опрацювати в роботі: *Вступ 1Аналіз предметної області та структури інформаційної системи 1.1 Аналіз предметної галузі 1.2 Аналіз існуючих програмних рішень 1.3 Аналіз існуючих програмних рішень 1.4 Постановка мети і задач кваліфікаційної роботи 2 Розробка вимог до інформаційної системи 2.1 Визначення функціональних вимог до інформаційної системи 2.2 Вибір засобів розробки та архітектури 2.3 Опис серверної архітектури застосунку 3 Розробка та реалізація системи 3.1 Опис головних бізнес-процесів розроблюваного веб-застосунку 3.2 Проєктування формату даних в системі 3.3 Розробка візуального інтерфейсу застосунку 3.4 Розробка серверної частини застосунку 3.5 Розробка основного модулю управління середовищем 4 Методологія роботи з*

даними в системі 4.1 Огляд параметрів середовища 4.2 Методології збору даних 4.3 Збір та аналіз даних в системі

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій: _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Отримання завдання на виконання роботи*	*11.03.2024*	
2	*Формування бізнес-вимог та бізнес ризиків*	*11.03.2024-28.03.2024*	
3	*Дослідження актуальності теми та аналогів*	*28.03.2024 – 9.04.2024*	
4	*Розробка дизайну*	*9.04.2024 – 22.04.2024*	
5	*Вибір технологій та інтеграцій*	*22.04.2024 – 3.05.2024*	
6	*Розробка серверної частини застосунку*	*3.05.2024 – 11.05.2024*	
7	*Розробка клієнтської частини*	*11.05.24 – 15.05.24*	
8	*Написання кваліфікаційної роботи*	*15.05.2024 - 15.06.2024*	
9	*Представлення кваліфікаційної роботи в ДЕК*	*22.06.2024*	

Дата видачі завдання 11.03.2024 р.

Студент _____ Хоменко І. С.
(підпис)

Керівник роботи _____ проф Міщеряков Ю. Р.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Кваліфікаційна робота: 89 арк., 33 рис., 25 джерел інформації.

ІНФОРМАЦІЙНА СИСТЕМА, ЖИТТЄВИЙ ЦИКЛ, IoT, МОНІТОРИНГ, HVAC, ПРОГНОЗУВАННЯ, USE CASE, CLASS DIAGRAM

Об'єкт досліджень – системи моніторингу навколишнього середовища на базі Інтернету речей для приміщень.

Предмет дослідження – моніторинг та контроль ключових показників навколишнього середовища за допомогою Інтернету речей.

Методами досліджень є розгортання мережі контролерів IoT, побудова конвеєрів обробки даних, інтеграція з інтелектуальними системами управління. Дослідницькі стратегії з використанням Інтернету речей для моніторингу екологічних показників включають вивчення та аналіз наукової літератури, технічних документів і теоретичних статей, пов’язаних з Інтернетом речей, бездротовими сенсорними мережами, екологічним моніторингом і системами контролю.

Дослідження охоплює моделювання та проєктування системи на основі Інтернету речей для збору даних, обробки та візуалізації екологічних показників.

Експериментальна фаза включає використання імітації датчиків IoT для збору даних про стан навколишнього середовища в реальному сценарії.

ABSTRACT

Explanatory note to the certification work of the master's degree contains: 89 pages, 33 pictures, 25 sources of information and 3 additional sections.

INFORMATION SYSTEM, LIFE CYCLE, IoT, MONITORING, HVAC, FORECASTING, USE CASE, CLASS DIAGRAM

The object of research is environmental monitoring systems based on the Internet of Things for premises.

The subject of the research is monitoring and control of key environmental indicators using the Internet of Things.

Research methods include the deployment of a network of IoT controllers, the construction of data processing pipelines, and integration with intelligent control systems. Research strategies using the Internet of Things for environmental monitoring include the study and analysis of scientific literature, technical papers and theoretical articles related to the Internet of Things, wireless sensor networks, environmental monitoring and control systems.

The research covers the modeling and design of an IoT-based system architecture for data collection, processing and visualization of environmental indicators.

The experimental phase involves the use of simulated IoT sensors to collect data about the state of the environment in a real-world scenario.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – База даних;

ІС – інформаційна система;

CASE – computer-aided software engineering;

DFD – Data Flow Diagram;

ERD – Entity-Relation Diagram;

IoT – інтернет речей;

ID – ідентифікатор;

IDEF0 – Integration Definition for Function Modeling;

UML – Unified Modeling Language.

ВСТУП

Підключення фізичних об’єктів, датчиків і систем через Інтернет стало трансформаційною технологією для забезпечення обміну даними та дистанційного зондування.

Підтримка сприятливих умов навколишнього середовища має вирішальне значення для здоров’я мешканців, продуктивності та ефективності. Традиційні підходи до моніторингу навколишнього середовища часто покладаються на збір даних вручну, що може зайняти багато часу, трудомісткості та бути схильним до людських помилок.

Технологія Інтернету речей забезпечує ефективний і дієвий підхід до реальної роботи, за допомогою якої параметри навколишнього середовища, такі як температура, вологість, якість повітря, освітлення та контролюється рівень шуму.

Пропонована система IoT складається з сенсорних вузлів, стратегічно розміщених по всьому кампусу. Вони оснащені датчиками, які можуть вимірювати дані про навколишнє середовище, доставляти їх до центральних шлюзів або хмарних платформ, а потім обробляти дані, аналізувати та візуалізувати дані за допомогою зручної інформаційної панелі, що дозволяє здійснювати моніторинг навколишнього середовища та стану навколишнього середовища в реальному часі.

Використовуючи технологію IoT, менеджери об’єктів і власники будівель можуть постійно контролювати й оптимізувати навколишнє середовище, забезпечувати відповідність нормативним стандартам і сприяти добробуту мешканців за допомогою попереджень і сповіщень, що забезпечують автоматичні відповіді, коли сигнали навколишнього середовища відхиляються від заздалегідь визначених порогів, щоб забезпечити швидку корекцію.

Крім того, переваги зібраних даних можна використовувати для прогнозної аналітики та моделей машинного навчання, полегшуючи визначення пріоритетів обслуговування та розподіл ресурсів. Наприклад, прогностичні

моделі можуть відстежувати потенційні події, такі як збій системи кондиціонування або погіршення якості повітря, щоб забезпечити своєчасне втручання та мінімізувати збої.

Інтеграція технології IoT у системи управління будівлями та платформи автоматизації розумного будинку/офісу може ще більше покращити використання екологічних показників. Інтелектуальні засоби керування можуть регулювати системи опалення, вентиляції, кондиціонування повітря та освітлення на основі даних у реальному часі, підвищуючи енергоефективність і знижуючи експлуатаційні витрати.

Таким чином, використання технології Інтернету речей для керування показниками навколишнього середовища пропонує багато переваг, зокрема покращену точність моніторингу, рішення на основі даних, комфорт мешканців, енергоефективність та економію коштів. Це відкриває шлях до розумніших, стійкіших та ефективніших громад.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СТРУКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1 Аналіз предметної галузі

Тема охоплює використання технологій Інтернету речей (IoT) для моніторингу, аналізу та керування різними середовищами в будівлях, офісах, будинках та інших середовищах. Ключові показники середовища, які можна враховувати, включають багато показників описаних далі.

Температура: контрольовані рівні температури важливі для комфорту мешканців та енергоефективності. Датчики IoT можуть відстежувати коливання температури, забезпечуючи автоматичне перемикання нагрівання/охолодження.

Вологість: Занадто велика вологість може призвести до розвитку цвілі та інших проблем, а занадто низька вологість може призвести до депресії. Датчики вологості IoT можуть оптимізувати рівень вологості.

Якість повітря: погана якість повітря в приміщенні через забруднюючі речовини, леткі органічні сполуки (ЛОС) або вакуум може негативно вплинути на здоров’я. Підключені до Інтернету речей пристрої контролю якості повітря можуть виявляти проблеми та запускати системи вентиляції.

Освітлення: оптимальні умови освітлення не тільки підвищують комфорт, але також впливають на продуктивність і споживання енергії. Системи керування освітленням IoT можуть автоматично регулювати освітлення залежно від заповнюваності та рівня природного освітлення.

Типи шуму: Надмірний шум від приладів, машин або надворі може відволікати увагу та шкодити самопочуттю. Датчики шуму IoT можуть ідентифікувати проблемні зони, які можна пом’якшити.

Енергоспоживання: технологія IoT може контролювати споживання енергії, забезпечуючи оптимізацію, яка зменшує відходи та знижує експлуатаційні витрати.

Реалізація IoT передбачає збір даних у реальному часі про ці середовища за допомогою різноманітних датчиків і пристроїв у кампусі, а потім передачу цих даних на центральний шлюз або хмарну платформу для обробки.

Передову аналітику, включаючи машинне навчання та штучний інтелект, можна застосовувати до даних Інтернету речей, щоб ідентифікувати закономірності, виявити аномалії та отримати інформацію для прийняття рішень. Прогнозні моделі можуть керувати інформаційним прогнозуванням можливостей, забезпечуючи захист і втручання.

Проаналізовані дані можна інтегрувати з системами управління будівлями (BMS) і платформами автоматизації розумного дому/офісу, що дозволяє контролювати такі системи, як HVAC, освітлення та вентиляція на основі факторів навколишнього середовища в реальному часі за допомогою об’єкта та людей, які мешкають в даній будівлі.

Крім того, технологія IoT може забезпечити віддалений моніторинг і контроль, може надати менеджерам об’єктів і власникам будівель доступ до інформації в реальному часі та вносити зміни з будь-якого місця, а також підвищити ефективність роботи.

Успішне впровадження IoT у цій сфері потребує мультидисциплінарного підходу, який спеціалізується на сенсорних технологіях, бездротових мережах, аналітиці даних, системах автоматизації, інженерії будівель тощо. Вирішення проблем, пов’язаних із безпекою даних, конфіденційністю, співпрацею та масштабованістю, також є важливим для ефективного впровадження.

Загалом, тема використання технологій Інтернету речей для моніторингу навколишнього середовища в університетах має величезний потенціал для підвищення комфорту мешканців, енергоефективності та підвищення стійкості та економії експлуатаційних витрат у різних будівлях і об’єктах.

1.2 Проблематика предметної галузі та перелік ключових процесів

Тема використання технологій Інтернету речей (IoT) для вирішення екологічних показників стикається з кількома проблемами. Ключовим питанням є розташування та охоплення датчиків, що включає визначення належного розміщення датчиків для забезпечення належного моніторингу, який усуває перешкоди, обмеження потужності тощо. Іншим питанням є збір і передача даних, де вибрати відповідний протокол бездротового зв’язку, керувати трафіком даних і здійснювати безпечну та надійну передачу даних.

Управління та зберігання даних є серйозними проблемами, зокрема робота з великими обсягами даних датчиків у режимі реального часу, забезпечення цілісності даних, конфіденційності та безпеки, а також забезпечення масштабованих рішень для зберігання та ефективність у використанні. Потрібні вдосконалення, аналогічно машинне навчання та штучне використовуються методи розвідки, тому що розпізнавання образів і виявлення аномалій необхідні для отримання корисної інформації з даних про навколишнє середовище в сильному.

Інтеграція та сумісність також є ключовими питаннями, включаючи інтеграцію систем Інтернету речей з існуючими системами управління будівлями (BMS) і платформами автоматизації для забезпечення можливостей безперебійного керування обміном даними для подолання проблем з підключенням Існує обробка між системами та протоколами Інтерфейс користувача та візуалізація є важливими, включаючи інтуїтивно зрозумілі інформаційні панелі для візуалізації даних. Необхідні інтерфейси для моніторингу в реальному часі та можливостей віддаленого доступу, а також для настроюваних оповіщень і звітів.

Автоматизоване керування та оптимізація є важливими стратегіями, включаючи використання інтелектуальних алгоритмів керування системами HVAC, освітлення та вентиляції, енергоефективність на основі умов навколишнього середовища та мешканців системи, вирішення питань

конфіденційності та безпеки, що дозволяє проводити прогнозне обслуговування та проактивне втручання. Забезпечуйте конфіденційність, захищайте конфіденційну інформацію, застосовуйте надійні заходи безпеки проти кіберзагроз і несанкціонованого доступу, а також подолайте потенційні вразливості в пристроях Інтернету речей і системах зв'язку.

Масштабованість і технічне обслуговування також є важливими міркуваннями, вимагаючи від систем Інтернету речей масштабування зі збільшенням розгортання датчиків, використання оновлень по повітрю (OTA) і можливостей віддаленого керування, а також забезпечення доступності, підтримки та встановлення поточних оновлень системи.

Основними застосуваннями технологій IoT для моніторингу навколишнього середовища є збір і аналіз вимог, проєктування та конфігурація сенсорної мережі, проєктування та встановлення датчиків, системи бездротового зв'язку, збір і передача даних, збір і зберігання даних, попередня обробка та підготовка даних, аналіз даних і розробка моделі, виявлення аномалій та вибір шаблонів, із системами побудови (BMS).

Інтеграція, розробка користувальницького інтерфейсу та інформаційної панелі, автоматизовані алгоритми контролю та оптимізації, керування звітами про сповіщення, реалізація політики безпеки та конфіденційності, прогнозне технічне обслуговування та проактивне планування втручання, масштабованість та ефективність, тестування та перевірка системи, розгортання та впровадження, нагляд та обслуговування, система оновлення та оновлення, навчання та підтримка користувачів, відповідність та звітність.

1.3 Аналіз існуючих програмних рішень

Програмні рішення, спрямовані на моніторинг навколишнього середовища на основі Інтернету речей і керування будівлями та об'єктами, зросли в геометричній прогресії за останні роки.

Ці рішення спрямовані на використання технологій Інтернету речей, сенсорних мереж, аналітики даних і можливостей автоматизації, які можна використовувати для оптимізації умов навколишнього середовища в приміщенні, підвищення комфорту мешканців і підвищення енергоефективності.

Ключовими гравцями в цьому просторі є великі компанії, такі як Schneider Electric, Siemens, Honeywell, IBM тощо, які надають передову платформу IoT і системи автоматизації будівель.

Ці рішення зазвичай реалізуються через інтеграцію датчиків. інформаційні панелі візуалізації даних, розширена аналітика, прогнозне технічне обслуговування, системи управління будівлею та забезпечують такі функції, як інтеграція з (BMS).

EcoStructure Building від Schneider Electric, Desigo CC від Siemens і платформа Forge Analytics від Honeywell є прикладами надійних комерційних пропозицій, які дозволяють у режимі реального часу відслідковувати такі параметри навколишнього середовища, як температура, вологість, якість повітря та енергоспоживання.

Окрім цих лідерів галузі, доступно багато інших ключових рішень від таких компаній, як Tridium (Niagara Framework), CIT (Commander IoT Platform), Optergy (Proton Building Analytics), Disruptive Technologies (Sensormetric IoT Platform), BuildingIQ.

Ці рішення зазвичай націлені на конкретні випадки використання або сегменти ринку, такі як комерційні будівлі, промислові будівлі або житлові розумні будинки.

Що стосується відкритого коду, то такі платформи, як Eclipse IoT, OpenHAB, Home Assistant і Domoticz, пропонують гнучкі та масштабовані способи створення систем моніторингу навколишнього середовища на основі Інтернету речей, хоча впровадження цих рішень із відкритим кодом і гібридизація можуть вимагати значного технічного досвіду і ресурсів.

Незалежно від конкретного рішення, ці програмні платформи зазвичай націлені на надання комплексних інструментів для інтеграції датчиків, збору

даних, обробки та візуалізації, а також автоматичного керування та адаптації до частих віддалених, оперативних та існуючих систем, а також пріоритетних функцій, таких як сувора безпека заходи щодо забезпечення конфіденційності даних і захисту від кіберзагроз.

Schneider Electric EcoStruxure Building: EcoStruxure – це комплексна платформа Інтернету речей, яка пропонує можливості управління будівлями, енергоменеджменту та моніторингу навколишнього середовища. Він інтегрується з різними будівельними системами та пристроями IoT, дозволяючи в режимі реального часу відстежувати температуру, вологість, якість повітря та споживання енергії.

Платформа забезпечує аналітику даних, прогнозне технічне обслуговування та функції автоматизованого керування для оптимізації продуктивності та енергоефективності будівлі. Візуальний інтерфейс застосунку зображено на рисунку 1.1.

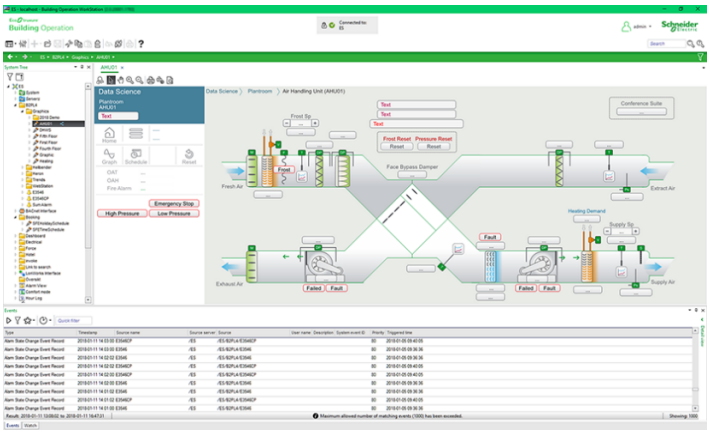

Рисунок 1.1 – інтерфейс застосунку Schneider Electric EcoStruxure Building

Система автоматизації будівель Siemens Desigo CC: Desigo CC – це система автоматизації будівель, яка інтегрується з пристроями та датчиками Інтернету речей для моніторингу та контролю різноманітних параметрів навколишнього середовища, включаючи температуру, якість повітря та освітлення.[4]

Візуальний інтерфейс застосунку зображено на рисунку 1.2. Він пропонує візуалізацію даних, управління енергією та можливості віддаленого доступу. Desigo CC також містить функції для виявлення несправностей, прогнозованого обслуговування та бездоганної інтеграції з іншими системами будівлі.

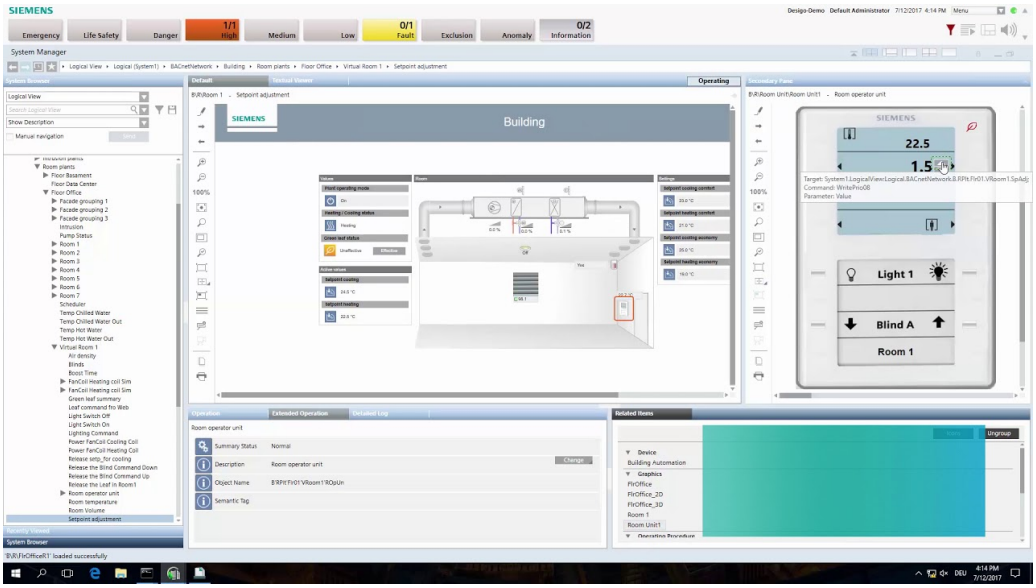

Рисунок 1.2 – інтерфейс застосунку Siemens Desigo CC Building Automation System

Платформа Honeywell Forge Analytics: Платформа Honeywell Forge Analytics – це рішення на основі Інтернету речей для моніторингу та оптимізації продуктивності будівлі. Він збирає дані з різних датчиків і систем, включаючи датчики навколишнього середовища для температури, вологості та якості повітря. Візуальний інтерфейс застосунку зображено на рисунку 1.3.

