ЛАБОРАТОРНА РОБОТА №2.

РОЗРОБКА БАЗИ ДАНИХ ДЛЯ СЕРВЕРНОЇ ЧАСТИНИ ПРОГРАМНОЇ СИСТЕМИ ТА ПРИКЛАДНОГО ПРОГРАМНОГО ІНТЕРФЕЙСУ (API).

Система моніторингу опалення у домогосподарствах

Версія 1.0, затверджена

Підготовлено Макогоном Б.О.

ПЗПІ-22-6

30.12.2024

У цьому проекті було прийнято низку рішень для забезпечення якісного проектування REST API. Основним принципом стало ресурсно-орієнтоване проектування, де кожна сутність, така як `Sensor`, `User`, `UserSetting` чи `SensorData`, отримала свій власний набір REST-ендпоїнтів.

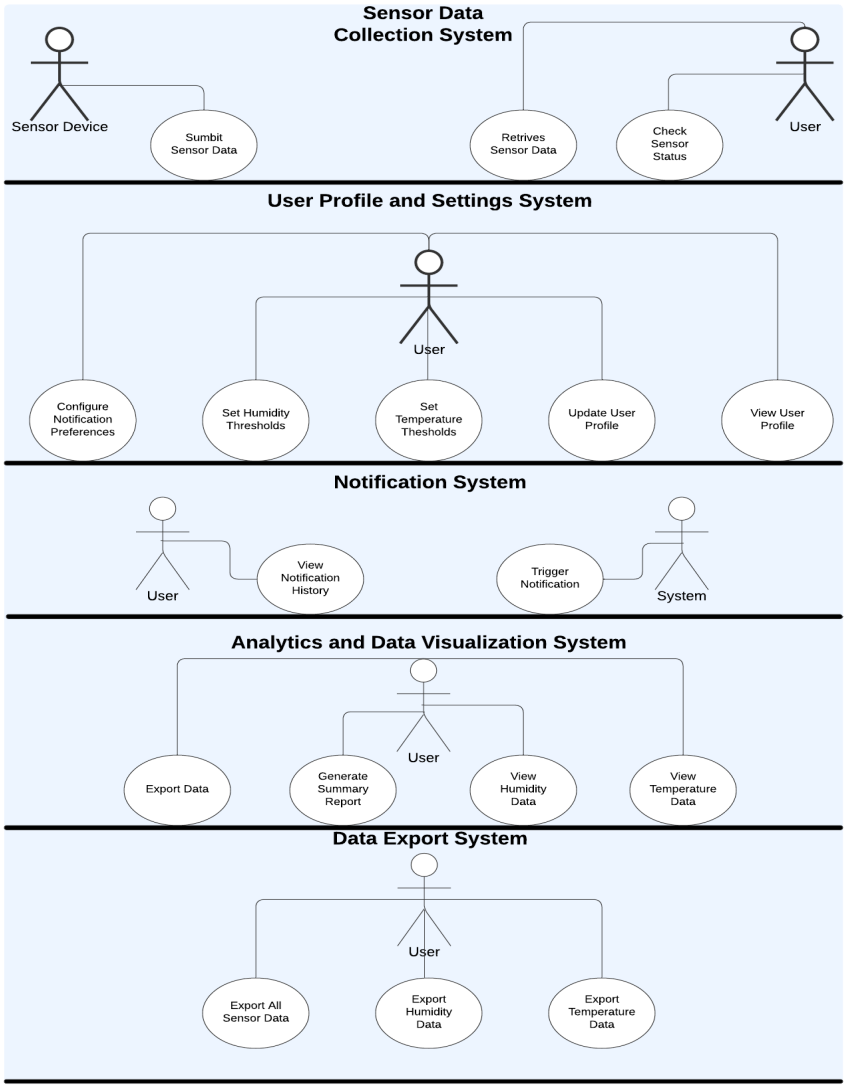

Рисунок 1. Use case diagram

Це відповідає концепції ресурсної орієнтації та дозволяє легко взаємодіяти з ресурсами через стандартизовані URI. Наприклад, `/api/resource` використовується для доступу до колекцій, а `/api/resource/{id}` — для роботи з конкретними елементами. Усі операції базуються на стандартних HTTP-методах:

GET для отримання даних, POST для створення, PUT для оновлення та DELETE для видалення.

Для ізоляції бізнес-логіки було реалізовано сервісний рівень, який відповідає за основні операції, тоді як контролери виконують лише маршрутизацію запитів і передачу даних між API та логікою.

