ЛАБОРАТОРНА РОБОТА №3.

РОЗРОБКА БІЗНЕС-ЛОГІКИ ТА ФУНКЦІЙ АДМІНІСТРУВАННЯ СЕРВЕРНОЇ ЧАСТИНИ ПРОГРАМНОЇ СИСТЕМИ

Система моніторингу опалення у домогосподарствах

Версія 1.0, затверджена

Підготовлено Макогоном Б.О.

ПЗПІ-22-6

15.11.2024

Бізнес-логіка керування користувачами та сенсорами в цьому проекті побудована за класичним тришаровим підходом: **контролери** обробляють вхідні запити, **сервіси** виконують бізнес-логіку, а **репозиторії** забезпечують взаємодію з базою даних. Це дозволяє ізолювати різні аспекти роботи системи, роблячи її гнучкою, масштабованою та простою в підтримці.

Керування користувачами

Бізнес-логіка для користувачів починається з автентифікації. Коли користувач намагається увійти в систему, він відправляє свої дані (email і пароль) через форму. Контролер `UserController` приймає ці дані через метод, наприклад, `login()`. Контролер виконує базову валідацію вхідних даних (наприклад, перевірку на порожні поля), після чого передає дані до сервісу `UserService`.

У `UserService` відбувається основна логіка автентифікації. Сервіс звертається до репозиторію `UserRepository`, щоб знайти користувача за email-адресою. Якщо користувача не знайдено, сервіс повертає помилку про невірні облікові дані. Якщо користувач знайдений, відбувається перевірка пароля. Для цього хеш пароля, що зберігається в базі даних, порівнюється з хешем введеного пароля. Якщо пароль правильний, користувач вважається автентифікованим, і об'єкт `UserDTO` передається назад до контролера. Контролер, у свою чергу, повертає клієнту відповідь із токеном сесії або перенаправляє його на сторінку, доступну лише для авторизованих користувачів.

Процес реєстрації користувачів працює за схожим принципом. Контролер `UserController` приймає дані нового користувача (email, пароль, інші поля) і передає їх до сервісу `UserService`. Сервіс перевіряє, чи email вже існує в базі даних, використовуючи метод `existsByEmail()` у `UserRepository`. Якщо email унікальний, пароль хешується за допомогою алгоритму (наприклад, BCrypt), після чого створюється новий об'єкт `User`, який зберігається в базу даних через `UserRepository`. Успішно зареєстрований користувач отримує відповідь із підтвердженням реєстрації.

Керування сенсорами

Сенсори тісно пов'язані з користувачами, тому кожен сенсор прив'язаний до конкретного користувача через зв'язок `ManyToOne`. Додавання нового сенсора починається з форми на клієнтській стороні, де користувач вказує дані про

сенсор, такі як його локація та тип. Контролер `SensorController` приймає цей запит і передає дані до сервісу `SensorService`.