Однією з ключових сильних сторін Forge Analytics є розширена аналітика та можливості машинного навчання. Він використовує складні алгоритми та прогностичну аналітику для визначення закономірностей, виявлення аномалій та отримання корисних ідей на основі даних про навколишнє середовище.

Цю інформацію можна використовувати для прогнозованого технічного обслуговування, дозволяючи платформі передбачати збої обладнання або зниження продуктивності до їх виникнення, мінімізуючи час простою та знижуючи витрати на обслуговування.

Крім того, Forge Analytics пропонує можливості виявлення несправностей і діагностики, допомагаючи швидко виявляти та усувати проблеми, що впливають на якість середовища в приміщенні. Цей проактивний підхід гарантує, що умови навколишнього середовища залишаються в прийнятних межах, сприяючи комфорту та добробуту пасажирів.

Платформа забезпечує розширену аналітику, прогнозне моделювання та можливості автоматизованого керування для оптимізації енергоспоживання, рівня комфорту та ефективності роботи.

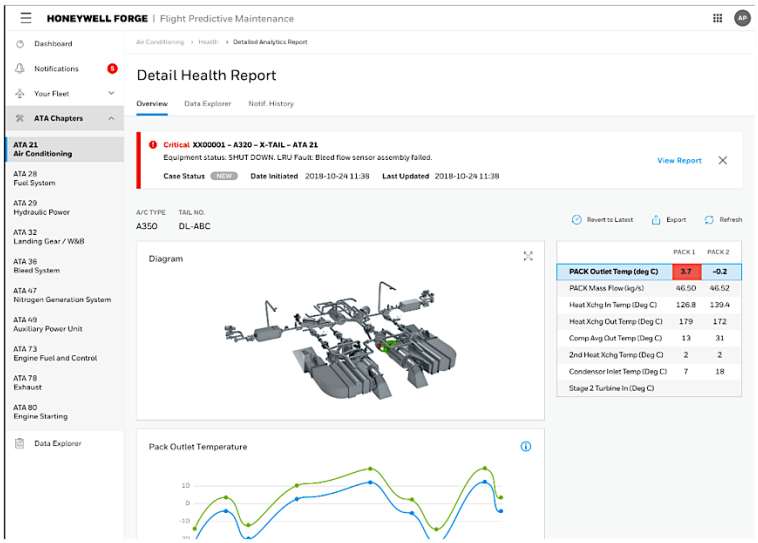

Рисунок 1.3 – інтерфейс застосунку IBM Watson IoT Platform

IBM Watson IoT Platform: IBM Watson IoT Platform – це комплексний набір інструментів і послуг для створення та розгортання рішень IoT.

Його можна інтегрувати з різними датчиками навколишнього середовища та системами управління будівлями, що дозволяє здійснювати моніторинг умов навколишнього середовища в реальному часі.

Візуальний інтерфейс застосунку зображено на рисунку 1.4. Платформа пропонує аналіз даних, можливості машинного навчання та інтеграцію з іншими службами IBM, такими як Watson Analytics і Watson Studio.

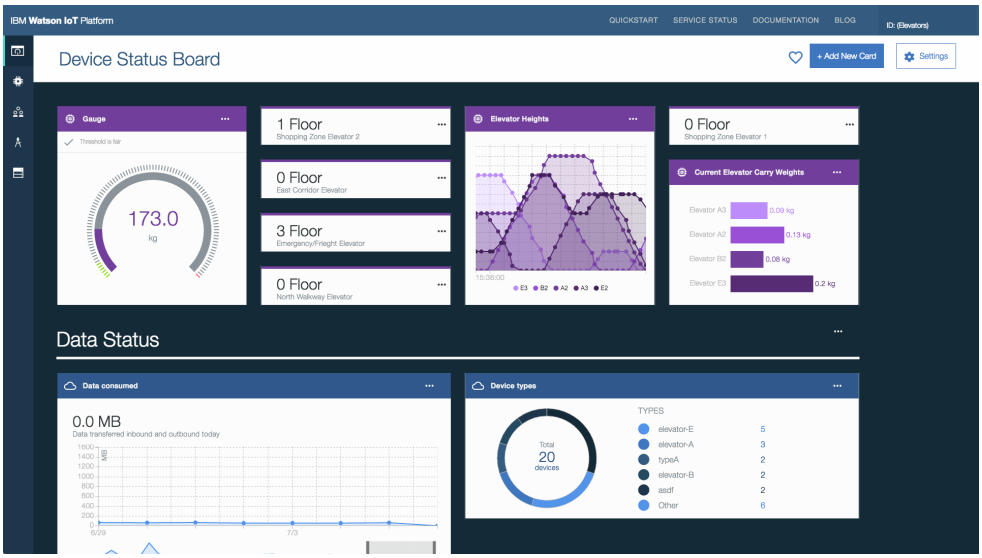

Рисунок 1.4 – інтерфейс застосунку Honeywell Forge Analytics Platform

Forge Analytics пропонує потужні можливості керування даними, включаючи безпечне прийом, зберігання та обробку даних. Він може обробляти великі обсяги даних із багатьох джерел, забезпечуючи цілісність даних, аналізуючи історичні дані та ідентифікуючи тенденції.

Платформа надає комплексний набір інструментів візуалізації даних, включаючи настроювані інформаційні панелі, діаграми та звіти, що дозволяє користувачам легко відстежувати та аналізувати екологічні дані в режимі реального часу. Ці візуалізації можна адаптувати до конкретних ролей і вподобань користувачів, забезпечуючи легкий доступ до відповідної інформації для різних зацікавлених сторін.

1.4 Постановка мети і задач кваліфікаційної роботи

Розробка дослідження полягає в аналізі можливостей інтеграції технологій Інтернету речей (IoT) та моделей впровадження для оптимізації управління робочими просторами в офісних приміщеннях та робочих організаціях.

Головною метою є розробка рекомендацій та практичних рішень для поліпшення комфорту працівників, зниження витрат та ефективного використання офісного простору.

У рамках дослідження передбачено аналіз поточного стану робочих просторів, вивчення можливостей використання сенсорів та IoT-пристроїв для збору даних, розробку аналітичних моделей для прогнозування та оптимізації умов робочого середовища, а також визначення шляхів автоматизації та покращення управління робочим простором.

Дослідження передбачає глибокий аналіз сучасних робочих просторів та виявлення проблем, які можуть бути вирішені шляхом інтеграції технологій Інтернету речей та моделей штучного інтелекту. Це включає в себе оцінку того, як офісні приміщення використовуються в повсякденній роботі, які аспекти можуть бути поліпшені, і які витрати можуть бути зменшені або ефективно використані.

Дослідження також включає вивчення можливостей використання сенсорів та IoT-пристроїв для збору різноманітних даних про робоче середовище. Ці дані можуть включати інформацію про температуру, вологість, освітлення, якість повітря, рух та інші параметри, які впливають на комфорт та продуктивність працівників.

Далі, дослідження передбачає розробку аналітичних моделей, які базуються на зібраних даних. Ці моделі можуть використовувати методи штучного інтелекту та машинного навчання для виявлення залежностей, аналізу та прогнозування патернів в робочому середовищі. Таким

чином, можна передбачити зміни та вжити заходи для оптимізації умов працівників.

Крім цього, дослідження визначає шляхи автоматизації та покращення управління робочим простором. Це може включати в себе автоматичне регулювання параметрів робочого середовища, планування використання офісного простору, а також впровадження систем безпеки та контролю доступу.

Загалом, дослідження спрямоване на розробку комплексного підходу до оптимізації робочого простору за допомогою IoT та ШІ з метою покращення умов праці та зменшення витрат.

2 РОЗРОБКА ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Визначення функціональних вимог до інформаційної системи

Розроблений програмний продукт матиме певні мінімальні системні вимоги, що будуть гарантувати правильну роботу програмного продукту. Серед цих вимог є наступні: необхідно мати встановлений .NET SDK версії не нижче 6.

Система має інтегруватися з різними датчиками навколишнього середовища, які зазвичай використовуються для моніторингу кімнати. Ці датчики повинні містити такі параметри, як температура, вологість, якість повітря, рівень вуглекислого газу, інтенсивність світла та рівень шуму. Для полегшення інтеграції слід забезпечити сумісність з існуючими протоколами датчиків.

Збір і обробка даних у режимі реального часу є важливими для програми. Інтегровані датчики повинні бути здатними до безперервного збору даних, гарантуючи, що інформація, яка надається користувачам, завжди актуальна та відображає умови навколишнього середовища, що переважають на місці. Обробка цієї інформації має бути швидкою та ефективною, дозволяючи швидко розслідувати та вживати заходів щодо виявлених змін або аномалій.

Функції попередження та оповіщення важливі для своєчасного реагування на нові умови або відхилення від заздалегідь визначених граничних значень середовища. Система повинна дозволяти користувачам встановлювати власні вимоги для кожного середовища та запускати сповіщення або попередження щоразу, коли ці обмеження перевищуються. Ці сповіщення мають поширюватися кількома каналами, включаючи, але не обмежуючись, електронною поштою, SMS і push-повідомленнями, щоб гарантувати, що користувачі можуть негайно вирішити виявлені проблеми.

Можливості віддаленого аналізу та контролю є важливими для підвищення ефективності роботи та оперативності. Користувачі можуть отримати доступ до програми віддалено через зручний інтерфейс, через веб-портал або мобільну

програму. Цей зв’язок має забезпечувати повне уявлення про поточне середовище на сайті, дозволяючи користувачам відстежувати тенденції та, якщо необхідно, вживати дії із новими планами, де це можливо.

Ведення детального історичного журналу має важливе значення для полегшення аналізу тенденцій, бізнес-аналізу та звітності про відповідність. Система повинна зберігати історичні дані про показники навколишнього середовища, гарантувати, що користувачі можуть отримати доступ до попередніх записів і створювати звіти за потреби. Ця історична інформація повинна бути відформатована та представлена у зручному для користувача форматі, що дозволить користувачам отримати уявлення та прийняти обґрунтовані рішення щодо успішного управління кампусом та екологічного забезпечення.

Загалом застосунок повинен забезпечувати надійний і орієнтований на користувача спосіб моніторингу показників навколишнього середовища в університетському містечку, забезпечуючи аналіз у реальному часі, можливості оповіщення, надає можливості користувача та дистанційного моніторингу.

Сам програмний продукт повинен мати 3 типу користувачів:

а) Авторизовані користувачі:

- мешканці: особи, які працюють, живуть або постійно проживають у приміщеннях, де розгорнуто додаток IoT. Вони можуть включати працівників, мешканців, студентів або відвідувачів;

- менеджери об’єктів: персонал, відповідальний за нагляд за повсякденними операціями та обслуговуванням приміщень. Вони використовують додаток IoT для моніторингу умов навколишнього середовища, вирішення питань технічного обслуговування та забезпечення комфорту та безпеки мешканців;

- обслуговуючий персонал: техніки та обслуговуючий персонал, яким доручено проводити ремонт, обслуговувати обладнання та забезпечувати належне функціонування систем контролю навколишнього середовища на основі даних, наданих програмою IoT;

- персонал служби безпеки: охоронці або офіцери, які використовують програму IoT для моніторингу показників навколишнього середовища з метою безпеки, наприклад виявлення несанкціонованого доступу, моніторинг зон обмеженого доступу або визначення загроз безпеці;

- персонал із охорони навколишнього середовища та безпеки (EHS): Професіонали, відповідальні за забезпечення відповідності екологічним нормам, стандартам охорони здоров'я та протоколам безпеки в приміщеннях. Вони покладаються на додаток IoT для відстеження показників навколишнього середовища, виявлення потенційних ризиків і впровадження коригувальних заходів.

б) Неавторизовані користувачі:

- гості/відвідувачі: особи, які не мають офіційного дозволу на доступ до програми IoT, але можуть тимчасово перебувати в приміщенні. Їхній доступ до програми обмежено для підтримки безпеки та конфіденційності;

- неавторизований персонал: особи, які намагаються отримати неавторизований доступ до програми IoT без дійсних облікових даних або належної авторизації. Це можуть бути зловмисники, хакери або особи зі злими намірами.

в) Адміністратори:

- системні адміністратори: IT-персонал або призначені адміністратори, відповідальні за керування конфігурацією, обслуговуванням і безпекою програми IoT. Вони мають привілейований доступ до налаштувань системи, облікових записів користувачів і адміністративних функцій;

- адміністратори користувачів: адміністратори, яким доручено керувати обліковими записами користувачів, дозволами та правами доступу в програмі IoT. Вони контролюють автентифікацію користувачів, призначають ролі та дозволи, а також обробляють ініціалізацію та деініціалізацію облікового запису;

- Технічна підтримка/адміністративна підтримка: персонал, відповідальний за надання технічної допомоги, вирішення проблем і вирішення запитів користувачів, пов'язаних із додатком IoT. Вони можуть допомагати

користувачам керувати обліковими записами, навігацією системою та вирішувати технічні проблеми.

Для різних типів користувачів були виділені наступні функції:

Авторизовані користувачі — це фізичні або юридичні особи, яким надано дозвіл на доступ і використання програми IoT для керування екологічними показниками в приміщеннях;

Вони мають дійсні облікові дані та автентифіковані системою для доступу до певних функцій і можливостей на основі їхніх ролей і дозволів.

Авторизовані користувачі можуть включати менеджерів об’єктів, обслуговуючий персонал, призначений персонал і мешканців із затвердженими правами доступу;

Неавторизовані користувачі — це фізичні або юридичні особи, які не мають дійсних облікових даних або не отримали дозволу на доступ до програми IoT.

Вони можуть спробувати отримати доступ до системи без належної автентифікації чи авторизації, але їхні спроби доступу повинні бути відхилені заходами безпеки системи;

Серед неавторизованих користувачів можуть бути неавторизований персонал, зовнішні організації або особи без законних причин для доступу до програми;

Адміністратори — це привілейовані користувачі, відповідальні за керування та нагляд за роботою програми IoT.

Вони мають підвищені дозволи та права доступу, що дозволяє їм налаштовувати параметри системи, керувати обліковими записами користувачів і наглядати за загальною функціональністю програми.

Зазвичай адміністратори відповідають за керування користувачами, конфігурацію системи, усунення несправностей і забезпечення відповідності протоколам і нормам безпеки.

2.2 Вибір засобів розробки та архітектури

Під час розробки даного проєкту була використана мова C#, адже вона є дуже гнучкою для розробки, об’єктно-орієнтовність мови дозволяє дуже структуровано розробляти як веб-застосунки, так і локальні.

Для розробки веб-застосунку був використаний фреймворк ASP.NET MVC, що є стандартним в платформі та дозволяє розробляти різни види веб-застосунків, від простих односторінкових, до важких за своєю структурою наповнення застосунків.

Платформа ASP.NET MVC базується на взаємодіючих трьох компонентів: контроллера, модели і представлення. Контроллер приймає запитів, обробляє користувальницький ввід, взаємодіє з моделлю і представляє і повертає користувачем результат обробки запиту.[4]

Модель містить шар, що описує організацію даних в застосуванні. Представлення отримує дані з контролера і генерує елементи користувацького інтерфейсу для відображення інформації.

Для управління обміном і вставками кодів, представлених використовуваним движоком. До версії MVC 5 використовувалися два движки: Web Forms і Razor.

В MVC 5 єдиний движок, вбудований за умовчанням, є Razor. Движок WebForms використовує файли .aspx, а Razor - файли .cshtml і .vbhtml для зберігання уявлень. Також можливо і використання сторонніх движків. Файли представлені не є стандартними статичними сторонами з кодом html, а в процесі генерації контролерів з використанням представлених комбінацій в класах, які потім генеруються на сторінці html.[3]

Для доступу до даних був використаний EntityFramework.

ADO.NET Entity Framework (EF) - об'єктно-орієнтована технологія доступу до даних, є об'єктно-реляційне відображення (ORM) рішення для .NET Framework від Microsoft. Забезпечує можливість взаємодії з об'єктами як за допомогою LINQ у вигляді LINQ до Entities, так і з використанням Entity SQL.

Для обміну даними веб-рішень, що використовуються як ADO.NET Data Services (Асторія), так і зв'язок з Windows Communication Foundation і Windows Presentation Foundation, дозволяє створювати багатоваріантні застосунки, реалізуючи один з шаблонів проєктування MVC, MVP або MVVM.[8]

У застосунках ASP.NET MVC нерідко використовується патерн репозиторій для інкапсулювання логіки роботи з джерелами даних. І нерідко ми оперуємо безліччю сутностей і моделей, для управління якими створюється також безліч класів-репозиторіїв. Патерн Unit of Work дозволяє спростити роботу з різними репозиторіями і дає впевненість, що все репозиторії використовуватимуть один і той же контекст даних.

Одним з найбільш часто використовуваних патернів при роботі з даними є патерн 'Репозиторий'. Репозиторій дозволяє абстрагуватися від конкретних підключень до джерел даних, з якими працює програма, і є проміжною ланкою між класами, безпосередньо взаємодіють з даними, і рештою програми.

Архітектура цибулі базується на принципі інверсії управління. Цибульна архітектура складається з декількох концентричних шарів, що взаємодіють один з одним у напрямку до ядра, що представляє домен. Архітектура не залежить від рівня даних, як у класичних багаторівневих архітектурах, а від реальних моделей доменів.[11]

Нижче наведено переваги реалізації архітектури цибулі:

а) Пластини цибулі архітектури з'єднані через інтерфейси. Імплантації забезпечуються під час роботи.

б) Архітектура додатків побудована поверх моделі домену.

в) Вся зовнішня залежність, наприклад, доступ до бази даних і виклики сервісу, представлена у зовнішніх шарах.

г) Відсутність залежностей внутрішнього шару від зовнішніх шарів.

д) Гнучка і стійка портативна архітектура.

е) Немає необхідності створювати спільні проєкти.

є) Можна швидко перевірити, оскільки ядро програми не залежить ні від чого.

2.3 Опис серверної архітектури застосунку

Архітектура, показана на рисунку 2.2 являє собою складну систему, призначену для моніторингу та управління навколишнім середовищем через мережу взаємопов'язаних сервісів і компонентів, використовуючи як веб-інтерфейси, так і мікросервіси на стороні сервера для комплексного моніторингу та аналізу.