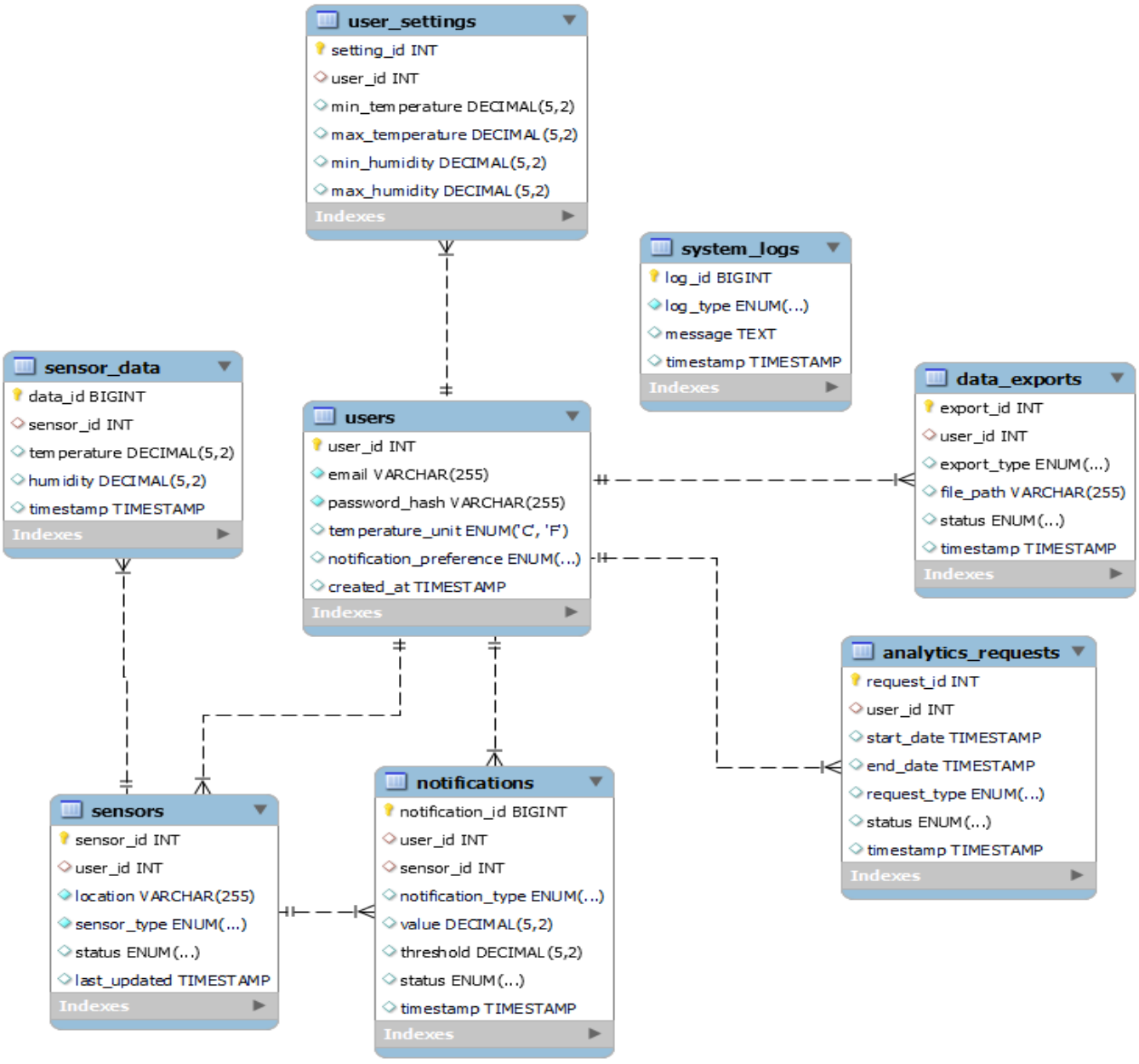

Рисунок 2. Heating ER model

Наприклад, контролер `SensorController` приймає запити, передає їх у `SensorService`, який виконує такі дії, як перевірка валідності, генерація даних чи збереження їх у базу даних.

У проекті активно використовуються DTO-класи (Data Transfer Objects), які забезпечують абстрагування внутрішньої структури бази даних від зовнішнього API. Наприклад, `SensorDTO` включає лише необхідні дані, такі як `id`, `location`, і `sensorType`, що спрощує обмін інформацією між клієнтом і сервером. Для перевірки вхідних даних застосовано механізм валідації за допомогою Java Bean Validation. Це дозволяє забезпечити, що тільки коректні дані передаються до сервісів, а помилки обробляються ще на рівні контролерів.

Окрема увага була приділена роботі із залежностями між ресурсами. Наприклад, сенсори (`Sensor`) мають зв’язок `ManyToOne` із користувачами (`User`), а дані сенсорів (`SensorData`) — зв’язок `ManyToOne` із сенсорами. Для доступу до залежних ресурсів реалізовано спеціальні ендпоїнти, такі як отримання всіх сенсорів для конкретного користувача за запитом `/api/sensors/user/{userId}`.

API спроектоване таким чином, щоб бути масштабованим і легким у підтримці. Кожен ресурс реалізовано незалежно, що дозволяє легко розширювати функціонал, додаючи нові ресурси чи операції без змін у вже існуючому коді. Чітка структура URL забезпечує інтуїтивно зрозумілу інтеграцію з фронтендом і сторонніми системами. Усі операції супроводжуються використанням стандартних HTTP-кодів статусів, таких як `200 OK` для успішних запитів, `400 Bad Request` для помилок клієнта та `404 Not Found` для ненайдених ресурсів.

У проекті також реалізовано автоматичне планування завдань. Наприклад, система генерує дані для сенсорів кожні п’ять секунд, що дозволяє моделювати

реальний збір даних. Дані зберігаються в таблиці бази даних `sensor_data`, що забезпечує доступність для подальшої аналітики.

Для користувачів реалізовано можливість налаштування параметрів через API `UserSetting`. Це дозволяє встановлювати порогові значення температури та вологості, які інтегруються з іншими функціями системи. Усі ці рішення разом створюють зручний, масштабований і надійний REST API, який відповідає вимогам сучасних веб-додатків.