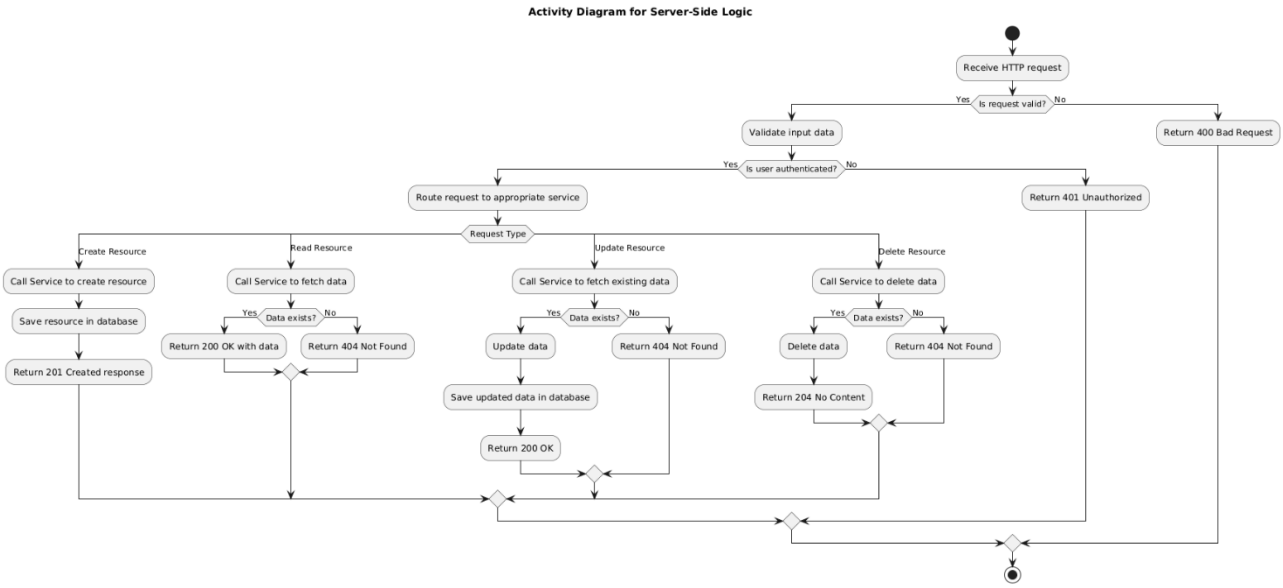

Рисунок 1. UML діаграма діяльності для серверної частини

У `SensorService` відбувається перевірка, чи існує користувач із зазначеним ID, використовуючи метод `findById()` у `UserRepository`. Якщо користувача не знайдено, сервіс повертає помилку. Якщо користувач існує, створюється новий об'єкт `Sensor`, у якому встановлюються зв’язки з користувачем і параметри сенсора. Об'єкт зберігається в базі даних через `SensorRepository`.

Оновлення сенсора реалізується подібним чином. Коли користувач хоче змінити дані сенсора, запит із новими значеннями надходить до контролера `SensorController`, який передає дані до `SensorService`. Сервіс спочатку перевіряє наявність сенсора в базі даних, а потім оновлює його поля, які були змінені. Після цього оновлений сенсор зберігається через `SensorRepository`.

Видалення сенсора також включає перевірку його існування в базі. Якщо сенсор існує, він видаляється методом `deleteById()` у `SensorRepository`.

Робота з даними сенсорів

Дані, що збираються сенсорами, зберігаються в таблиці `sensor_data`. Генерація цих даних відбувається автоматично за допомогою запланованого завдання (scheduler), яке кожні 5 секунд викликає сервіс `SensorDataGenerationService`. Сервіс отримує всі сенсори через `SensorRepository` і для кожного генерує