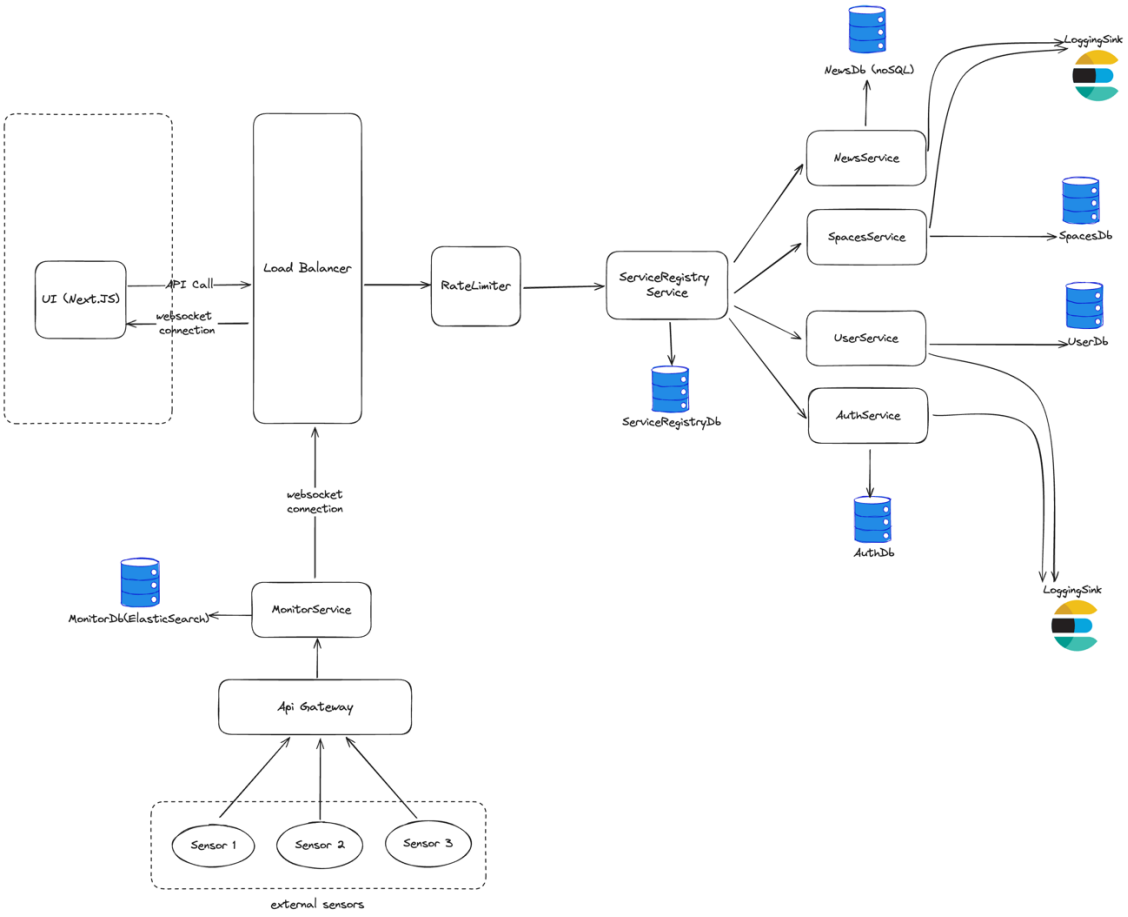

Рисунок 2.2 – Загальна діаграми архітектури системи

На базовому рівні система починається з інтерфейсу користувача. Цей інтерфейс ретельно розроблений з використанням React.js, щоб забезпечити безперебійну та інтуїтивно зрозумілу роботу. Користувачі можуть переміщатися по інтерфейсу, щоб відстежувати показники навколишнього середовища в режимі реального часу, керувати і отримувати доступ до аналізу історичних

даних. Взаємодія в режимі реального часу забезпечується за допомогою з'єднання WebSocket, яке забезпечує безперервний потік даних від серверної служби до інтерфейсу, гарантуючи, що інформація, яка відображається, завжди актуальна без необхідності оновлювати сторінку.

На рівні користувацького інтерфейсу система використовує користувацький інтерфейс на основі React.js, який підключається до серверних сервісів за допомогою викликів API та з'єднань WebSocket. Таке налаштування дозволяє користувацькому інтерфейсу динамічно оновлюватися в режимі реального часу, щоб кінцеві користувачі отримували негайний зворотній зв'язок та оновлення про середовище, яке вони контролюють. Інтерфейс користувача розроблений таким чином, щоб бути чуйним і зручним для користувача, надаючи інструменти та візуалізації, які дозволяють користувачам приймати обґрунтовані рішення на основі даних у реальному часі та історичних даних.

У центрі архітектури знаходиться шлюз API, який діє як основна точка входу для клієнтських запитів, виконуючи при цьому такі функції, як балансування навантаження і обмеження швидкості для забезпечення масштабованості системи і захисту від потенційних зловживань. Шлюз спрощує клієнтський інтерфейс системи, зменшуючи складність взаємодії між клієнтськими додатками та сервісами.

Система використовує балансувальники навантаження, які відіграють важливу роль в управлінні розподілом вхідного мережевого трафіку між декількома екземплярами серверів. Це гарантує, що жоден сервер не буде перевантажений, підвищуючи загальну надійність і доступність системи. Це також покращує масштабованість системи, розподіляючи навантаження на основі продуктивності сервера та поточного рівня попиту.

Система включає компонент RateLimiter, який забезпечує стабільність API, не дозволяючи окремим користувачам і сервісам надсилати надмірні запити, які можуть призвести до погіршення якості обслуговування. Цей компонент необхідний для підтримки продуктивності та доступності системи в пікові періоди.

За цими компонентами стоїть набір спеціалізованих мікросервісів, кожен з яких призначений для вирішення певного аспекту функціональності системи.

MonitorService підключається до зовнішніх датчиків і збирає дані про навколишнє середовище, такі як температура, вологість і якість повітря, в режимі реального часу. Служба обробляє вхідні дані і в деяких випадках збагачує або трансформує їх перед тим, як зберігати у спеціальній базі даних (тут вона називається MonitorDB). MonitorDB використовує оптимізовану для часових рядів базу даних, таку як Elasticsearch, щоб її можна було ефективно отримати і проаналізувати.

Служба ServiceRegistry виступає в ролі центрального вузла для реєстрації та виявлення сервісів, дозволяючи реєструвати нові екземпляри сервісів і відкривати їх для інших сервісів. Це полегшує динамічне масштабування і гарантує, що всі компоненти системи можуть ефективно взаємодіяти при додаванні або видаленні сервісів.

Спеціалізовані сервіси, такі як NewsService, SpacesService, UserService і AuthService, мають власну базу даних. Ці сервіси керують різними полями даних на платформі: NewsService управляє оновленням і передачею інформації, SpacesService управляє просторовими даними і функціями управління середовищем, UserService управляє профілями користувачів і контролем доступу, AuthService забезпечує функції автентифікації та авторизації для захисту системи.

Всі сервіси реєструють свої операції в централізованому LoggingSink, який є ключовим для моніторингу, налагодження та аудиту безпеки системи. Цей централізований механізм реєстрації гарантує, що журнали зберігаються в узгодженому форматі і легко доступні для аналізу.

AuthService відіграє важливу роль у забезпеченні авторизації та автентифікації всіх взаємодій відповідно до політики безпеки системи. Це важливо для запобігання несанкціонованому доступу та захисту конфіденційних даних.

Шлюз API є ключем до управління цією взаємодією. Він обробляє всі запити і перенаправляє їх до відповідних служб, одночасно керуючи такими наскрізними питаннями, як автентифікація, завершення SSL-з'єднання і конфігурація CORS. Це не лише спрощує розробку та підтримку клієнтських додатків, але й підвищує безпеку за рахунок централізованого контролю автентифікації та авторизації.[8]

Дані проходять через систему через мережу мікросервісів, кожен з яких спеціалізується на певній функціональній області. Наприклад, сервіс MonitorService безпосередньо взаємодіє із зовнішніми датчиками, розміщеними в навколишньому середовищі. Ці датчики можуть вимірювати різні параметри, такі як температура, вологість, рівень CO2 та інші показники навколишнього середовища, важливі для забезпечення комфорту та безпеки будівлі. Дані, зібрані з цих датчиків, надсилаються назад до сервісу MonitorService, який обробляє їх і зберігає в базі даних MonitorDB, що має потужні можливості повнотекстового пошуку та ефективної обробки даних часових рядів, які, ймовірно, будуть реалізовані в Elasticsearch.

Надійність системи підвищує служба ServiceRegistry, яка підтримує механізми динамічного виявлення та реєстрації, необхідні для мікросервісної архітектури. Ця служба гарантує, що екземпляри різних сервісів залишаються видимими для решти системи, коли вони обертаються вгору або вниз відповідно до вимог масштабування.

Інші служби домену, такі як NewsService, SpacesService, UserService і AuthService, відповідають за конкретні аспекти роботи системи:

Кожна служба реєструється в централізованому LoggingSink, що полегшує усунення несправностей, аудит і моніторинг безпеки. Це централізоване ведення журналів має важливе значення для підтримки огляду стану та активності системи і допомагає адміністраторам швидко виявляти та реагувати на потенційні проблеми.

Таким чином, система є надійною та гнучкою платформою, яка поєднує обробку даних у реальному часі з потужними функціями взаємодії з

користувачем для забезпечення комплексного рішення для моніторингу навколишнього середовища. Її архітектура розроблена для забезпечення масштабованості, надійності та безпеки і підходить для широкого спектру застосувань, від "розумних" будівель до систем промислового моніторингу.

3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Опис головних бізнес-процесів розроблюваного веб-застосунку

Використання технології Інтернету речей для управління екологічними показниками в приміщенні - це інноваційний підхід до підвищення комфорту, безпеки та ефективності.

Система, що розробляється, включає ряд інноваційних рішень, зокрема, використання розумних сенсорів і вузлів для збору даних про ключові параметри середовища, такі як температура, вологість, якість повітря, рівень освітленості та шум. Ці дані аналізуються та обробляються в реальному часі, забезпечуючи оперативне реагування на будь-які зміни та можливість оптимізації умов у приміщеннях.

Завдяки підключенню фізичних об'єктів і датчиків через Інтернет можна автоматизувати збір і аналіз даних для забезпечення точного моніторингу стану навколишнього середовища в режимі реального часу.

Система має бізнес-функції що покривають вимоги до системи, а саме:

- збір та обробка даних;
- моніторинг та регулювання;
- сповіщення та реагування на події;
- ролевий доступ до системи;
- управління доступом до даних;
- персоналізація інтерфейсу.

Якщо описувати збір та обробку даних, то це можна описати таким чином: зібрані сенсорами дані передаються в режимі реального часу, що дозволяє забезпечити актуальність інформації і швидке реагування на зміни умов середовища.

Використання автоматизованих інструментів для обробки даних уможливлює глибокий аналіз отриманих відомостей, їх класифікацію та

візуалізацію через інформаційні панелі, які надають зрозумілі звіти та графіки для кінцевих користувачів. Саме цей основний бізнес-процес зображений на рисунку 3.1.

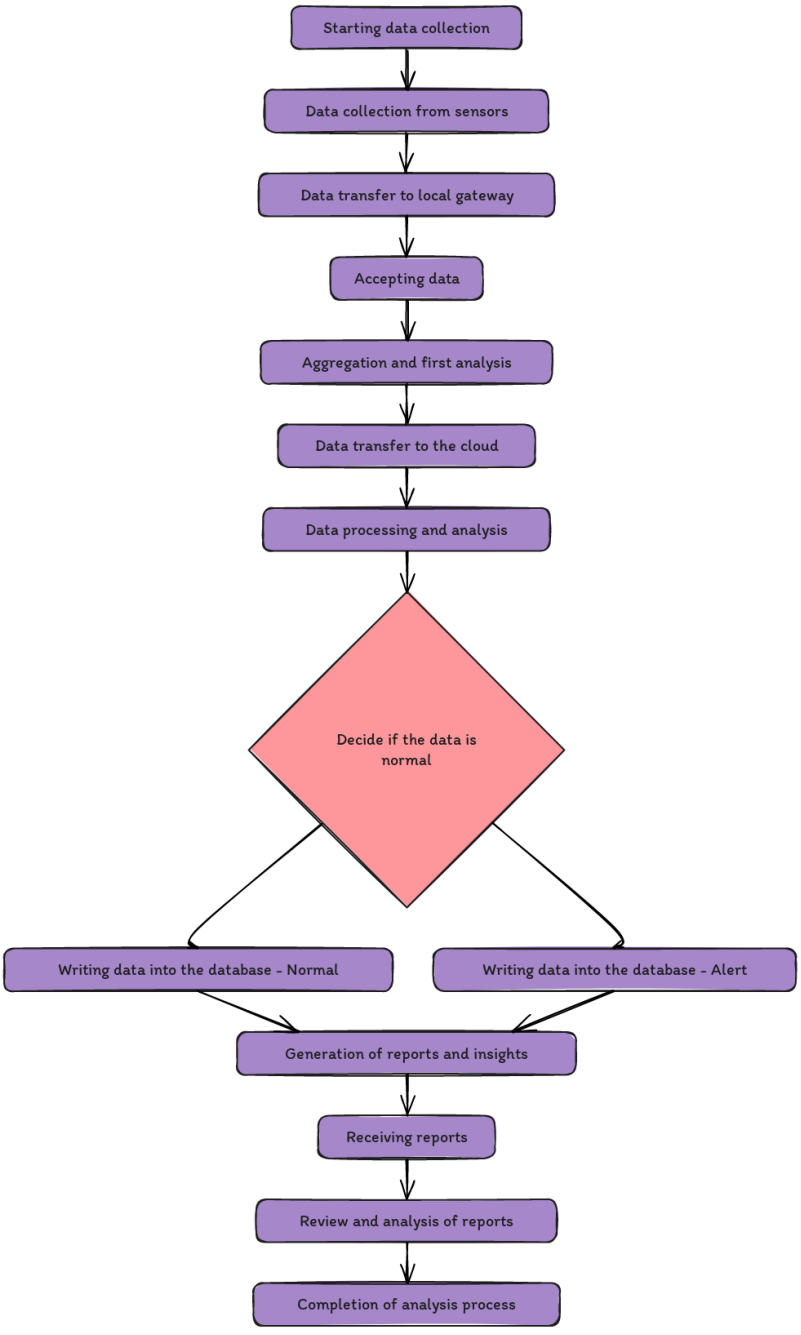

Рисунок 3.1 – Діаграма опису бізнес-процесу збору для обробки даних

Також важливим для системи є процес моніторингу і регулювання. Система моніторингу дозволяє постійно тримати на контролі рівень комфорту і безпеки в приміщеннях. Автоматичне регулювання систем опалення, вентиляції,

кондиціонування повітря та освітлення здійснюється на основі аналізу даних, отриманих від сенсорів. Це дозволяє не тільки підтримувати оптимальні умови але й оптимізувати енергоспоживання. Діаграма цього процесу зображена на рисунку 3.2.

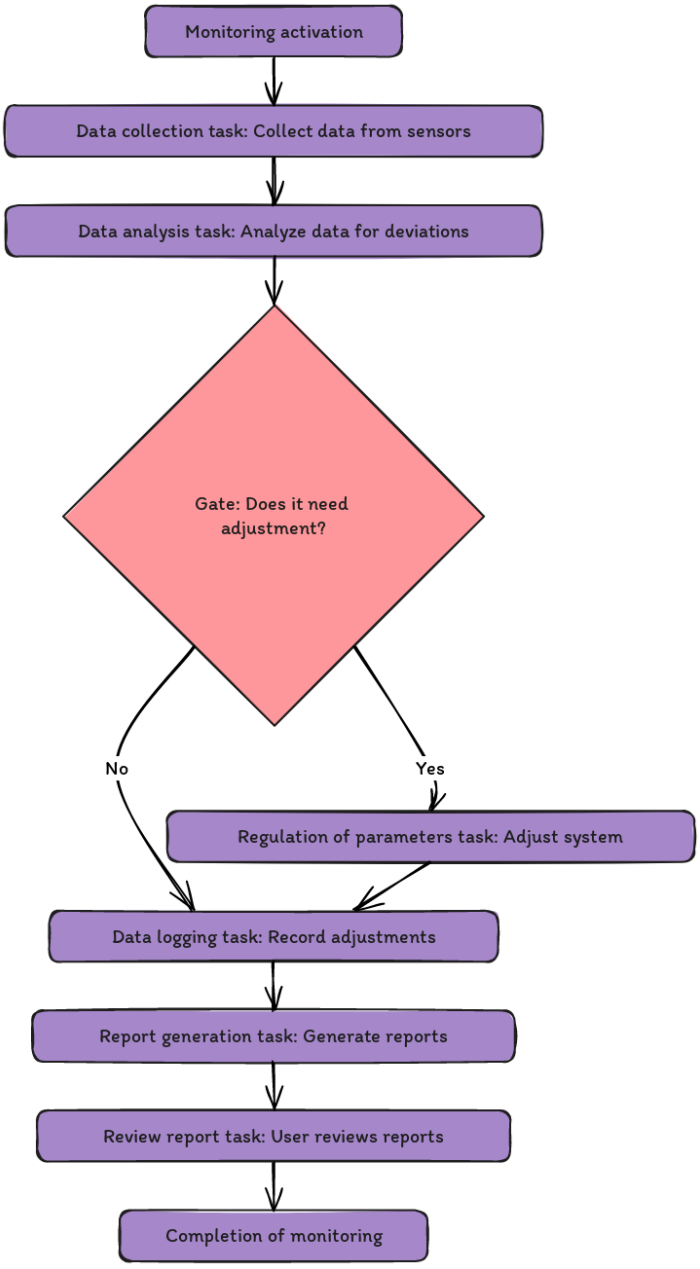

Рисунок 3.2 – Діаграма опису бізнес-процесу моніторингу

Також важливою частиною системи є сповіщення та реагування на події. Важливою частиною системи є механізм сповіщень, який автоматично інформує

управління об'єктом або власників про будь-які відхилення від норм у параметрах середовища.

Це дозволяє швидко реагувати на потенційні проблеми, мінімізуючи ризики для здоров'я та безпеки мешканців.

Крім того, використання прогностичних аналітичних моделей допомагає визначити потенційні несправності обладнання чи систем, що сприяє плануванню обслуговування та зниженню непередбачених збоїв.

Система моніторингу відповідає за постійний збір даних, їх аналіз і ідентифікацію аномалій або нестандартних ситуацій.

Оператор (або адміністратор) відповідає за прийняття рішень щодо подальших дій на основі отриманих сповіщень. Технічний персонал здійснює необхідні технічні втручання для вирішення ідентифікованих проблем.

Такий підхід не лише забезпечує високу точність моніторингу та автоматизацію управління середовищем, але й вносить важливий вклад у підвищення якості життя, енергоефективність та зниження витрат на утримання об'єктів.

3.2 Проєктування формату даних в системі

В ході аналізу предметної області були виділені ключові сутності, що використовуються у базі даних інформаційної системи. Якщо розглянути базу даних

Space таблиця зберігатиме інформацію про кожен простір, такий як кімнати, зони, офіси тощо. Поля цієї таблиці зображені на рисунку 3.3.

Поле	Тип даних	Опис
`space_id`	INTEGER (PK)	Унікальний ідентифікатор простору
`name`	VARCHAR(255)	Назва простору
`location`	VARCHAR(255)	Фізичне розташування простору
`description`	TEXT	Опис простору
`created_at`	TIMESTAMP	Дата та час створення запису
`updated_at`	TIMESTAMP	Дата та час останнього оновлення

Рисунок 3.3 – Зображення сутності Space

Таблиця для зберігання інформації про сенсори, котрі розміщені у кожному просторі зображена на рисунку 3.4.

Поле	Тип даних	Опис
`sensor_id`	INTEGER (PK)	Унікальний ідентифікатор сенсора
`space_id`	INTEGER (FK)	Ідентифікатор простору
`type`	VARCHAR(50)	Тип сенсора (напр., температура)
`model`	VARCHAR(255)	Модель сенсора
`status`	VARCHAR(50)	Статус сенсора (активний, неактивний)
`installed_at`	TIMESTAMP	Дата встановлення
`last_checked`	TIMESTAMP	Остання перевірка сенсора

Рисунок 3.4 – Зображення сутності Sensor

Таблиця для зберігання вимірювань з кожного сенсора зображена на рисунку 3.5.

Поле	Тип даних	Опис
`measurement_id`	INTEGER (PK)	Унікальний ідентифікатор вимірювання
`sensor_id`	INTEGER (FK)	Ідентифікатор сенсора
`value`	DECIMAL	Значення вимірювання
`measured_at`	TIMESTAMP	Дата та час вимірювання

Рисунок 3.5 – Зображення сутності Measurements

Таблиця для зберігання інформації про тривоги та сповіщення зображена на рисунку 3.6.

