Міністерство освіти і науки України Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

Звіт

З лабораторної роботи №5

З дисципліни «Аналіз та рефакторинг коду»

на тему: «РОЗГОРТАННЯ ПРОГРАМНОЇ СИСТЕМИ ТА ДЕМОНСТРАЦІЯ ЇЇ РОБОТИ»

Виконавець:

ст. гр. ПЗПІ-22-4 Попов Б. С.

Перевірив:

ст. викладач кафедри ПІ Сокорчук І. П.

Харків 2024

# Мета роботи:

Розгорнути реалізовану програмну систему, перевірити та налаштувати роботу розгорнутої програмної системи.

# Хід роботи:

* 1. **Опис розгорнутої програмної системи**

Серверна частина:

• Мова програмування: Серверна частина розроблена на JavaScript. Фреймворк: Для розробки веб-додатків та REST API використовується Node.js разом із Express.js. Цей технологічний стек забезпечує високопродуктивний обмін даними, необхідний для моніторингу стану IoT-пристроїв у реальному часі.

• База даних: Для зберігання та управління даними, такими як температура двигуна, тиск у шинах і GPS-координати, використовується MongoDB. Моделі та схеми для роботи з базою даних реалізовані за допомогою бібліотеки Mongoose, що забезпечує зручну взаємодію з MongoDB.

• Контролери та маршрутизація: Для обробки HTTP-запитів реалізовано контролери, що забезпечують ефективну маршрутизацію запитів та взаємодію з основними ресурсами системи, такими як методи GET, POST, PUT і DELETE.

• Міграції та оновлення: Для ініціалізації бази даних та додавання початкових даних використовуються міграційні скрипти, що автоматизують процес налаштування бази.

Функціональність: Система забезпечує збір, обробку та аналіз даних про стан IoT-пристроїв, зокрема температуру двигуна, тиск у шинах і геолокацію. Крім того, реалізовано відправлення цих даних на MQTT-брокер для подальшої передачі їх до інших систем для обробки.

Інтеграція з IoT: Система інтегрована з IoT-пристроями через MQTT-брокер, що дозволяє отримувати дані з датчиків транспортних засобів у реальному часі та обробляти ці дані на сервері. Для комунікації між IoT-пристроєм і сервером використовується MQTT-протокол, що гарантує стабільність і ефективність передачі даних.

# Опис розгортання

Серверна частина:

1. Інсталяція залежностей: Всі необхідні залежності були встановлені за допомогою команди npm install.
2. Налаштування середовища: У файлі .env було налаштовано середовище, зокрема підключення до MQTT-сервера для обміну даними.
3. Запуск сервера: Сервер було запущено за допомогою команди npm start. Для тестування функціональності використовувалися запити до API через Postman.
4. Перевірка функціональності:
   * Запити до API обробляються коректно.
   * Дані успішно зберігаються в базі даних для подальшого аналізу.
   * Система належно передає та отримує дані через MQTT від IoT-пристроїв.

Інтеграція IoT:

1. Підключення ESP32 до Wi-Fi та налаштування передачі даних на сервер: Мікроконтролер ESP32 був налаштований для підключення до Wi-Fi та передачі даних через MQTT.
2. Передача зібраних даних через MQTT до сервера: Дані, зібрані від сенсорів (температура двигуна, тиск у шинах, геолокація), передаються через MQTT на сервер для подальшої обробки.
3. Обробка та зберігання даних: Сервер успішно обробляє отримані дані та зберігає їх у базі даних MongoDB для подальшого аналізу.

Використані інструменти та технології:

* Node.js: Використовувався для розробки серверної частини.
* MongoDB: Використовувалася для зберігання даних.
* Postman: Інструмент для тестування API.
* MQTT: Використовувався для з'єднання IoT-пристроїв із сервером.
* Visual Studio Code: Використовувався для розробки та налагодження коду.
* GitHub: Використовувався для керування версіями коду.

# Висновок:

Програмна система була успішно розгорнута та протестована в локальному середовищі, де всі її компоненти працюють без помилок і забезпечують стабільну роботу. Основні функціональні можливості системи включають:

* Обробка даних від IoT-пристроїв: Система здійснює збір і обробку реальних даних, що надходять від IoT-сенсорів, таких як температура двигуна, тиск у шинах і геолокація, забезпечуючи точність і своєчасність інформації.
* Генерація та надсилання нагадувань користувачам: Автоматизована система нагадувань дозволяє користувачам своєчасно отримувати інформацію про необхідність технічного обслуговування або інших важливих дій.
* Зберігання даних у базі даних MongoDB: Всі зібрані дані зберігаються в базі даних MongoDB, що забезпечує їх безпечне збереження та легкий доступ для подальшого аналізу та звітності.

Система побудована з урахуванням можливості майбутнього розгортання на хмарному сервері, що дозволить значно підвищити масштабованість та доступність. Хмарне розгортання забезпечить можливість обробки більших обсягів даних та гарантує високу доступність сервісу для користувачів в будь-який час.

У майбутньому планується додатковий розвиток функціональності системи, включаючи інтеграцію з іншими платформами для покращення користувацького досвіду та розширення можливостей моніторингу транспортних засобів.