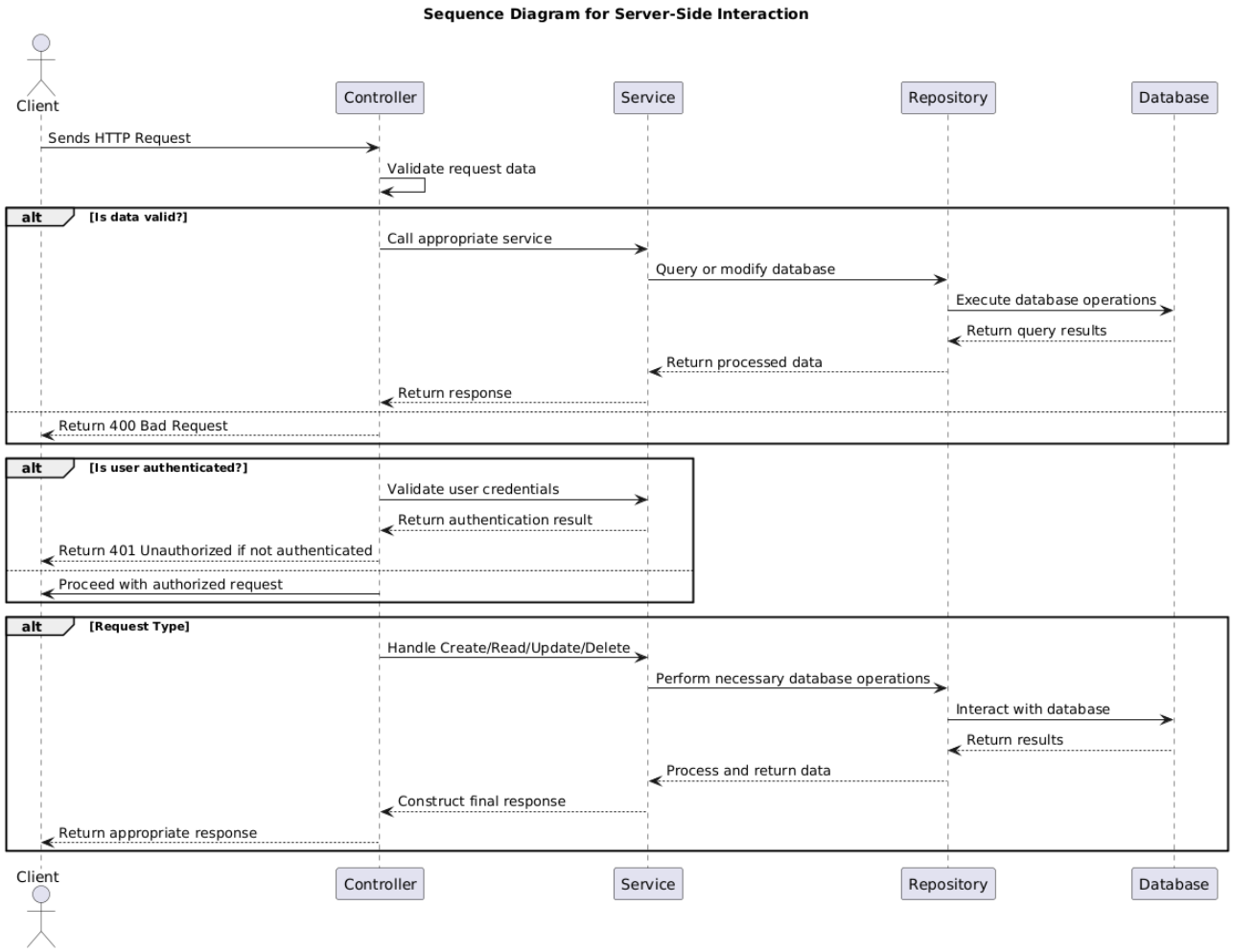

Рисунок 2 . UML діаграма взаємодії для серверної частини.

випадкові значення температури та вологості. Ці дані зберігаються в таблиці через `SensorDataRepository`.

Користувач може отримати дані для конкретного сенсора через API-запит до ендпоїнту `/api/sensors/{id}/data`. Контролер `SensorController` викликає сервіс `SensorService`, який отримує всі дані для сенсора з бази даних, використовуючи `SensorDataRepository`. Дані повертаються у вигляді списку DTO, які містять інформацію про температуру, вологість і час вимірювання.

Користувацькі налаштування

Для кожного користувача існує можливість встановлювати порогові значення температури та вологості через сутність `UserSetting`. Налаштування пов’язані з користувачем через зв'язок `OneToOne`. Коли користувач хоче переглянути чи змінити свої налаштування, контролер `UserSettingController` отримує їх через

`UserSettingService`. Сервіс, у свою чергу, звертається до `UserSettingRepository`, який виконує запит на основі ID користувача.

Оновлення налаштувань працює схожим чином. Нові значення порогів передаються в контролер, потім у сервіс, де вони перевіряються, і зрештою зберігаються в базу даних через `UserSettingRepository`.

Інтеграція логіки

Уся бізнес-логіка проекту орієнтована на інтеграцію користувачів із сенсорами та даними. Наприклад, після встановлення користувацьких налаштувань система може перевіряти, чи перевищують зібрані дані порогові значення, і надсилати сповіщення. Така інтеграція забезпечує високий рівень автоматизації та взаємодії між компонентами системи.

Цей підхід дозволяє легко додавати нові функціональні можливості, наприклад, інші типи сенсорів або складніші правила обробки даних, зберігаючи чітку й логічну структуру коду.