Поле	Тип даних	Опис
`alert_id`	INTEGER (PK)	Унікальний ідентифікатор тривоги
`sensor_id`	INTEGER (FK)	Ідентифікатор сенсора, який спрацював
`message`	TEXT	Повідомлення тривоги
`alerted_at`	TIMESTAMP	Дата та час спрацювання тривоги

Рисунок 3.6 – Зображення сутності Alert

Для сервісу "Users" в системі, яка призначена для управління показниками навколишнього середовища, важливо мати структуровану базу даних, що може керувати інформацією про користувачів, їхні ролі та взаємодію з системою.

Основна таблиця для зберігання інформації про користувачів. Кожен запис відображає окремого користувача системи.

Таблиця Users в системі виконує ключову роль, зберігаючи всю основну інформацію про користувачів, які взаємодіють з системою. Вона організована для забезпечення ефективного доступу та управління даними користувачів, включаючи їх ідентифікаційні дані, контактну інформацію, а також дати створення та оновлення профілів.

Ця таблиця також служить основою для забезпечення безпеки та ідентифікації користувачів, а її дані використовуються для аутентифікації та авторизації доступу до різних частин системи. Використання такої таблиці дозволяє системі масштабуватися та адаптуватися до зростаючих потреб у керуванні користувачами у складних інформаційних середовищах. Сама таблиця зображена на рисунку 3.7.

Поле	Тип даних	Опис
`user_id`	INTEGER (PK)	Унікальний ідентифікатор користувача
`username`	VARCHAR(255)	Ім'я користувача для входу
`password_hash`	VARCHAR(255)	Хеш пароля
`email`	VARCHAR(255)	Електронна пошта користувача
`first_name`	VARCHAR(255)	Ім'я користувача
`last_name`	VARCHAR(255)	Прізвище користувача
`created_at`	TIMESTAMP	Дата та час створення запису
`updated_at`	TIMESTAMP	Дата та час останнього оновлення

Рисунок 3.7 – Зображення сутності User

Таблиця для зберігання інформації про ролі користувачів в системі зображена на рисунку 3.8.

Поле	Тип даних	Опис
`role_id`	INTEGER (PK)	Унікальний ідентифікатор ролі
`role_name`	VARCHAR(255)	Назва ролі
`description`	TEXT	Опис ролі

Рисунок 3.8 – Зображення сутності ролей

У системі, що використовується для управління показниками навколишнього середовища в приміщеннях, також застосовується MongoDB для керування даними, зокрема для зберігання новин. MongoDB, будучи документо-орієнтованою базою даних NoSQL, ідеально підходить для цього завдання завдяки своїй гнучкості та здатності ефективно обробляти великі обсяги даних.

Використання MongoDB дозволяє системі динамічно адаптуватися до змінних вимог до даних без необхідності перегляду всієї схеми бази даних. Кожна новина в системі зберігається у формі документа всередині колекції, що значно спрощує додавання, оновлення та видалення записів.

Документи в колекції новин мають поля, які описують заголовок новини, час її створення та останнього редагування, короткий опис та повний текст новини у форматі HTML. Також є посилання на зображення, пов'язані з кожною новиною. Окрім того, кожен документ містить унікальний ідентифікатор, що генерується MongoDB, який використовується для швидкого доступу та управління новинами.

Така структура забезпечує високу швидкість обробки запитів, особливо важливо для функціоналу, який вимагає миттєвого реагування, такого як відображення актуальних новин та їхнє оновлення. MongoDB також підтримує реплікацію та горизонтальне масштабування, що є критично важливим для забезпечення високої доступності та довгострокової масштабованості системи.

Таким чином, використання MongoDB в цій системі дозволяє не тільки гнучко управляти структурою даних новин, але й забезпечує необхідну швидкість та надійність в роботі з даними, що є важливим аспектом для будь-якої сучасної інформаційної системи.

У системі, яка керує показниками навколишнього середовища у приміщеннях, існує таблиця новин, яка використовує MongoDB для зберігання даних. Ця таблиця містить кілька ключових полів, що дозволяють детально описати кожну новину та забезпечити легкий доступ до її вмісту. Кожен запис у таблиці представляє окрему новину і включає такі поля:

– унікальний ідентифікатор новини, який використовується як первинний ключ для швидкого доступу та управління записами;

– заголовок новини, який чітко вказує на її тему і є важливим для залучення уваги користувачів;

– дата та час створення новини, що дозволяє користувачам відстежувати актуальність інформації;

– дата та час останнього редагування новини, що забезпечує інформацію про останні оновлення;

– короткий опис новини, який надає швидкий огляд її змісту та сприяє кращому розумінню теми перед читанням повного тексту;

– повний текст новини у форматі HTML, що дозволяє включати різноманітне форматування та мультимедійні елементи для більш ефективної презентації інформації;

– посилання на зображення, асоційоване з новиною, що може бути використане для візуального представлення теми або як елемент дизайну.

Використання MongoDB для зберігання цієї таблиці дозволяє системі легко масштабуватися та ефективно обробляти великі обсяги даних, забезпечуючи високу продуктивність та доступність інформації.

Кожен елемент даних зберігається в оптимізованому форматі, що сприяє швидкому пошуку та відновленню інформації, необхідної користувачам системи.

Таблиця з новинами зображена на рисунку 3.9.

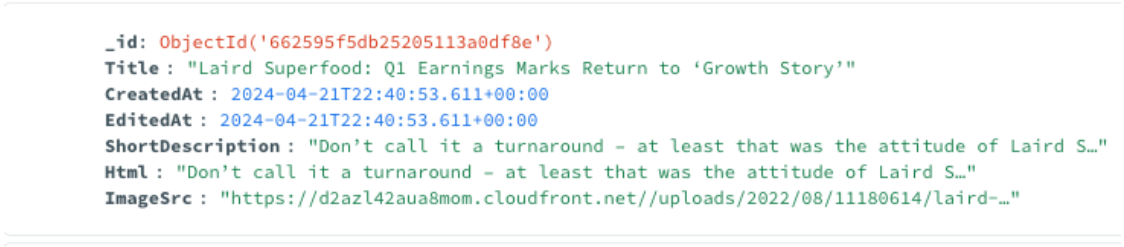

Рисунок 3.9 – Зображення сутності новин в MongoDb

3.3 Розробка візуального інтерфейсу застосунку

На скріншотах зображено користувацький інтерфейс системи управління навколишнім середовищем, розробленої на базі Next.js. Інтерфейс складається з декількох розділів, які включають головний дашборд, налаштування та обліковий запис користувача.

Головний дашборд містить різноманітні віджети, що відображають ключові метрики системи, такі як бюджет, кількість клієнтів, прогрес завдань і загальний прибуток. Віджети представлені у формі карток із зрозумілими піктограмами та вказівниками змін, які дозволяють користувачу швидко оцінити поточний стан і динаміку системи. Наприклад, є картка з графіком продажів, який показує місячну динаміку бронювань, і кругова діаграма, яка ілюструє розподіл трафіку користувачів за типами пристроїв.

У розділі налаштувань користувач має можливість керувати своїми персональними налаштуваннями та вибирати параметри сповіщень, що включають електронні листи та мобільні повідомлення. Також є опції для зміни пароля та інших важливих налаштувань безпеки акаунта.

На сторінці облікового запису користувача присутні поля для оновлення особистих даних, таких як ім'я, електронна адреса та номер телефону. Це дозволяє користувачам легко підтримувати актуальність своїх контактних даних у системі.

Загалом, інтерфейс є чистим та сучасним, з легко доступними розділами та налаштуваннями, що сприяє зручності користувачів та ефективності управління ресурсами системи управління.

На рисунку 3.10 представлено головний дашборд системи управління середовищем. Ця сторінка відображає ключові бізнес-метрики, такі як бюджет, кількість клієнтів, прогрес завдань, загальний прибуток, а також детальний графік продажів і діаграму джерел трафіку.

Графіки і діаграми виконані в сучасному стилі з використанням різних відтінків синього та помаранчевого кольорів, що дозволяє швидко зорієнтуватися в даних.

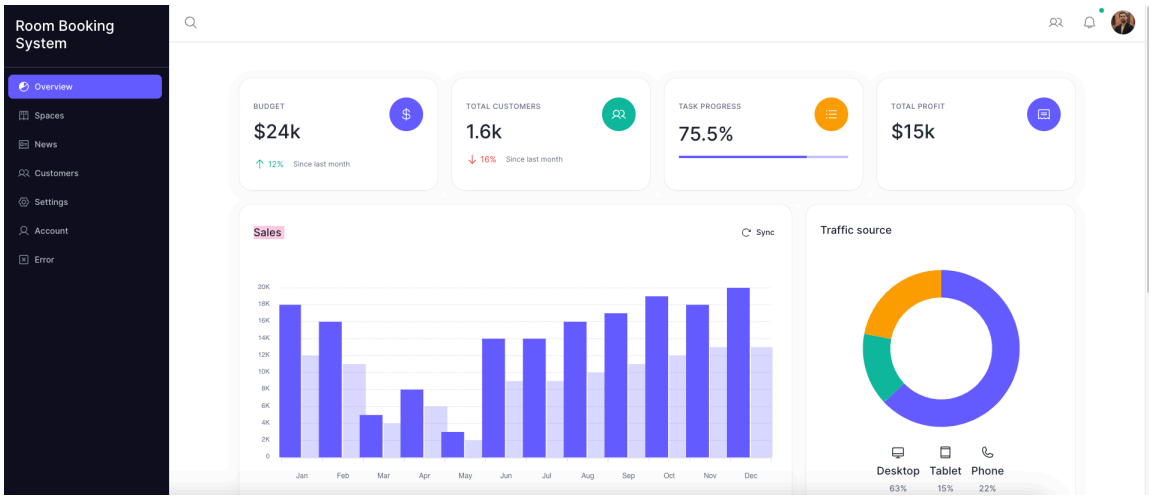

Рисунок 3.10 – Основний дашборд системи

На рисунку 3.11 відображено інтерфейс налаштувань системи. У цьому розділі користувачі можуть змінювати параметри сповіщень, такі як електронна пошта і мобільні повідомлення, та управляти налаштуваннями безпеки свого облікового запису, включаючи пароль.

Інтерфейс чітко організований і легкий у використанні.

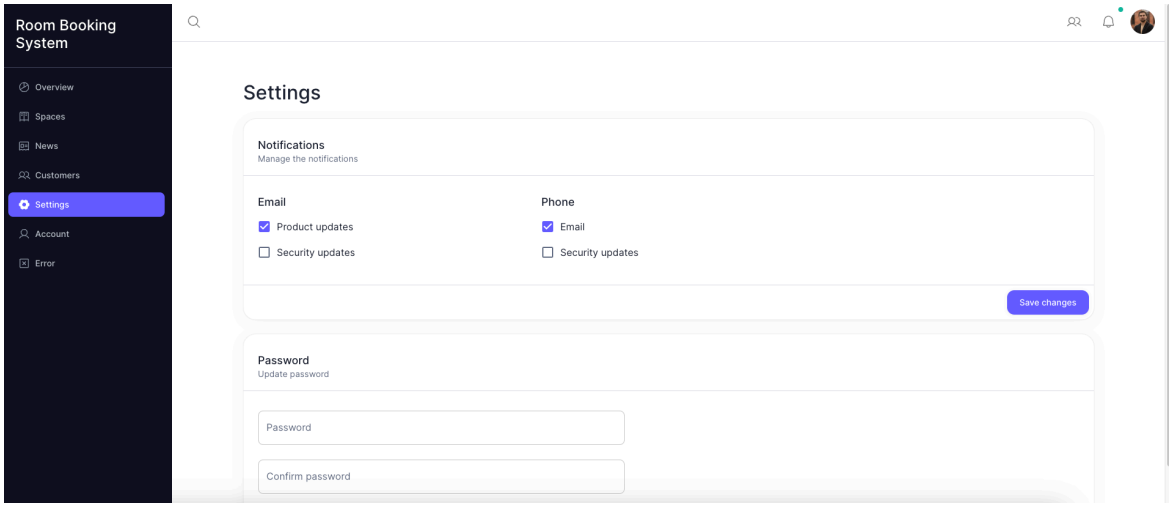

Рисунок 3.11 – Сторінка налаштування системи

На рисунку 3.12 показана сторінка профілю користувача. Тут відображена інформація про користувача, включно з його фотографією, ім'ям, прізвищем, електронною адресою і телефонним номером. Користувачі можуть редагувати свої персональні дані за допомогою кнопки "Оновити інфо".

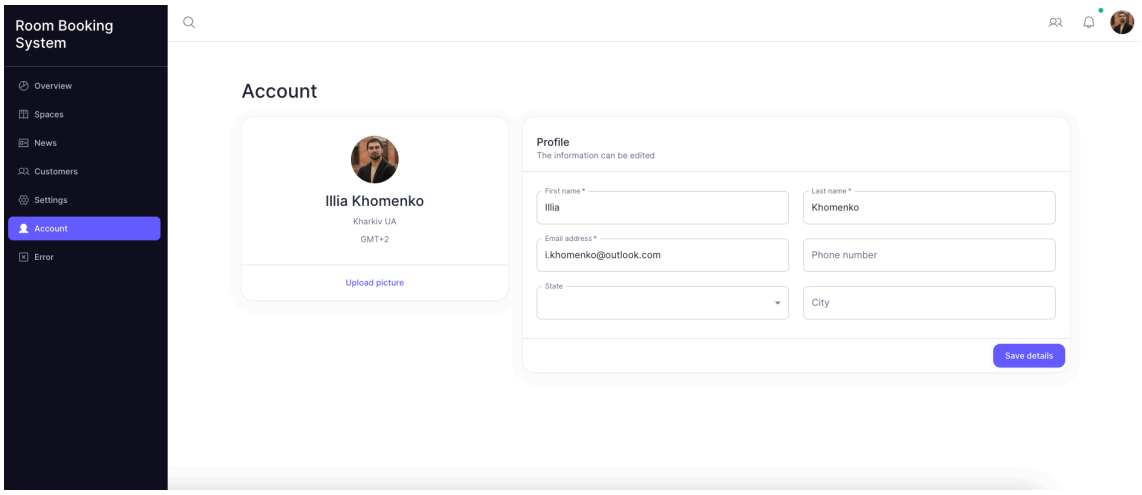

Рисунок 3.12 – Сторінка профілю

На рисунку 3.13 представлена аналітика використання системи. Цей розділ містить графіки, які відображають кількість бронювань, зайнятість та інші показники ефективності за різні періоди. Графіки мають зрозумілу колірну диференціацію і легкі для сприйняття.

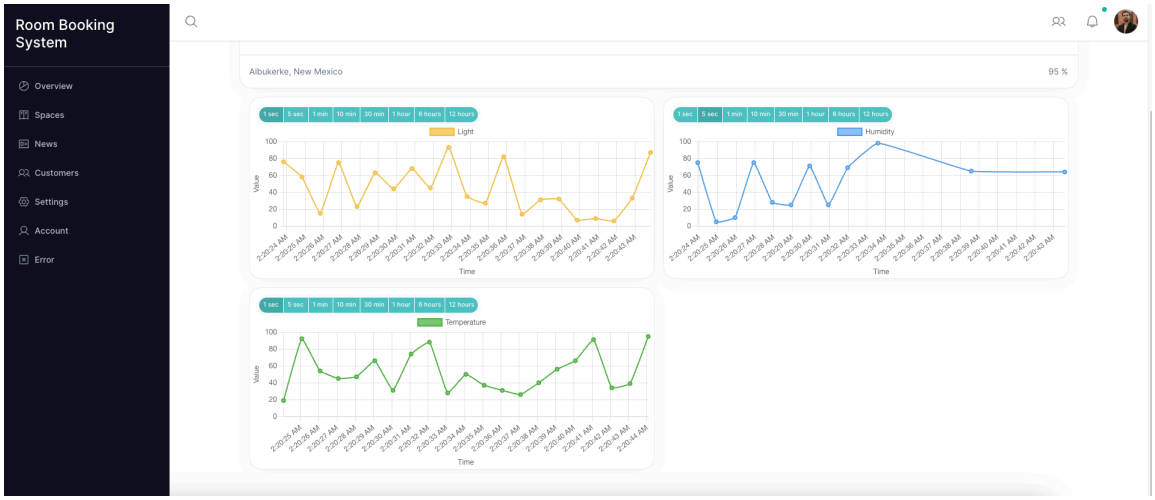

Рисунок 3.13 – Сторінка збору статистики

На рисунку 3.14 зображено додаткові можливості системи, що включають відображення можливих локацій. Кожен сервіс представлений у вигляді картки з логотипом, коротким описом і кнопкою для активації або налаштування інтеграції.

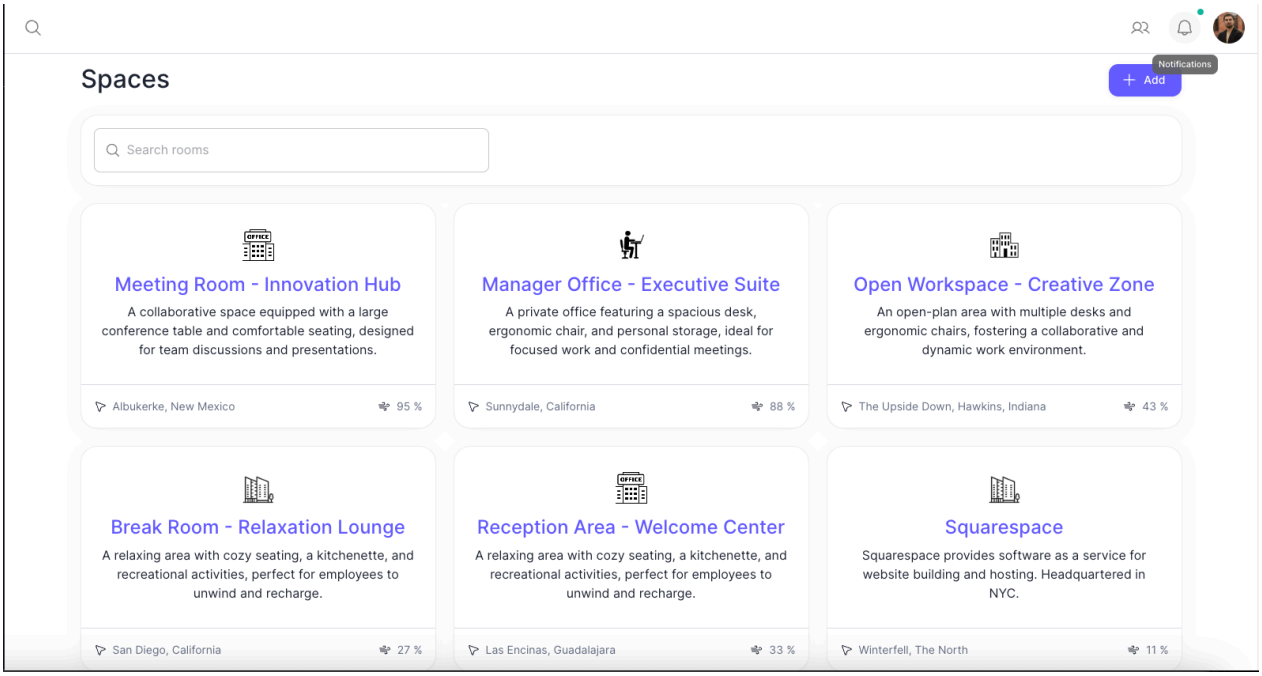

Рисунок 3.14 – Сторінка локацій

Компонент RealtimeChart використовує React hooks для управління станом та ефектами. Він ініціює з'єднання з сервером через WebSocket, де отримує дані про температуру в реальному часі. Після встановлення з'єднання, компонент відправляє запит на сервер із вказаним інтервалом оновлення даних. Сервер відправляє оновлення даних, які компонент отримує та обробляє для оновлення графіка.

