Міністерство освіти та науки України Харківський національний університет радіоелектроніки

Кафедра ПІ

Звіт

З лабораторної роботи 3

Тема роботи: «РОЗРОБКА БІЗНЕС-ЛОГІКИ ТА ФУНКЦІЙ АДМІНІСТРУВАННЯ СЕРВЕРНОЇ ЧАСТИНИ ПРОГРАМНОЇ СИСТЕМИ»

з дисципліни «Аналіз та рефакторинг коду»

Виконав: Перевірив:

ст. гр. ПЗПІ-22-10 ст. викладач Сокорчук І.П.

Клецов М.Д.

Харків 2024

**Мета роботи:** На лабораторній роботі №2 потрібно розробити бізнес-логику та функції адміністрування серверної частини програмної системи.

**Хід роботи:**

Під час виконання лабораторної роботи була створена серверна частина програмної системи для аналізу авто-трафіку. Реалізовано функції адміністрування. Система підтримує розмежування ролей, що дозволяє адміністраторам переглядати списки користувачів, додавати та видаляти записи, а також оновлювати дані в базі.

Для забезпечення безпеки реалізовано механізм автентифікації та авторизації з використанням JWT токенів. Додатково впроваджено middleware для перевірки прав доступу, що обмежує виконання адміністративних функцій лише для користувачів із відповідними правами.

Взаємодія з базою даних здійснюється через запити для роботи з таблицями користувачів засобів. Проведено тестування всіх функцій для перевірки їхньої працездатності та стабільності.

1. **Текстовий опис інженерних рішень**

Серверна частина системи аналізу авторафіку побудована на основі Node.js з використанням бази даних PostgreSQL. Основні компоненти включають маршрути для обробки запитів, контролери для реалізації бізнес-логіки та моделі для взаємодії з базою даних.

Маршрути забезпечують доступ до функцій керування, такі як перегляд, додавання, оновлення та видалення записів. Контролери обробляють дані, виконуючи SQL-запити для роботи з таблицями, такими як " Sensors" та " TrafficJams".

Для безпеки реалізовано механізми автентифікації та авторизації за допомогою JWT токенів, а middleware перевіряють права доступу до адміністративних функцій.

1. **Фрагменти програмного коду:**

* **Бізнес логіки (математичних методів обробки даних) серверної частини;**

Функція для додавання маршруту:

**const express = require("express");**

**const router = express.Router();**

**const db = require("../config/db");**

**const { authenticateUser, authorizeRole } = require("../middlewares/authMiddleware");**

**const User = require("../models/userModel");**

**// 📌 Отримати всіх користувачів (доступно всім)**

**router.get("/", authenticateUser, async (req, res) => {**

**try {**

**const result = await db.query("SELECT \* FROM users ORDER BY id ASC");**

**res.json(result.rows);**

**} catch (error) {**

**res.status(500).json({ message: error.message });**

**}**

**});**

**// 📌 Створити нового користувача (admin, superadmin)**

**router.post("/", authenticateUser, authorizeRole("admin", "superadmin"), async (req, res) => {**

**try {**

**const { username, email, password } = req.body;**

**const newUser = await User.create(username, email, password);**

**res.status(201).json(newUser);**

**} catch (error) {**

**res.status(400).json({ message: error.message });**

**}**

**});**

**// 📌 Оновити користувача (тільки superadmin)**

**router.put("/:id", authenticateUser, authorizeRole("superadmin"), async (req, res) => {**

**try {**

**const { id } = req.params;**

**const { username, email, password } = req.body;**

**const updatedUser = await User.update(id, username, email, password);**

**res.json(updatedUser);**

**} catch (error)**

**{**

**res.status(400).json({ message: error.message });**

**}**

**});**

**// 📌 Видалити користувача (тільки superadmin)**

**router.delete("/:id", authenticateUser, authorizeRole("superadmin"), async (req, res) => {**

**try {**

**const { id } = req.params;**

**await User.delete(id);**

**res.json({ message: "Користувача успішно видалено" });**

**} catch (error) {**

**res.status(500).json({ message: error.message });**

**}**

**});**

**module.exports = router;**

* **Адміністрування серверної частини.**

Функція для показу та видалення всіх користувачів (admin only):

const express = require("express");

const User = require("../models/userModel");

const router = express.Router();

// 📌 Отримати всіх користувачів

router.get("/", async (req, res) => {

try {

const users = await User.getAll();

res.json(users);

} catch (error) {

res.status(500).json({ message: error.message });

}

});

// 📌 Додати нового користувача

router.post("/", async (req, res) => {

try {

const { username, email, password } = req.body;

const newUser = await User.create(username, email, password);

res.status(201).json(newUser);

} catch (error) {

res.status(400).json({ message: error.message });

}

});

// 📌 Оновити інформацію про користувача

router.put("/:id", async (req, res) => {

try {

const { id } = req.params;

const { username, email, password } = req.body;

const updatedUser = await User.update(id, username, email, password);

res.json(updatedUser);

} catch (error) {

res.status(400).json({ message: error.message });

}

});

// 📌 Видалити користувача

router.delete("/:id", async (req, res) => {

try {

const { id } = req.params;

await User.delete(id);

res.json({ message: "Користувача успішно видалено" });

} catch (error) {

res.status(500).json({ message: error.message });

}

});

module.exports = router;

**Висновки:** У процесі розробки серверної частини системи аналізу авто-трафіку було реалізовано функції для управління користувачами, а також адміністрування доступу за допомогою ролей. Використання бази даних PostgreSQL дозволило ефективно зберігати та маніпулювати даними про користувачів. Впровадження механізмів автентифікації та авторизації за допомогою JWT токенів забезпечило безпеку доступу до адміністративних функцій.

**Посилання на відео:** https://youtu.be/fuQ8qAbgdpw

ДОДАТОК А

UML діаграма діяльності

Изображение выглядит как снимок экрана, текст, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1 – UML діаграма діяльності

ДОДАТОК Б

UML діаграма взаємодії

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.ч

Рисунок 2 – UML діаграма взаємодії між клієнтом і сервером UML діаграма взаємодії між клієнтом і сервером