Дані для графіка зберігаються у структурі, яка динамічно оновлюється. Нові дані додаються на графік, і, залежно від інтервалу оновлення, старі дані можуть бути видалені для оптимізації відображення. Це забезпечує візуалізацію поточної інформації без перевантаження графіка.

Налаштування графіка визначені таким чином, щоб адаптувати відображення під різні часові інтервали. Графік налаштовано на автоматичну адаптацію до розмірів контейнера, з відображенням осей та налаштуванням мінімальних та максимальних значень для осі Y. Часова шкала на осі X автоматично адаптується до обраного інтервалу оновлення, показуючи дані від секунд до годин, що дозволяє користувачам з легкістю спостерігати за тенденціями та змінами в даних.

Такий підхід не тільки забезпечує ефективну візуалізацію даних у реальному часі, але й підвищує інтерактивність і корисність системи моніторингу, дозволяючи користувачам швидко реагувати на зміни в оточуючому середовищі. Лістинг коду наведений у лістингу 3.15

```
'use client';

import React, { useEffect, useState } from 'react';
import { Line } from 'react-chartjs-2';
import {
    Chart as ChartJS,
    CategoryScale,
    LinearScale,
    PointElement,
    LineElement,
    Tooltip,
    Legend,
```

```
    TimeScale,
    TimeSeriesScale,
    ChartOptions
} from 'chart.js';
import 'chartjs-adapter-date-fns';  /

ChartJS.register(
    CategoryScale,
    LinearScale,
    PointElement,
    LineElement,
    Tooltip,
    Legend,
    TimeScale,
    TimeSeriesScale
);

interface ChartData {
    labels: Date[];
        label: string;
        data: number[];
        borderColor: string;
        backgroundColor: string;
        fill: boolean;
        lineTension: number;
        borderWidth: number;
    }[];
}

interface RealtimeChartProps {
    updateInterval: number;
    chartColor: string;
    backgroundColor: string;
    label: string;
}

const initialData: ChartData = {
    labels: [],
    datasets: [
        {
            label: 'Temperature',
```

```
            data: [],
            borderColor: 'rgb(75, 192, 192)',
            backgroundColor: 'rgba(75, 192, 192, 0.5)',
            fill: false,
            lineTension: 0.1,
            borderWidth: 2,
        },
    ],
};

const RealtimeChart: React.FC<RealtimeChartProps> = ({ updateInterval, chartColor,
backgroundColor, label }) => {
    initialData.datasets[0].label = label;
    initialData.datasets[0].borderColor = chartColor;
    initialData.datasets[0].backgroundColor = backgroundColor;

    const [data, setData] = useState<ChartData>(initialData);

    useEffect(() => {
        const websocket = new WebSocket('ws://localhost:5054/ws');

        websocket.onopen = () => {
            // Send a message to the server to change the data sending rate based on
the interval
            websocket.send(JSON.stringify({ type: "change-rate", interval:
updateInterval }));
        };

        websocket.onmessage = (event) => {
            const message = JSON.parse(event.data);
            if (message.Data && !isNaN(message.Data)) {
                updateChartData(message.Data);
            }
        };

        return () => {
            websocket.close();
        };
    }, [updateInterval]);

    const updateChartData = (newDataPoint: number) => {
```

```
        setData(prevData => {
            const newTime = new Date();
            const newData = [...prevData.datasets[0].data, newDataPoint];
            const newLabels = [...prevData.labels, newTime];

            const maxPoints = updateInterval <= 30000 ? 30 : updateInterval <= 60000 ?
20 : 10;
            while (newData.length > maxPoints) {
                newData.shift();
                newLabels.shift();
            }

            return {
                ...prevData,
                labels: newLabels,
                datasets: [{
                    ...prevData.datasets[0],
                    data: newData,
                }]
            };
        });
    };

    const option : ChartOptions<'line'> = {
        responsive: true,
        scales: {
            x: {
                type: 'time',
                time: {
                    unit: updateInterval <= 30000 ? 'second' : updateInterval <= 600000
? 'minute' : 'hour',
                    displayFormats: {
                        second: 'h:mm:ss a',
                        minute: 'h:mm a',
                        hour: 'MMM d, h a'
                    }
                },
                display: true,
                title: {
                    display: true,
                    text: 'Time'
```

```
                }
            },
            y: {
                display: true,
                title: {
                    display: true,
                    text: 'Value'
                },
                suggestedMin: 0,
                suggestedMax: 100
            }
        },
        maintainAspectRatio: false
    };

    return (
        <div style={{ height: '300px' }}>
            <Line data={data} options={option} />
        </div>
    );
};

export default RealtimeChart;
```

Форма входу розроблена для забезпечення зручності користувачів та високої безпеки при автентифікації.

Компонент SignInForm використовує хуки React для керування станом, такі як useState для відстеження відображення пароля та активності входу в систему. Використання хука useRouter з Next.js дозволяє програмно керувати маршрутизацією, наприклад, оновлювати сторінку після успішного входу.

Форма включає поля для введення електронної адреси та пароля, кожне з яких обробляється за допомогою компонента Controller з бібліотеки react-hook-form, що дозволяє легко інтегрувати поля введення з валідатором форм, заснованим на схемі валідації zod. Валідація поля електронної адреси перевіряє, чи введено коректний формат email, та чи поле не є порожнім. Аналогічна перевірка проводиться і для поля пароля.

Додаткові елементи інтерфейсу, такі як кнопки для перегляду або приховування пароля, реалізовані за допомогою іконок EyeIcon та EyeSlashIcon. Це забезпечує користувачам можливість контролювати видимість свого пароля під час вводу, підвищуючи їх комфорт і безпеку використання форми.

У разі виникнення помилок під час спроби входу, на формі відображаються сповіщення про помилки, що дозволяє користувачам швидко ідентифікувати та виправити проблеми. Посилання на відновлення забутого пароля та реєстрацію нового акаунту також є частиною форми, забезпечуючи легкий доступ до цих функцій.

Такий підхід не лише сприяє підвищенню безпеки під час авторизації, але й забезпечує високу зручність і інтуїтивно зрозумілу взаємодію з системою, що є особливо важливим для систем, які використовуються в широкомасштабних та різноманітних додатках.

Лістинг коду авторизаційної форми показано в лістингу нижче:

```
'use client';

import * as React from 'react';
import RouterLink from 'next/link';
import { useRouter } from 'next/navigation';
import { zodResolver } from '@hookform/resolvers/zod';
import Alert from '@mui/material/Alert';
import Button from '@mui/material/Button';
import FormControl from '@mui/material/FormControl';
import FormHelperText from '@mui/material/FormHelperText';
import InputLabel from '@mui/material/InputLabel';
import Link from '@mui/material/Link';
import OutlinedInput from '@mui/material/OutlinedInput';
import Stack from '@mui/material/Stack';
import Typography from '@mui/material/Typography';
import { Eye as EyeIcon } from '@phosphor-icons/react/dist/ssr/Eye';
import { EyeSlash as EyeSlashIcon } from '@phosphor-icons/react/dist/ssr/EyeSlash';
import { Controller, useForm } from 'react-hook-form';
import { z as zod } from 'zod';

import { paths } from '@/paths';
import { authClient } from '@/lib/auth/client';
```

```
import { useUser } from '@/hooks/use-user';

const schema = zod.object({
  email: zod.string().min(1, { message: 'Email is required' }).email(),
  password: zod.string().min(1, { message: 'Password is required' }),
});

type Values = zod.infer<typeof schema>;

const defaultValues = { email: 'i.khomenko@outlook.com', password: 'Secret1' }
satisfies Values;

export function SignInForm(): React.JSX.Element {
  const router = useRouter();

  const { checkSession } = useUser();

  const [showPassword, setShowPassword] = React.useState<boolean>();

  const [isPending, setIsPending] = React.useState<boolean>(false);

  const {
    control,
    handleSubmit,
    setError,
    formState: { errors },
  } = useForm<Values>({ defaultValues, resolver: zodResolver(schema) });

  const onSubmit = React.useCallback(
    async (values: Values): Promise<void> => {
      setIsPending(true);

      const { error } = await authClient.signInWithPassword(values);

      if (error) {
        setError('root', { type: 'server', message: error });
        setIsPending(false);
        return;
      }

      await checkSession?.();
```

```
      router.refresh();
    },
    [checkSession, router, setError]
  );

  return (
    <Stack spacing={4}>
      <Stack spacing={1}>
        <Typography variant="h4">Sign in</Typography>
        <Typography color="text.secondary" variant="body2">
          Don&apos;t have an account?{' '}
          <Link component={RouterLink} href={paths.auth.signUp} underline="hover"
variant="subtitle2">
            Sign up
          </Link>
        </Typography>
      </Stack>
      <form onSubmit={handleSubmit(onSubmit)}>
        <Stack spacing={2}>
          <Controller
            control={control}
            name="email"
            render={({ field }) => (
              <FormControl error={Boolean(errors.email)}>
                <InputLabel>Email address</InputLabel>
                <OutlinedInput {...field} label="Email address" type="email" />
                {errors.email ? <FormHelperText>{errors.email.message}</FormHelperText>
: null}
              </FormControl>
            )}
          />
          <Controller
            control={control}
            name="password"
            render={({ field }) => (
              <FormControl error={Boolean(errors.password)}>
                <InputLabel>Password</InputLabel>
                <OutlinedInput
                  {...field}
                  endAdornment={
```

```
                    showPassword ? (
                      <EyeIcon
                        cursor="pointer"
                        fontSize="var(--icon-fontSize-md)"
                        onClick={(): void => {
                          setShowPassword(false);
                        }}
                      />
                    ) : (
                      <EyeSlashIcon
                        cursor="pointer"
                        fontSize="var(--icon-fontSize-md)"
                        onClick={(): void => {
                          setShowPassword(true);
                        }}
                      />
                    )
                  }
                  label="Password"
                  type={showPassword ? 'text' : 'password'}
                />
                {errors.password ?
<FormHelperText>{errors.password.message}</FormHelperText> : null}
              </FormControl>
            )}
          />
          <div>
            <Link component={RouterLink} href={paths.auth.resetPassword}
variant="subtitle2">
              Forgot password?
            </Link>
          </div>
          {errors.root ? <Alert color="error">{errors.root.message}</Alert> : null}
          <Button disabled={isPending} type="submit" variant="contained">
            Sign in
          </Button>
        </Stack>
      </form>
      <Alert color="warning">
        Use{' '}
        <Typography component="span" sx={{ fontWeight: 700 }} variant="inherit">
```

```
          i.khomenko@outlook.com
        </Typography>{' '}
        with password{' '}
        <Typography component="span" sx={{ fontWeight: 700 }} variant="inherit">
          Secret1
        </Typography>
      </Alert>
    </Stack>
  );
}
```

В цілому можна зробити висновок щодо візуального інтерфейсу дипломного проєкту, розробленого на базі Next.js та Material-UI, свідчить про високий рівень інтеграції сучасних технологій взаємодії користувача та ефективності управління системою управління середовищем кімнат. Чіткий та інтуїтивно зрозумілий дизайн інтерфейсу забезпечує легкий доступ до основних функцій та звітності, що є ключовим для оптимізації користувацького досвіду та підвищення ефективності оперативного управління.

Візуальні елементи, такі як графіки, інформаційні панелі та форми з валідацією, виконані на високому рівні, забезпечують користувачам зручність при використанні системи і дозволяють швидко аналізувати поточні дані для прийняття обґрунтованих рішень.

Основна частина інтерфейсу містить інструменти для регулювання параметрів середовища. Користувачі можуть налаштовувати рівень освітленості (Light), температуру (Temperature) та вологість (Humidity) за допомогою трьох горизонтальних повзунків. Кожен повзунок має свою шкалу та маркер, який можна пересувати для встановлення бажаного рівня відповідного параметра. Кольори повзунків підкреслюють різні параметри: освітленість відображається жовтим, температура - червоним, а вологість - синім кольорами.

Посередині інтерфейсу розташована кнопка "Apply", яка дозволяє застосувати внесені зміни параметрів. Після натискання кнопки, нові налаштування передаються до системи управління, яка автоматично коригує роботу пристроїв, таких як кондиціонери та зволожувачі, для досягнення бажаних умов у приміщенні. Цей інтерфейс зображений на рисунку 3.17.

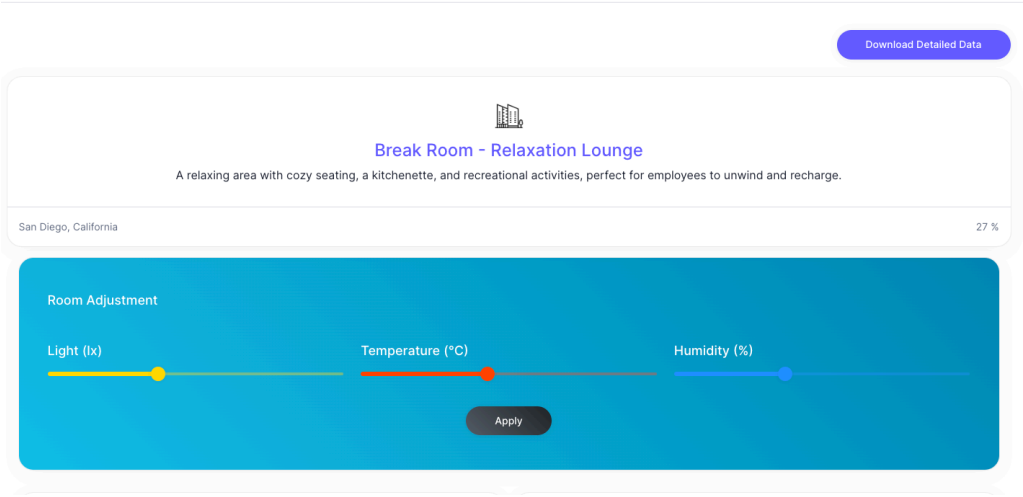

Рисунок 3.17 – Інтерфейс управління показниками середовища

При експорті даних з інтерфейсу користувач натискає кнопку "Download Data", розміщену у верхньому правому куті екрану. Ця дія ініціює процес генерації CSV-файлу, який містить детальні історичні дані про параметри навколишнього середовища, такі як освітленість, температура і вологість. Дані оновлюються кожні 10 секунд, забезпечуючи високий рівень деталізації. Система швидко формує файл і автоматично починає його завантаження на пристрій користувача.

Завантажений файл дає користувачу можливість проаналізувати зібрані дані у зручному форматі, використовуючи їх для подальшого аналізу та прийняття рішень щодо оптимізації умов у приміщенні.

Загалом, застосунок демонструє ефективне поєднання функціональності, безпеки та естетичної привабливості, вибудовуючи міцну основу для подальшого розширення функціоналу та інтеграції додаткових сервісів. Це робить систему не лише зручною для користувачів, але й адаптивною до майбутніх вимог ринку та технологічних оновлень.

3.4 Розробка серверної частини застосунку

У рамках дипломного проєкту була розроблена серверна частина системи, спрямована на управління новинами та авторизацію користувачів. Центральним елементом цієї системи є різноманітні ендпоінти, кожен з яких відіграє свою роль для забезпечення ефективності та безпеки діяльності.

Авторизаційні ендпоінти забезпечують реєстрацію нових користувачів, вхід до системи та управління сесіями, що включає оновлення токенів доступу та відновлення паролів. Ці ендпоінти критично важливі для підтримки безпеки користувацьких даних та забезпечення контрольованого доступу до функцій системи.

Ендпоінти управління новинами дозволяють користувачам публікувати, оновлювати, видаляти та переглядати новини. Це дає можливість швидко реагувати на події та утримувати аудиторію в курсі останніх подій і оновлень. Використання таких ендпоінтів сприяє залученню відвідувачів та підвищенню інформованості користувачів системи.

Цілісність цієї системи підтримується завдяки тісній інтеграції між ендпоінтами авторизації та управління контентом, що забезпечує плавність процесів і високий рівень захисту даних. Всі взаємодії з сервером відбуваються через захищені з'єднання, гарантуючи конфіденційність та цілісність передаваних даних. Це створює надійну основу для системи, яка може ефективно масштабуватися та адаптуватися до зростаючих вимог користувачів і технологічного розвитку. Ендпоінти представлені на рисунку 3.18, а детальний опис зроблений нижче.

POST /api/news – цей ендпоінт використовується для додавання нових новин до системи. Користувачі можуть використовувати цю функціональність для публікації новин в системі, що дозволяє швидко реагувати на актуальні події або оновлення. Вхідні дані для цього запиту зазвичай включають заголовок, текст статті та можливо мультимедійні компоненти.

PUT /api/news/{id} – через цей ендпоінт користувачі можуть оновлювати вже опубліковані новини. Це критично важливо для виправлення помилок, оновлення інформації або зміни контенту статтей. Функціональність оновлення дозволяє підтримувати контент свіжим та релевантним.

DELETE /api/news/{id} – ендпоінт для видалення новин з системи. Ця можливість важлива для управління контентом, дозволяючи адміністраторам чи редакторам видаляти застарілі або недоречні статті, тим самим підтримуючи високу якість та актуальність інформації у системі.

GET /api/news – цей ендпоінт використовується для отримання списку всіх новин, опублікованих у системі.

Така структура ендпоінтів сприяє гнучкому управлінню новинним контентом і розширює можливості користувачів системи у створенні, модифікації та споживанні інформації, забезпечуючи при цьому ефективне розповсюдження та актуалізацію новин.

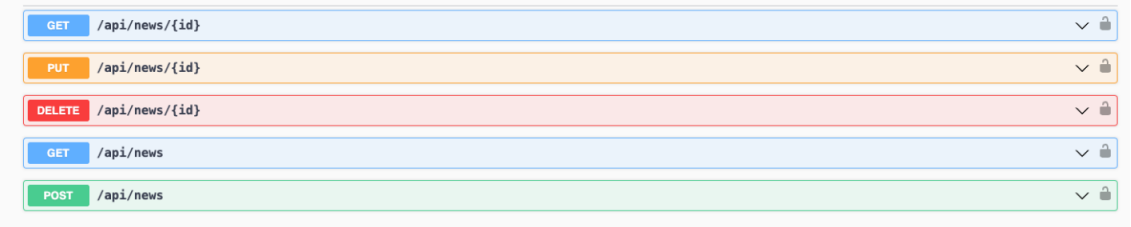

Рисунок 3.18 – Ендпоінти управління новинами

Серверна архітектура проєкту містить ряд ендпоінтів, які забезпечують основні операції створення, читання, оновлення та видалення (CRUD) для новин, а також реалізують процеси авторизації.

Ці ендпоінти інтегровані в систему таким чином, щоб забезпечити безпеку, ефективність та високу доступність сервісів. Наступний опис деталізує функціональність кожного з цих ендпоінтів, демонструючи як вони взаємодіють з користувачами та як обробляють дані в системі. Лістинг коду показаний нижче:

```
[HttpGet("{id}")]
public async Task<IActionResult> GetNewsById(string id)
```

```
{
    var query = new GetNewsByIdQuery(id);

    var result = await Sender.Send(query);

    return result.IsSuccess ? Ok(result.Value) : BadRequest(result.Error);
}

[HttpGet]
[AllowAnonymous]
public async Task<IActionResult> GetNews([FromQuery] PaginationModel model)
{
    var query = new GetNewsPartByRecentQuery(model.Page, model.Limit);

    var result = await Sender.Send(query);

    return result.IsSuccess ? Ok(result.Value) : BadRequest(result.Error);
}

[HttpPost]
public async Task<IActionResult> CreateNews(CreateNewsRequest request)
{
    var command = new CreateNewsCommand(CreateNewsRequest.Map(request));

    var result = await Sender.Send(command);

    return result.IsSuccess ? Ok() : BadRequest(result.Error);
}

[HttpPut("{id}")]
public async Task<IActionResult> UpdateNews(string id, UpdateNewsRequest request)
{
    var command = new UpdateNewsCommand(UpdateNewsRequest.Map(id, request));

    var result = await Sender.Send(command);

    return result.IsSuccess ? Ok() : BadRequest(result.Error);
}

[HttpDelete("{id}")]
public async Task<IActionResult> DeleteNews(string id)
```

```
{
    var command = new DeleteNewsCommand(id);

    var result = await Sender.Send(command);

    return result.IsSuccess ? Ok() : BadRequest(result.Error);
}
```

Використання команд та запитів (Command and Query objects) свідчить про архітектурний підхід CQRS (Command Query Responsibility Segregation), який розділяє виконання операцій модифікації даних та їх запитів, що підвищує чистоту коду та спрощує його підтримку.

Асинхронність: Всі методи використовують асинхронне програмування, що забезпечує кращу продуктивність системи, дозволяючи обробляти великі обсяги запитів без блокування основного потоку виконання.

Створення новини (Create): Метод CreateNews приймає з запиту дані для створення новини, створює команду з цими даними, виконує її та повертає результат. Успіх операції веде до повернення статусу OK.

Читання новин (Read): Існує два методи для читання: GetNewsById для отримання новини за ID та GetNews з пагінацією для отримання списку новин. Обидва методи використовують запити до системи, обробляють результати та повертають відповідні HTTP відповіді.

Оновлення новини (Update): Метод UpdateNews використовує дані з запиту для оновлення новини, генерує відповідну команду та відправляє її до обробки. У відповідь на успіх повертається статус OK.

Видалення новини (Delete): Метод DeleteNews створює команду для видалення новини за заданим ID та обробляє її, повертаючи статус OK при успішному видаленні.

У всіх методах використовується підхід до обробки помилок, який при невдачі повертає клієнту статус BadRequest разом із відповідним повідомленням про помилку. Це забезпечує зрозумілість системи для користувачів та сприяє виявленню та виправленню помилок при взаємодії з API.

Загалом, ці ендпоінти формують потужну серверну логіку, яка ефективно управляє новинним контентом, забезпечує безпеку даних та сприяє розширенню функціональності системи в майбутньому.

У рамках дипломного проєкту серверна частина системи також обслуговує ендпоінти, які забезпечують управління користувацькими обліковими записами та процедурами безпеки.

Ці ендпоінти включають в себе реєстрацію нових користувачів, авторизацію та відновлення доступу до облікових записів, що є ключовими для забезпечення надійної роботи системи.

POST /register - цей ендпоінт призначений для створення нових облікових записів. Користувачі вводять свої персональні дані, такі як ім'я, електронна пошта та пароль, що проходять валідацію перед збереженням у базі даних. Завдяки цьому процесу, система забезпечує вхід користувачів, які відповідають установленим критеріям безпеки.

POST /login - використовується для входу в систему, перевіряючи введені користувачем облікові дані і, у разі успіху, надаючи токен доступу, який забезпечує подальше сеансове управління.

POST /refresh - цей ендпоінт дозволяє оновити токени доступу, забезпечуючи безперервність сеансів користувачів без необхідності повторного входу.

GET /confirmEmail - використовується для підтвердження електронної пошти користувачів, що є важливим для верифікації їхньої ідентичності та активізації облікового запису.

POST /resendConfirmationEmail - дозволяє відновити відправлення листа для підтвердження електронної пошти у випадку, якщо попереднє сповіщення не було доставлено або втрачено.

POST /forgotPassword і POST /resetPassword - ці ендпоінти відповідають за процеси відновлення забутих паролів. Вони дозволяють користувачам ініціювати процедуру зміни пароля, забезпечуючи кроки для верифікації користувача та встановлення нового пароля.

GET /manage/info і POST /manage/2fa - забезпечують користувачам можливість переглядати та управляти налаштуваннями своїх облікових записів, включаючи встановлення двофакторної автентифікації для збільшення рівня безпеки.

Ці ендпоінти формують повний цикл управління користувацькими даними та безпекою, від реєстрації до управління акаунтом, забезпечуючи систему потужними інструментами для захисту інформації та зручності користувачів. Саме ці ендпоінти зображені на рисунку 3.20.

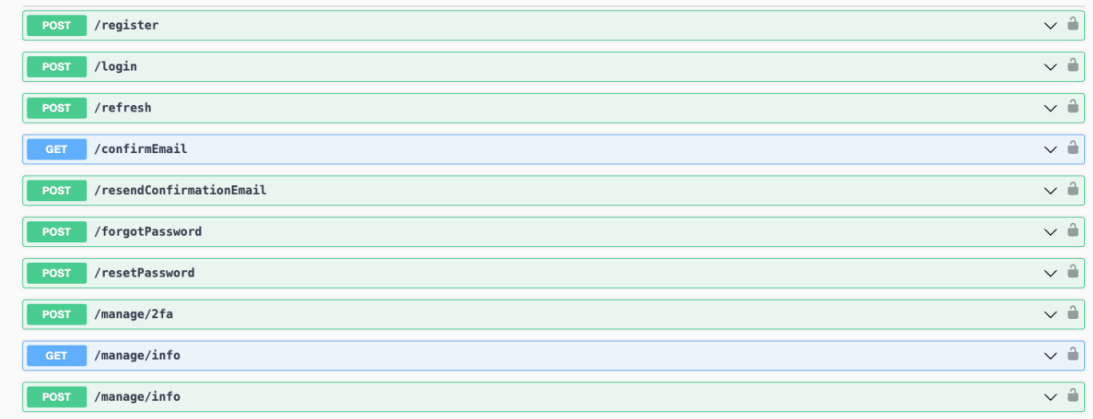

Рисунок 3.20 – Ендпоінти авторизації

У рамках дипломного проєкту серверний код, реалізований на ASP.NET, виконує критично важливу роль у забезпеченні динамічної взаємодії з користувачами через вебсокети, що дозволяє системі обробляти та передавати різні види параметрів середовища в реальному часі.

Ця функціональність особливо важлива для додатків, які вимагають неперервного моніторингу та відповіді на зміни умов середовища, таких як системи керування показниками навколишнього середовища в приміщеннях. В коді обробляються запити на підключення до вебсокетів, встановлюючи з'єднання при успішній валідації запиту та відхиляючи невідповідні запити. В коді обробляються запити на підключення до вебсокетів, встановлюючи з'єднання при успішній валідації запиту та відхиляючи невідповідні запити.

Після встановлення з'єднання, сервер обробляє вхідні повідомлення від клієнтів, які можуть містити команди для зміни інтервалу оновлення даних або запити на відправку даних.

Це забезпечує гнучке управління взаємодією в залежності від потреб користувача. Залежно від вхідних команд, сервер налаштовує частоту відправлення даних, таких як параметри середовища, що зчитуються з відповідних датчиків. Ця інформація серіалізується у формат JSON та відправляється назад до клієнта.

При отриманні сигналу на закриття з'єднання від клієнта сервер коректно закриває сесію, забезпечуючи адекватне завершення взаємодії.

Цей підхід не лише підвищує відповідність системи до змін у параметрах середовища, але й забезпечує високу масштабованість та ефективність обробки даних, що є важливим для будь-якої системи, що вимагає постійного моніторингу та відповіді у реальному часі.

3.5 Розробка основного модулю управління середовищем

Система моніторингу та контролю навколишнього середовища забезпечує підтримку оптимальних умов в приміщенні за допомогою автоматичного управління температурою, вологістю і освітленням. Система складається з датчиків, алгоритмів обробки даних і контролерів для управління таким обладнанням, як Кондиціонери, Зволожувачі та освітлювальні прилади.

Датчики встановлюються в кожній області кімнати і використовують веб-сокети для безперервного вимірювання і передачі даних про рівні температури, вологості і освітленості в центральну систему управління. Це дозволяє системі миттєво реагувати на будь-які зміни, забезпечуючи безперервний потік даних в режимі реального часу.

Коли користувач відправляє запит на зміну певного показника, наприклад, перепаду температури, система обробляє цей запит і відправляє відповідну команду на контролер кондиціонера. Контролер регулює роботу кондиціонера і

знижує температуру до заданого рівня. У той же час система враховує вплив кондиціонера на вологість повітря, оскільки Кондиціонер видаляє вологу з повітря. Зволожувач повітря буде працювати автоматично, щоб підтримувати комфортний рівень вологості.

Коли користувач надсилає запит на зміну рівня освітлення, система надсилає відповідну команду контролеру освітлення, який регулює інтенсивність світла до бажаного рівня. Датчики продовжують контролювати рівень освітленості і передавати дані в режимі реального часу, дозволяючи системі швидко реагувати на будь-які зміни.

Алгоритм системи також включає механізм зворотного зв'язку для підтримки стабільності параметрів.

Якщо система виявляє, що температура або вологість змінюються занадто швидко, відрегулюйте роботу відповідного пристрою, зменшіть інтенсивність або вимкніть її, щоб уникнути надмірних значень.

Через прискорене зниження температури додатковий режим охолодження активується, коли система виявляє, що падіння відбувається повільно. жовтень. Це досягається збільшенням потужності кондиціонера або включенням додаткового охолоджуючого пристрою. Таким чином, система дозволяє швидше досягти бажаного рівня температури.

Центральна система управління обробляє дані датчиків і вимоги користувачів і забезпечує точність і ефективність регулювання.

При зміні температури система враховує вплив вологості і включає відповідний пристрій, щоб запобігти надмірному падінню вологості, яке може викликати дискомфорт. Аналогічним чином, при зміні освітлення система враховує природне освітлення і оптимізує роботу освітлювального приладу для досягнення бажаного рівня освітленості з мінімальним споживанням енергії.

Розглянемо приклад зниження кімнатної температури з 25 °C до 20 °C. Коли користувач надсилає запит на зниження температури, система обробляє цей запит і надсилає команду на контролер кондиціонера.

Потім контролер активує Кондиціонер, щоб знизити температуру. Під час роботи кондиціонера датчик постійно вимірює температуру і передає дані в центральну систему управління.

Центральна система контролює швидкість зниження температури. Коли система виявляє, що зниження відбувається повільно, включається режим прискореного охолодження, і інтенсивність кондиціонування повітря збільшується. Це дозволяє швидко досягти бажаної температури і забезпечує комфорт користувача в найкоротші терміни.

Система також використовує алгоритми зворотного зв'язку для підтримки стабільності параметрів навколишнього середовища.

Загальна схема системи така: датчик вимірює параметри, і дані надсилаються до центральної системи, яка потім аналізує їх через веб-сайт і приймає рішення щодо управління пристроєм.

На рисунку 3.21 показано, як дані надходять у систему та як контролер отримує команди для налаштування параметрів навколишнього середовища для забезпечення стабільності та комфорту.

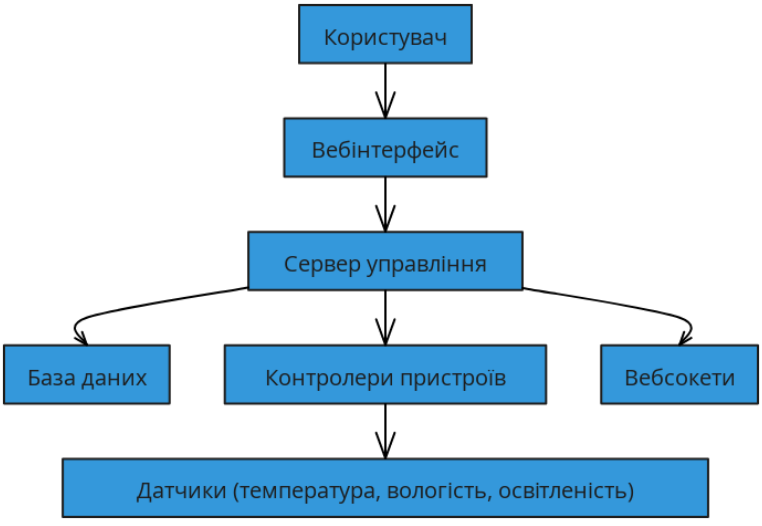

Рисунок 3.21 – Схема управління середовищем

Команда відправляється на контролер, який регулює роботу пристрою і підтримує комфортні умови в приміщенні.

Серверна частина системи моніторингу та контролю параметрів навколишнього середовища виконує кілька основних функцій: обробка даних, отриманих від датчиків, обробка запитів користувачів на зміну параметрів, управління пристроями (кондиціонерами, зволожувачами, освітлювальними приладами) та забезпечення можливості експорту даних у форматі CSV.

Основні компоненти серверної частини включають сервер обробки вебсокетів, базу даних для збереження історичних даних та логіку управління.

Процес роботи серверної частини розділяються на декілька етапів серед яких: отримання даних від датчиків, обробка запитів користувачів, управління пристроями та збереження даних

Датчики, розташовані в приміщеннях, постійно вимірюють параметри навколишнього середовища (температура, вологість, освітленість) та передають ці дані до сервера через вебсокети. Сервер обробляє ці дані в режимі реального часу.

Користувачі надсилають запити на зміну параметрів через вебінтерфейс. Ці запити надходять до сервера, який аналізує їх і передає відповідні команди на контролери пристроїв.

Контролери пристроїв (кондиціонерів, зволожувачів, освітлювальних приладів) отримують команди від сервера та змінюють свої налаштування для досягнення бажаних параметрів навколишнього середовища.

Сервер зберігає історичні дані про параметри середовища в базі даних. Це забезпечує можливість подальшого аналізу та експорту даних.

Сервер також має механізми для обробки високого навантаження, оскільки він обробляє великий обсяг даних, що надходять від датчиків у режимі реального часу. Використання MongoDB дозволяє зберігати дані в зручному для аналізу форматі та швидко здійснювати доступ до них. Це забезпечує надійну роботу системи навіть за умов високого навантаження, що виникає при великій кількості підключених датчиків та користувачів, що активно використовують систему.

Крім того, серверна частина інтегрує механізми зворотного зв'язку, що дозволяють автоматично коригувати роботу пристроїв на основі аналізу даних. Наприклад, якщо система виявляє, що температура знижується повільніше, ніж очікувалося, вона може активувати режим прискореного охолодження. Це забезпечує підтримку оптимальних умов у приміщенні з максимальною ефективністю та мінімальними затратами ресурсів.

Таким чином, серверна частина не лише обробляє дані та управляє пристроями, але й забезпечує динамічне реагування на зміни в умовах середовища, що підвищує загальну ефективність і надійність системи.

Коли користувач натискає кнопку "Download Data", сервер генерує CSV-файл, який містить історичні дані про параметри середовища за обраний період. Файл завантажується на пристрій користувача.

Датчики збирають дані про параметри навколишнього середовища, відправляє їх в центральну систему управління, обробляє запити користувачів, регулює роботу пристрою через контролер, забезпечуючи оптимальні умови в приміщенні.

Таким чином, система екологічного моніторингу та контролю є комплексним рішенням для забезпечення комфортних умов у приміщенні. Вона використовує новітні технології для збору та обробки даних, автоматизовані алгоритми для прийняття рішень та інтелектуальні контролери для управління пристроями. Це дозволяє досягти високого рівня комфорту і ефективності при одночасному зниженні енергоспоживання та оптимізації роботи всіх систем.

4 МЕТОДОЛОГІЯ РОБОТИ З ДАНИМИ В СИСТЕМІ

4.1 Огляд параметрів середовища

Температура середовища відіграє критичну роль у визначенні продуктивності праці. Різні дослідження підтверджують, що коливання температури можуть значно впливати на здатність працівників зосереджуватися, зберігати високий рівень мотивації та ефективно виконувати свої робочі обов’язки.

Численні дослідження визначають оптимальний температурний діапазон для максимальної продуктивності як приблизно 21-23 градуси Цельсія. У цьому діапазоні спостерігається найвищий рівень комфорту для більшості людей, що сприяє збереженню фокусу та зниженню втоми.

При температурі нижче оптимального діапазону зростає витрата енергії на підтримку тілесного тепла, що веде до зниження рівня активності та підвищення втоми. Це, в свою чергу, може призвести до зниження уваги та посилення помилок у роботі.

Підвищені температури спричиняють перегрів організму, що вимагає додаткової енергії для терморегуляції. Це може викликати зневоднення, втому та зниження ментальної концентрації.

Працівники можуть відчувати дискомфорт, зниження мотивації та загальне зниження продуктивності.

В різних секторах економіки вплив температури на продуктивність може мати різні наслідки. Наприклад, в офісних приміщеннях підтримання оптимальної температури є критичним для забезпечення тривалого зосередження та ефективної роботи з документами. У виробничих приміщеннях, де фізичні навантаження вищі, оптимальне управління температурою допомагає знизити ризики здоров'ю і підвищити безпеку праці.

Для підвищення продуктивності та комфорту робочого середовища рекомендується встановлювати та підтримувати системи клімат-контролю, що

забезпечують стабільне регулювання температури. Також важливо забезпечувати працівникам доступ до засобів індивідуального терморегулювання, наприклад, вентиляторів або персональних обігрівачів, в залежності від їхніх потреб та переваг. Саме графік впливу температури на продуктивність праці зображено на рисунку 4.1.

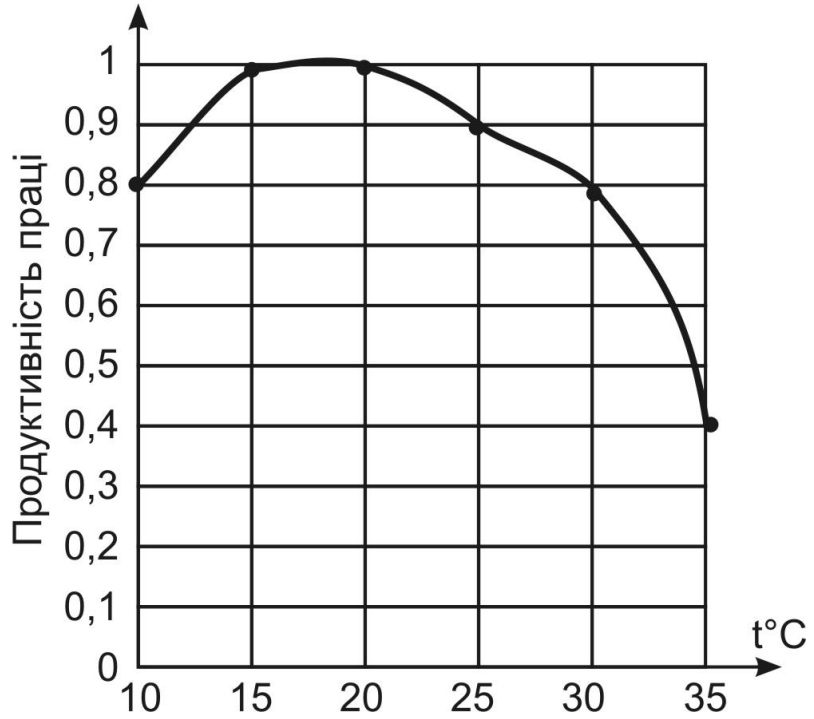

Рисунок 4.1 – Графік впливу температури на продуктивність праці

Освітленість в робочому просторі є одним із ключових факторів, що впливають на продуктивність працівників та їхнє самопочуття. Наявність адекватного освітлення сприяє зниженню втоми, підвищенню концентрації та загалом кращому виконанню трудових завдань.

Оптимальне освітлення залежить від ряду чинників, включаючи тип діяльності, особисті переваги працівників та особливості робочого простору. Достатнє освітлення знижує ризик втоми очей, що часто виникає в результаті тривалої роботи за комп'ютером або при читанні документів.

Втома очей може призвести до головного болю та зниження здатності зосереджуватися. Натуральне світло має позитивний вплив на регуляцію біоритмів людини, покращуючи загальний настрій і продуктивність. Наявність

великих вікон або використання освітлювальних приладів, що імітують денне світло, може допомогти у підтримці цього балансу.

Забезпечення адекватного освітлення в робочому просторі є інвестицією у здоров'я та продуктивність працівників, що в кінцевому підсумку призводить до підвищення ефективності робочих процесів і зниження витрат на охорону здоров'я. Графік впливу освітленості на продуктивність праці зображено на рисунку 4.2.

Рисунок 4.2 – Графік впливу освітленості на продуктивність праці

Вологість середовища має суттєвий вплив на комфорт та продуктивність працівників. Занадто низька або висока вологість може створювати некомфортні умови, які негативно впливають на здоров'я і здатність людей ефективно працювати.

Правильне регулювання вологості є ключовим аспектом створення оптимального робочого середовища. Низька вологість може викликати сухість у роті, подразнення очей та шкіри, що знижує комфорт та здатність концентруватися. Висока вологість збільшує потовиділення, що також знижує комфорт і може сприяти розвитку цвілі в приміщеннях, що негативно впливає на якість повітря і здоров'я людей.

Вологість впливає на внутрішню температуру тіла, здатність тіла регулювати температуру через потовиділення, а також загальний фізичний комфорт. Відповідно, коли вологість є оптимальною, люди відчувають себе більш комфортно і здатні краще зосередитися на своїх завданнях.

Для моніторингу та контролю рівня вологості у робочих просторах важливо використовувати гігрометри, які дозволяють відстежувати вологість у реальному часі.

Зволожувачі або осушувачі повітря можуть бути використані для корегування рівня вологості до бажаного діапазону.

Вентиляція допомагає контролювати рівень вологості і знижує концентрацію шкідливих речовин у повітрі. Провітрювання приміщень є особливо важливим у холодні місяці, коли вологість може знижуватися через опалення.

4.2 Методології збору даних

Для забезпечення ефективного моніторингу та управління середовищем у сучасних інтелектуальних системах управління будівлями, необхідно використовувати передові методології збору даних. Цей процес охоплює використання спеціалізованих датчиків, ефективні комунікаційні протоколи, інтегровані системи збору даних та розширені засоби аналізу та візуалізації.

Дані збираються за допомогою різноманітних датчиків, які вимірюють температуру, вологість і освітленість кожні 10 секунд. Ці дані передаються на

сервер через вебсокети для подальшої обробки та зберігання. Використовується платформа .NET для реалізації цих функцій.

Обробка вебсокетів показана в наступному коді:

```
using System;
using System.Net.WebSockets;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using Newtonsoft.Json;

public class SensorData
{
    public double Temperature { get; set; }
    public double Humidity { get; set; }
    public double Light { get; set; }
}

public class WebSocketClient
{
    private static async Task SendSensorDataAsync(ClientWebSocket ws)
    {
        while (ws.State == WebSocketState.Open)
        {
            var data = new SensorData
            {
                Temperature = temp
                Humidity = hum
                Light = light
            };

            var json = JsonConvert.SerializeObject(data);
            var bytes = Encoding.UTF8.GetBytes(json);

            await ws.SendAsync(new ArraySegment<byte>(bytes),
WebSocketMessageType.Text, true, CancellationToken.None);
            await Task.Delay(10000);
        }
    }

    public static async Task Main(string[] args)
```

```
    {
        using (var ws = new ClientWebSocket())
        {
            await ws.ConnectAsync(new Uri("ws://localhost:5054/ws"),
CancellationToken.None);
            await SendSensorDataAsync(ws);
        }
    }
}
```

Дані, отримані з датчиків, можуть містити шуми та аномалії, які можуть вплинути на точність аналізу та прийняття рішень. Попередня обробка даних є критичним етапом у забезпеченні їх надійності та точності. Цей процес включає фільтрацію шуму, виявлення та видалення аномалій, а також заповнення пропусків у даних.

Метод ковзного середнього використовується для згладжування даних та видалення випадкових флуктуацій, що не відображають реальних змін параметрів середовища. Цей метод полягає у заміні кожного значення на середнє значення декількох сусідніх значень. Формула ковзного середнього виглядає наступним чином:

$$\mathrm{MA}_t = \frac{1}{n} \sum_{i=0}^{n-1} x_{t-i}$$

де MAt – ковзне середнє в момент часу t, n – кількість періодів у ковзному вікні, x – значення показника.

Реалізація цього методу показана в лістингу нижче:

```
public static double[] MovingAverage(double[] data, int windowSize)
{
    double[] result = new double[data.Length - windowSize + 1];
    for (int i = 0; i < result.Length; i++)
    {
        double sum = 0;
        for (int j = 0; j < windowSize; j++)
        {
            sum += data[i + j];
        }
```

```
            result[i] = sum / windowSize;
        }
        return result;
    }
```

Аномалії можуть виникати через короткочасні збої в роботі датчиків або вплив зовнішніх факторів.

Для виявлення аномалій використовується метод порогового аналізу, при якому значення, що виходять за межі встановлених діапазонів, вважаються аномальними і видаляються або замінюються на середнє значення.

Пороговий аналіз дозволяє визначити межі нормальних значень для кожного параметра середовища.

Реалізація методу порогового аналізу показано в даному лістингу:

```
public static double[] RemoveAnomalies(double[] data, double lowerBound, double
upperBound)
{
    for (int i = 0; i < data.Length; i++)
    {
        if (data[i] < lowerBound || data[i] > upperBound)
        {
            data[i] = (i > 0) ? data[i - 1] : 0;
        }
    }
    return data;
}
```

Іноді дані можуть бути відсутніми через технічні причини, такі як втрати в передачі даних або несправність датчиків. Для заповнення таких прогалин використовується метод лінійної інтерполяції, який обчислює відсутні значення на основі попередніх та наступних значень.

Формула лінійної інтерполяції:

$$x_i = x_{i-1} + \frac{(x_{i+1} - x_{i-1})}{(t_{i+1} - t_{i-1})} \cdot (t_i - t_{i-1})$$

де x_i – значення параметра в момент часу t_i, x_{i-1} і x_{i+1} – значення параметра в попередній і наступний моменти часу відповідно, t_i – час, для якого необхідно

визначити значення параметра, t_{i-1} і t_{i+1} – час попереднього і наступного вимірювань відповідно.

Реалізація методу лінійної інтерполяції показано в даному лістингу:

```
public static double[] InterpolateMissingData(double[] data)
{
    for (int i = 1; i < data.Length - 1; i++)
    {
        if (data[i] == 0)
        {
            int j = i + 1;
            while (j < data.Length && data[j] == 0)
            {
                j++;
            }
            if (j < data.Length)
            {
                double slope = (data[j] - data[i - 1]) / (j - i + 1);
                for (int k = i; k < j; k++)
                {
                    data[k] = data[i - 1] + slope * (k - i + 1);
                }
            }
        }
    }
    return data;
}
```

PID-регулятор є одним з найефективніших методів автоматичного регулювання в системах управління.

Він використовується для підтримки температури на заданому рівні шляхом постійного коригування вихідного сигналу системи управління на основі похибки між бажаним і фактичним значенням температури.

Основна ідея роботи PID-регулятора полягає в тому, що він обчислює вихідний сигнал, який складається з трьох компонентів: пропорційної, інтегральної та диференційної складових. Кожна з цих складових виконує свою функцію в процесі регулювання.

Пропорційна складова залежить від величини поточної похибки. Вона визначається як добуток коефіцієнта пропорційності K_p та похибки e(t):

$$P(t) = K_p \cdot e(t)$$

Ця складова забезпечує основну корекцію вихідного сигналу. Чим більша похибка, тим більший коригувальний сигнал генерується регулятором.

Інтегральна складова враховує сумарну похибку за весь час роботи регулятора. Вона обчислюється як добуток коефіцієнта інтегральної складової K_i та інтегралу похибки:

$$I(t) = K_i \int e(t)\,dt$$

Ця складова допомагає усунути сталу похибку, яка може залишатися після дії пропорційної складової. Вона враховує накопичену похибку за певний період часу, що дозволяє більш точно підтримувати задану температуру.

Диференційна складова обчислюється як добуток коефіцієнта диференційної складової K_d на швидкість зміни похибки:

$$D(t) = K_d \frac{de(t)}{dt}$$

Диференційна складова реагує на швидкість зміни похибки і допомагає зменшити осциляції системи, забезпечуючи більш плавний перехід до заданого значення.

Реалізація PID контролеру показана в лістингу нижче:

```
public class PIDController
{
    private double _kp;
    private double _ki;
    private double _kd;
    private double _previousError;
    private double _integral;

    public PIDController(double kp, double ki, double kd)
    {
```

```
        _kp = kp;
        _ki = ki;
        _kd = kd;
    }
    public double Compute(double setpoint, double actual, double deltaTime)
    {
        var error = setpoint - actual;
        _integral += error * deltaTime;
        var derivative = (error - _previousError) / deltaTime;
        _previousError = error;
        return _kp * error + _ki * _integral + _kd * derivative;
    }
}
```

В системі моніторингу та контролю параметрів середовища PID-регулятор відіграє ключову роль у підтриманні оптимальної температури. Він дозволяє автоматично коригувати роботу системи опалення або кондиціонування, забезпечуючи комфортні умови для користувачів. Використання PID-регулятора в поєднанні з іншими алгоритмами обробки даних, такими як згладжування та видалення аномалій, забезпечує високу точність і надійність системи.

Таким чином, PID-регулятор є ефективним інструментом для автоматичного регулювання температури, який забезпечує стабільність і комфорт в приміщенні. Його використання дозволяє підтримувати задані параметри середовища з високою точністю, що є критично важливим для забезпечення продуктивності та здоров'я користувачів.

Такий комплексний підхід до збору та аналізу даних дозволяє не тільки забезпечувати оптимальні умови для здоров'я та продуктивності людей, а й сприяє енергоефективності та екологічності сучасних будівельних рішень.

Зібрані дані акумулюються в централізованій базі даних, яка може бути реалізована на SQL Server, MongoDB. Це дозволяє забезпечити швидкий доступ до даних для аналізу та звітування.

Для аналізу зібраних даних можуть бути використані інструменти машинного навчання та штучного інтелекту, наприклад, ML.NET, що дозволяє

імплементувати моделі прогнозування та аналізу поведінки параметрів середовища безпосередньо в рамках системи.

Цей комплексний підхід до збору та аналізу даних в рамках .NET екосистеми не тільки покращує точність і оперативність управління середовищем, але й забезпечує гнучкість та масштабованість системи.

4.3 Збір та аналіз даних в системі

Для забезпечення ефективного управління середовищем в інтелектуальних будівлях, ми розробили систему, яка використовує передові аналітичні інструменти і технології на базі .NET.

Система збирає дані з множини датчиків, використовуючи високопродуктивні протоколи передачі даних і забезпечує глибокий аналіз цих даних з метою оптимізації робочого середовища.

Аналіз даних в системі моніторингу та контролю параметрів навколишнього середовища здійснюється на основі зібраних даних з різноманітних датчиків, встановлених у приміщенні.

Хоча в роботі представлено лише усереднені показники, система також підтримує збір і аналіз даних з високою точністю, включаючи посекундний моніторинг, який можна завантажити для подальшого аналізу.

Усереднені показники дозволяють оцінити загальний стан параметрів навколишнього середовища протягом певного періоду. Це забезпечує загальне уявлення про комфорт і умови в приміщенні без необхідності обробки великого обсягу даних.

Для передачі даних по освітленості в системі використовуються сучасні сенсори та мікроконтролери, зокрема, датчик освітленості BH1750. Цей датчик здатен точно вимірювати рівень освітленості в навколишньому середовищі і передавати ці дані за допомогою інтерфейсу SPI. Зокрема, BH1750 передає дані у вигляді пакетів, що містять байти даних, включаючи MSB (Most Significant

Byte) і LSB (Least Significant Byte). Наприклад, передане повідомлення може виглядати як комбінація байтів: 0x00, 0x34, 0x12, 0x00.

Повна схема роботи системи збору даних про освітленість зображена на рисунку 4.3.

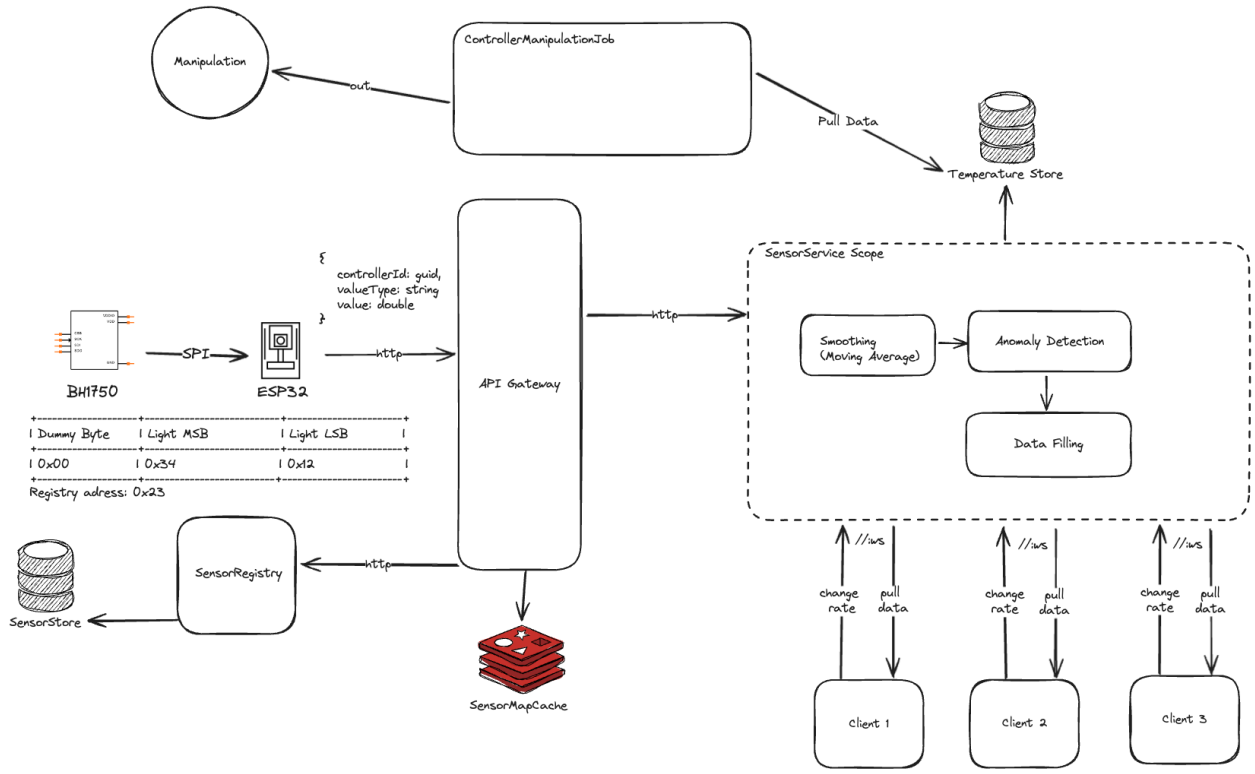

Рисунок 4.3 – Схема роботи системи збору даних про освітленість

Мікроконтролер ESP32, підключений до датчика через SPI, відповідає за прийом цих даних і їх подальшу обробку. ESP32 отримує вимірювання освітленості від BH1750 і конвертує їх у формат, придатний для передачі по HTTP. Дані передаються у вигляді JSON-повідомлень, що показано в лістингу нижче:

```
{
	"controllerId": "guid",
	"valueType": "light",
	"value": 500.0
}
```

Далі ці дані передаються на API-шлюз, який є центральним елементом комунікації в системі. API-шлюз приймає дані від ESP32 та передає їх у сервіси обробки даних.

Сервіс обробки даних SensorService відповідає за згладжування та видалення аномалій у даних. Для згладжування даних використовується метод ковзного середнього, що дозволяє усунути випадкові коливання та покращити точність вимірювань.

Аномалії, виявлені в процесі обробки, позначаються як невірні дані і замінюються на середнє значення з попередніх та наступних вимірювань. Таким чином, система забезпечує більш стабільні та точні дані для подальшого аналізу.

Після попередньої обробки дані зберігаються в базі даних Light Store, де вони можуть бути використані для подальшого аналізу та генерації звітів. Також, дані можуть бути кешовані в SensorMapCache для швидкого доступу.

Вся система інтегрується з клієнтськими пристроями через вебсокет-з'єднання. Це дозволяє клієнтам отримувати дані в реальному часі, змінювати параметри частоти оновлення даних та керувати іншими аспектами роботи системи. Користувачі можуть взаємодіяти із системою через зручний інтерфейс, що забезпечує повний контроль над параметрами середовища.

Система моніторингу та регулювання вологості та температури працює майже ідентично з системою освітленості, проте існують деякі ключові відмінності в способі збору даних та їх обробки.

Для моніторингу вологості використовується датчик BME280. Цей датчик забезпечує високоточне вимірювання відносної вологості навколишнього середовища. Дані з BME280 передаються до мікроконтролера ESP32 за допомогою інтерфейсу SPI. Вихідний сигнал датчика передається у вигляді пакетів байтів, що включають MSB (Most Significant Byte) і LSB (Least Significant Byte). Наприклад, передане повідомлення може містити байти 0x00, 0x68, 0x3A, де перший байт є dummy byte, а наступні два байти містять значення вологості.

ESP32 отримує ці дані, конвертує їх у формат JSON і відправляє на API-шлюз за допомогою HTTP-запитів. Типове JSON-повідомлення може виглядати так:

```
{
    "controllerId": "guid",
    "valueType": "humidity",
    "value": 46.0
}
```

API-шлюз приймає ці дані і передає їх далі до компонента обробки даних (SensorService Scope). Цей компонент виконує згладжування даних методом ковзного середнього (Moving Average) для видалення шумів та аномалій. Далі, після аналізу, дані зберігаються у базі даних (Humidity Store) для подальшого використання і аналізу.

Повна схема роботи системи збору даних про вологість зображена на рисунку 4.4.

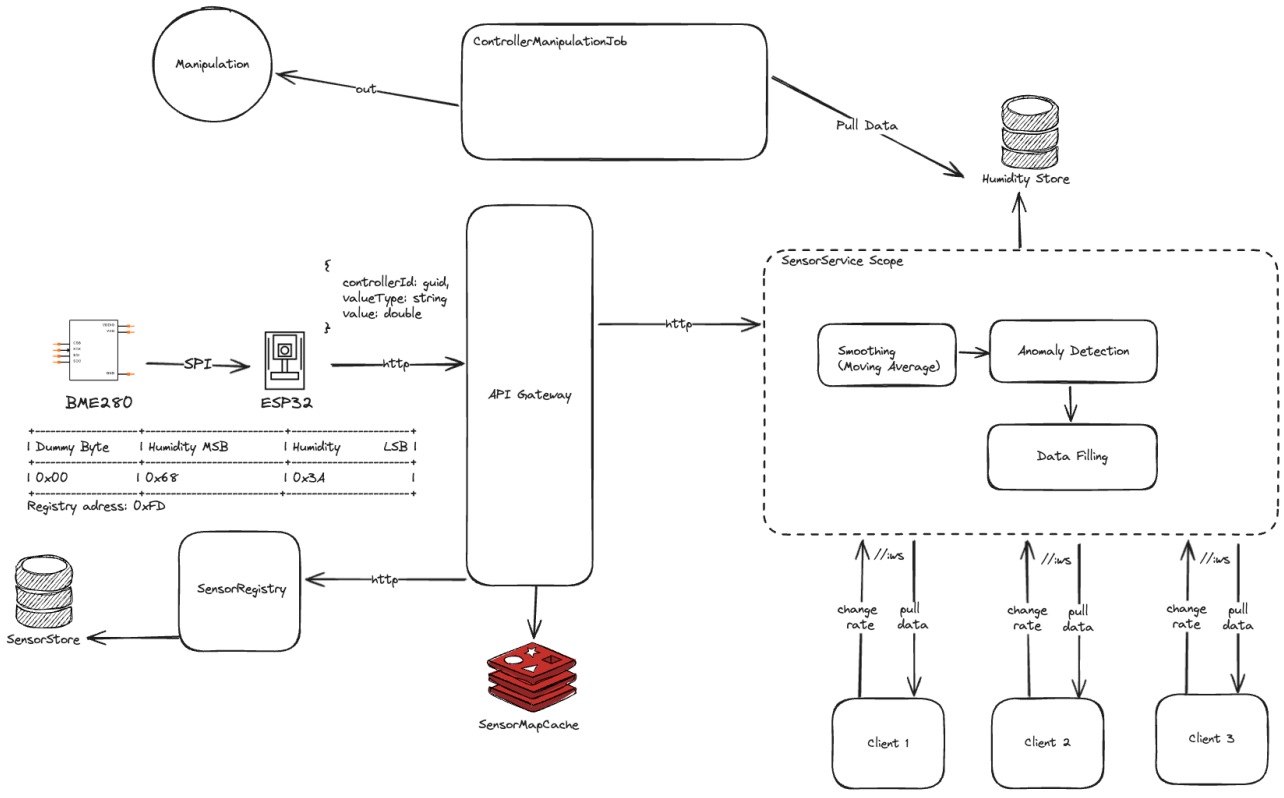

Рисунок 4.4 – Схема роботи системи збору даних про вологість

Система моніторингу температури працює подібним чином, але використовує датчик DHT22, який вимірює температуру та вологість одночасно. Дані з DHT22 передаються до ESP32 через інтерфейс SPI. Наприклад, вихідний сигнал датчика може містити байти 0x00, 0x68, 0x3A, де перший байт є dummy byte, а наступні байти містять значення температури.

ESP32 також конвертує ці дані у формат JSON і відправляє на API-шлюз. Типове повідомлення для температури може виглядати так:

```
{
    "controllerId": "guid",
    "valueType": "temperature",
    "value": 23.0
}
```

API-шлюз приймає ці дані і передає їх до компонента обробки даних, який виконує згладжування та видалення аномалій. Після обробки дані зберігаються у базі даних (Temperature Store) для подальшого аналізу.

Повна схема роботи системи збору даних про вологість зображена на рисунку 4.5.

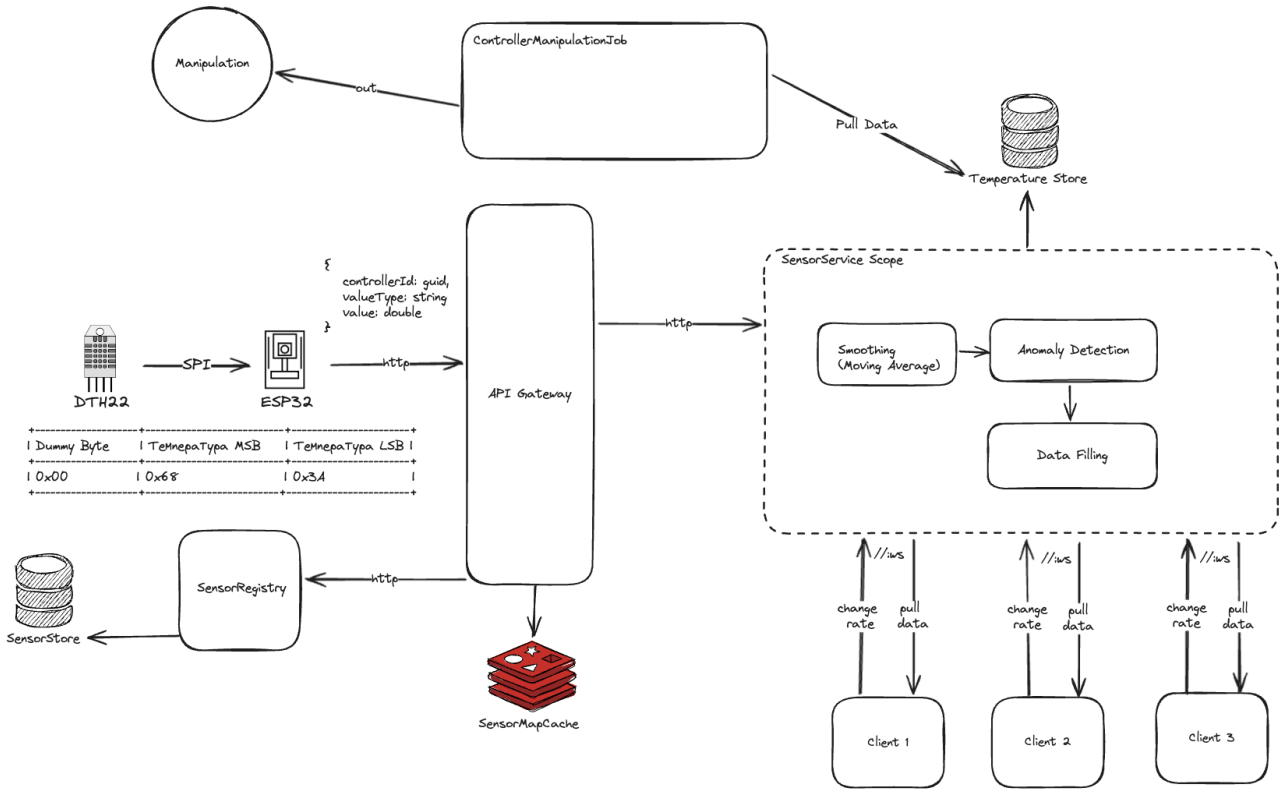

Рисунок 4.5 – Схема роботи системи збору даних про температуру

Для забезпечення ефективного збору даних у системі реалізовано спеціальний процес, який по частинах витягує дані з бази даних. Цей процес виконується в декількох потоках, що дозволяє значно підвищити продуктивність і надійність системи збору даних.

Реалізація цього сервісу зображена в даному лістингу:

```
using System;
using System.Collections.Concurrent;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;

public class DataCollector
{
    private ConcurrentBag<SensorData> _collectedData;
    private List<Sensor> _sensors;
    private int _numThreads;

    public DataCollector(List<Sensor> sensors, int numThreads)
    {
        _collectedData = new ConcurrentBag<SensorData>();
        _sensors = sensors;
        _numThreads = numThreads;
    }

    public void StartCollection(CancellationToken cancellationToken)
    {
        var tasks = new List<Task>();

        foreach (var sensor in _sensors)
        {
            var task = Task.Run(() => CollectDataFromSensor(sensor,
cancellationToken));
            tasks.Add(task);
        }

        Task.WhenAll(tasks).Wait();
    }
```

```
    private void CollectDataFromSensor(Sensor sensor, CancellationToken
cancellationToken)
    {
        while (!cancellationToken.IsCancellationRequested)
        {
            var data = sensor.ReadData();
            _collectedData.Add(data);

            Thread.Sleep(sensor.ReadInterval);
        }
    }

    public List<SensorData> GetCollectedData()
    {
        return _collectedData.ToList();
    }
}

public class Sensor
{
    public int ReadInterval { get; set; }
    public SensorData ReadData()
    {
    }
}

public class SensorData
{
    public DateTime Timestamp { get; set; }
    public double Temperature { get; set; }
    public double Humidity { get; set; }
    public double Light { get; set; }
}
```

На основі аналізу даних система генерує звіти для адміністраторів системи. Звіти можуть включати візуалізації даних, такі як графіки і теплові карти, які показують розподіл різних параметрів по будівлі. Рекомендації можуть включати пропозиції щодо оптимізації параметрів або планів обслуговування обладнання.

Використовуючи ASP.NET Core та WebSocket для створення реактивного веб-інтерфейсу, ми забезпечуємо, що дані від датчиків обробляються і відображаються в реальному часі. Для більш глибокої аналітики та довготривалого зберігання, дані відправляються до центральної бази даних SQL Server, де вони нормалізуються.

На основі аналізу даних, наша система не тільки надає докладні звіти про стан середовища в будівлі, але й автоматично адаптує різні системи (опалення, охолодження, освітлення) для оптимального комфорту та енергоефективності. Всі ці дії засновані на даних, що забезпечують максимальну продуктивність та задоволення користувачів будівлі.

Цей підхід не тільки покращує умови праці, але й сприяє зниженню витрат на утримання будівлі, демонструючи значні переваги інтеграції передових технологій в управління будівлями.

4.4 Управління даними в системі

Система керування параметрами навколишнього середовища використовує комплексну архітектуру для забезпечення точного моніторингу та регулювання температури, вологості та освітленості. Основними компонентами цієї системи є датчики, контролери, сервіс збору та обробки даних, а також сервіс управління контролерами.

На початку, датчики, розташовані в приміщенні, вимірюють відповідні параметри навколишнього середовища. Дані з датчиків передаються до мікроконтролера ESP32 через SPI інтерфейс. Для прикладу, датчик BME280 вимірює вологість та передає значення у форматі байтів 0x00, 0x68, 0x3A, де перший байт є dummy byte, а наступні два байти містять значення вологості. Аналогічно, датчик BH1750 вимірює освітленість і передає значення у форматі байтів 0x00, 0x34, 0x12.

ESP32 отримує ці дані, конвертує їх у формат JSON і відправляє на API-шлюз за допомогою HTTP-запитів.

API-шлюз приймає ці дані і передає їх далі до компонента обробки даних (SensorService Scope). Цей компонент виконує згладжування даних методом ковзного середнього (Moving Average) для видалення шумів та аномалій. Після аналізу дані зберігаються у відповідній базі даних (Temperature Store, Humidity Store).

Після обробки та збереження даних, сервіс RoomMetrics Service отримує запити на оптимальні параметри для конкретного приміщення. Наприклад, запит на оптимальну температуру виглядає так:

```
GET /GetOptimal?roomId
```

Відповідь містить оптимальні значення параметрів для приміщення, збережені в системі. Наприклад, відповідь на запит може виглядати так:

```
{
    "roomId": "guid",
    "valueType": "temperature",
    "optimalValue": 22.0
}
```

Ці значення використовуються для регулювання параметрів середовища. Контролери отримують команди на основі цих оптимальних значень. Команда на зміну значення надсилається з API-шлюзу до сервісу управління контролерами (ControllerManagement Service).

ControllerManagement Service перевіряє значення параметрів і відправляє відповідні команди до контролерів ESP32. Команди надсилаються через UART. Наприклад, команда для увімкнення кондиціонера виглядає так:

```
{
    "command": "ON",
    "value": 22
}
```

ESP32 отримує команду через UART, обробляє її і передає сигнал до кондиціонера або іншого пристрою. Якщо потрібно змінити значення температури, команда для ESP32 може виглядати наступним чином:

```
ON,22\n
```

Це означає, що кондиціонер увімкнено і встановлено температуру на 22 градуси.

Зміна вологості відбувається аналогічно. Датчик BME280 вимірює вологість, передає значення через SPI на ESP32, який конвертує ці дані у JSON формат і відправляє через HTTP на API-шлюз. Після обробки даних у SensorService Scope, ці дані зберігаються у Humidity Store. Команди для зволожувачів або осушувачів повітря також надсилаються через UART з використанням аналогічних форматів.

Для освітлення використовується датчик BH1750, який передає дані про рівень освітленості. ESP32 отримує ці дані через SPI і надсилає їх у вигляді JSON-повідомлень на API-шлюз. Після обробки та згладжування даних у SensorService Scope, ці дані зберігаються у відповідній базі даних. Команди для зміни рівня освітленості надсилаються аналогічно через UART, наприклад:

```
LIGHT,500\n
```

Це означає, що рівень освітленості має бути встановлений на 500 люкс.

Діаграми компонентів системи показують, як кожен компонент взаємодіє з іншими, забезпечуючи безперервний збір, обробку та управління даними в реальному часі. Це дозволяє підтримувати оптимальні умови в приміщеннях, що сприяє підвищенню комфорту та продуктивності. Діаграма роботи системи зображена на рисунку 4.6

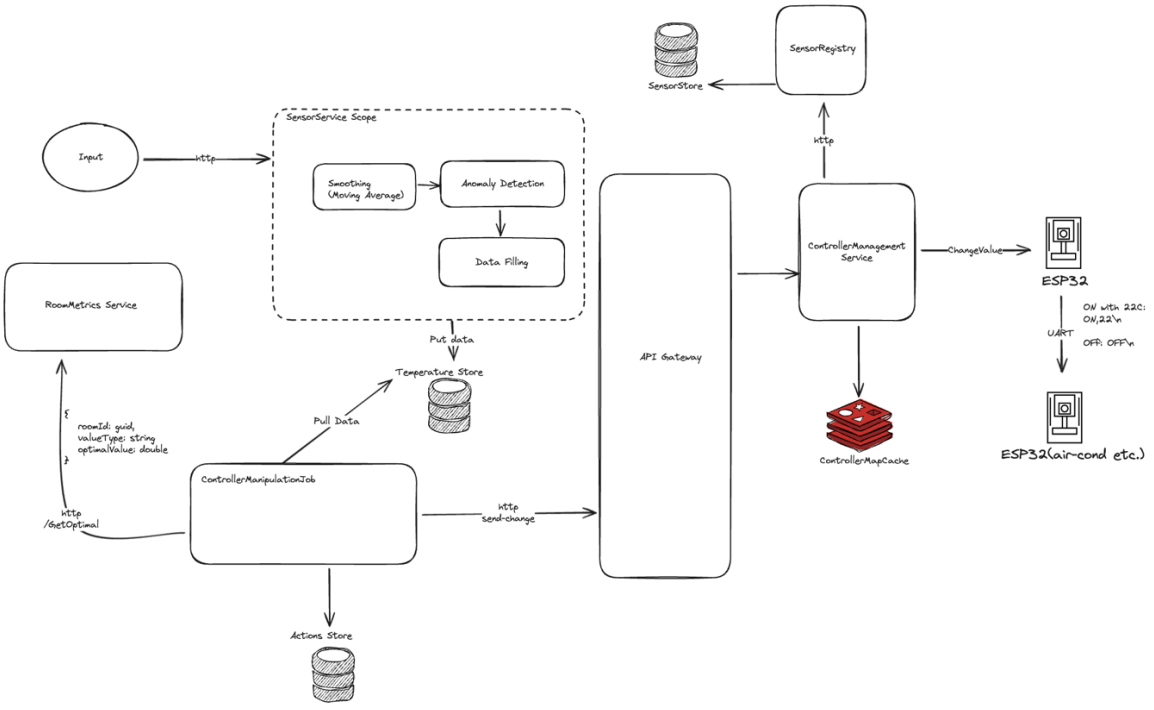

Рисунок 4.6 – Схема роботи системи управління даними

Таким чином, система забезпечує повний цикл моніторингу та управління параметрами навколишнього середовища. Всі компоненти взаємодіють через стандартизовані інтерфейси, що забезпечує надійність та гнучкість системи. Важливим елементом є згладжування та виявлення аномалій, які дозволяють підтримувати точність та надійність вимірювань.

ВИСНОВКИ

У цій роботі були розроблено комплексну програмну систему для моніторингу екологічних показників в робочих приміщеннях.

В роботі було обґрунтовано вибір мови програмування C#, а також вибір СУБД MS SQL Server.

Було проведено аналіз об’єкту, що потребував автоматизації роботи, проведено функціональне моделювання, створено UML-діаграми (діаграма прецедентів, діаграма класів, діаграми послідовностей, діаграма активності).

Система задовольняє потреби різноманітних зацікавлених сторін, включаючи мешканців, менеджерів об’єктів, обслуговуючий персонал, персонал з безпеки.

Проєкт передбачає детальний аналіз і класифікацію користувачів на три окремі групи: авторизовані користувачі, неліцензовані користувачі та контролери. Кожен із користувачів був деталізований, щоб переконатися, що він має правильні права доступу та правильну роль в управлінні IoT.

Протягом усього процесу розробки застосовувався комплексний підхід, включаючи імітацію інтеграції датчиків, збір і обробку даних у режимі реального часу, механізми оповіщення, можливості віддаленого моніторингу та контролю, реєстрацію історичних даних тощо. Ці функціональні вимоги були ретельно розроблені для вирішення конкретних потреби та обов'язки кожної групи користувачів.

Завдяки проведеному аналізу предметної галузі і визначенню основних функціональних вимог був розроблений повністю працездатний застосунок, що задовольняє усім вимогам щодо автоматизації процесу управління середовищем робочих кімнат, спрощення процесу управління та ведення статистики.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Building scalable cloud databases: вебсайт. URL: https://docs.microsoft.com/en-us/azure/azure-sql/database/elastic-database-client-library/ (дата звернення 21.03.2024).

2. Multi-Tenancy in ASP.NET Core : вебсайт. URL: https://docs.abp.io/en/abp/latest/Multi-Tenancy (дата звернення 07.03.2024)

3. Грабер М. SQL. Вид. 2-е. : ЛОРИ, 2003. 644 с.

4. «Екстремальне програмування: розробка через тестування» [Електронне видання] / Автор: Кент Бек – 2019. – 224с.

5. «Чиста архітектура. Мистецтво розробки програмного забезпечення» [Електронне видання] / Автор: Роберт Мартін – 2018. – 352 с.

6.

7. Smith J. Entity Framework Core in Action: Pakt Publishing, 2017. 247 p.

8. Russ U. A Project Guide to UX Design: For user experience designers in the field or in the making. 2nd Edition : New Riders, 2012 361 p.

9. Maklin H. Adaptive Code: Agile coding with design patterns and SOLID principles, 2nd Edition : Williams, 2017. 448 p.

10. Huang J. J., Ong C. S., Tzeng G. H. Optimal fuzzy multi-criteria expansion of competence sets using multi-objectives evolutionary algorithms. // Expert Systems with Applications. 2016. № 30. P. 739 – 745.

11. Belton V., Stewart T. J. Multiple criteria decision analysis: An integrated approach. – Boston: Kluwer Academic Publishers, 2012. 167 p.

11. Rosalie J.O. Assessing the Readiness of Firms for CRM: A Literature Review and Research Model, 2010. – 354 p.

12. Li, S., Da Xu, L., & Zhao, S. (2015). The Internet of Things: A Survey. Information Systems Frontiers, 17(2), 243–259.

13. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645–1660.

14. Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. Computer Networks, 54(15), 2787–2805.

15. Haller, S., Karnouskos, S., & Schroth, C. (2009). The Internet of Things in an Enterprise Context. In P. Barnaghi, S. Meissner, & M. Presser (Eds.), Enterprise Interoperability (pp. 14–28). Springer, London.

16. Vermesan, O., & Friess, P. (Eds.). (2014). Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems. River Publishers.

17. Yang, L. T., Jin, H., & Li, X. (2011). An Integrated Development Environment for Internet of Things. IEEE Transactions on Industrial Informatics, 7(2), 179–190.

18. Stankovic, J. A. (2014). Research Directions for the Internet of Things. IEEE Internet of Things Journal, 1(1), 3–9.

19. Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of Things for Smart Cities. IEEE Internet of Things Journal, 1(1), 22–32.

20. Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2014). Context Aware Computing for The Internet of Things: A Survey. IEEE Communications Surveys & Tutorials, 16(1), 414–454. https://doi.org/10.1109/SURV.2013.042313.00197

21. Bassi, A., Bauer, M., Fiedler, M., Kramp, T., & van Kranenburg, R. (Eds.). (2013). Enabling Things to Talk: Designing IoT solutions with the IoT Architectural Reference Model. Springer.

22. Mike Cohn, Agile Estimating and Planning : Prentice Hall, 2015. 368 p.

23. Андрессон Е., Грінспун Ф., Грумет А. Проєктування інтернет-застосунків. Масачусетс: MIT Press, 2006. 401 с.

24. «Spring Microservices in Action 1st Edition» [Електронне видання] / Автор: John Carnell - 2017. – 384с.

25. «Head First Object-Oriented Analysis and Design 1st Edition» [Електронне видання] / Автори: Brett D. McLaughlin , Gary Pollice , Dave West - 2006. – 636с.