МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий Кафедра «Электроника и сети ЭВМ»

## Лабораторная работа №2

«Введение в JavaScript. Программное взаимодействие с HTML документами на основе DOM API»

## по дисциплине

«WEB - технологии»

РУКОВОДИТЕЛЬ:

Биткин Д.Е..

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

Скорынин С.С.

(подпись) (фамилия, и.,о.)

22-ПО

(шифр группы)

Работа защищена « »

С оценкой

Нижний Новгород 2024 г

**Задание на лабораторную работу**: Приложение должно быть написано согласно паттерну MVС.

Опционально: используйте луковую архитектуру.

Создать CRUD (Create Read Update Delete) web-сервис, соответствующий принципам REST.

Реализовать web-сервис соответствующий принципам REST (можно не реализовывать кэш, код по требованию, слои).

Подключить swagger к веб-сервису(обязательно описать реализованные API в swagger)

1. Реализовать следующие методы или типов запросов HTTP для взаимодействия с сервером:

* GET
* POST
* PUT
* DELETE

1. Для документирования API добавить в проект Swagger UI
2. Продемонстрировать работу сервиса через браузер (запросы, возвращаемы ответ).

Хранилище реализовать в памяти.

Индивидуальное задание:

Вариант 4:

Реализовать веб-сайт со следующими страницами:

* страница для добавления/удаления картинок;
* страница для постраничного просмотра и отметки картинок (нравится / не нравится);
* страница для отображения рейтинга картинок;
* контакты предприятия;
* Хранилище реализовать в памяти с ограничением в 50 записей.
* Должны быть реализованы кнопки навигации и взаимодействия с пользователем.
* Должно быть реализовано интерактивное взаимодействие с пользователем.

Код

Models/imaje.js

class Image {

    constructor(id, path) {

        this.id = id; // Уникальный идентификатор изображения.

        this.path = path; // Путь к изображению (например, '/images/sample.jpg').

        this.likes = 0; // Количество лайков.

        this.dislikes = 0; // Количество дизлайков.

    }

}

module.exports = Image; // Экспортируем класс, чтобы использовать его в других файлах.

Public/css/style.css

body {

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 0;

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    background-color: #f3f3f3;

}

header {

    position: absolute;

    top: 0;

    text-align: center;

    width: 100%;

    padding: 10px;

    background: #333;

    color: white;

}

.slider-container {

    display: flex;

    align-items: center;

    justify-content: center;

    position: relative;

    max-width: 600px;

    width: 100%;

}

.slider {

    text-align: center;

    flex: 1;

    position: relative;

}

.slider img {

    max-width: 100%;

    max-height: 400px;

    border-radius: 10px;

    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.3);

    margin-bottom: 10px;

}

/\* Информация под изображением \*/

.image-info {

    text-align: center;

    margin-top: 10px;

    font-size: 1rem;

    color: #555;

}

.image-info p {

    margin: 5px 0;

}

/\* Кнопки лайков \*/

.like-buttons {

    display: flex;

    justify-content: center;

    gap: 10px;

}

.like-button {

    background-color: #28a745;

    color: white;

    border: none;

    padding: 10px 20px;

    margin: 5px;

    font-size: 1rem;

    cursor: pointer;

    border-radius: 5px;

    transition: background-color 0.3s ease;

}

.like-button:hover {

    background-color: #218838;

}

/\* Фиксированные кнопки переключения \*/

.nav-button {

    background-color: #007BFF;

    color: white;

    border: none;

    padding: 10px 15px;

    font-size: 1.5rem;

    cursor: pointer;

    border-radius: 50%;

    margin: 0 10px;

    z-index: 1000;

}

.nav-button:hover {

    background-color: #0056b3;

}

.nav-button.fixed {

    position: fixed;

    top: 50%;

    /\* Центрируем кнопки по вертикали \*/

    transform: translateY(-50%);

    /\* Центрируем кнопки точно \*/

}

#prevButton {

    left: 20px;

    /\* Фиксируем кнопку "влево" слева \*/

}

#nextButton {

    right: 20px;

    /\* Фиксируем кнопку "вправо" справа \*/

}

header h1 {

    margin: 0;

}

nav ul {

    list-style: none;

    display: flex;

    justify-content: center;

    margin: 0;

    padding: 0;

}

nav ul li {

    margin: 0 10px;

}

nav ul li a {

    color: white;

    text-decoration: none;

    font-weight: bold;

}

nav ul li a:hover {

    text-decoration: underline;

}

main {

    padding: 20px;

    text-align: center;

}

button {

    background: #333;

    color: #fff;

    border: none;

    padding: 10px 15px;

    cursor: pointer;

}

button:hover {

    background: #555;

}

.gallery-container {

    display: flex;

    flex-wrap: wrap;

    justify-content: center;

    align-items: center;

    gap: 20px;

    min-height: 200px;

    text-align: center;

}

.gallery-item {

    border: 1px solid #ddd;

    padding: 10px;

    border-radius: 5px;

    background: #f9f9f9;

    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

    text-align: center;

}

.gallery-item img {

    max-width: 100%;

    height: auto;

    margin-bottom: 10px;

}

.empty-message {

    font-size: 1.5rem;

    color: #666;

    margin: 20px 0;

    text-align: center;

}

Public/js/gallery.js

document.addEventListener('DOMContentLoaded', async () => {

    const sliderImage = document.getElementById('sliderImage');

    const imageId = document.getElementById('imageId');

    const likes = document.getElementById('likes');

    const dislikes = document.getElementById('dislikes');

    const prevButton = document.getElementById('prevButton');

    const nextButton = document.getElementById('nextButton');

    const likeButton = document.getElementById('likeButton');

    const dislikeButton = document.getElementById('dislikeButton');

    let images = [];

    let currentIndex = 0;

    // Загружаем изображения с сервера

    try {

        const response = await fetch('/api/images');

        images = await response.json();

        if (images.length === 0) {

            sliderImage.src = '';

            imageId.innerText = 'Галерея пуста';

            likes.innerText = '0';

            dislikes.innerText = '0';

            return;

        }

        showImage(0);

    } catch (error) {

        console.error('Ошибка загрузки изображений:', error);

    }

    // Функция для отображения текущего изображения

    function showImage(index) {

        if (images.length === 0) return;

        const image = images[index];

        sliderImage.src = image.path;

        imageId.innerText = image.id;

        likes.innerText = image.likes;

        dislikes.innerText = image.dislikes;

    }

    // Перелистывание влево

    prevButton.addEventListener('click', () => {

        if (images.length === 0) return;

        currentIndex = (currentIndex - 1 + images.length) % images.length; // Кольцевой переход

        showImage(currentIndex);

    });

    // Перелистывание вправо

    nextButton.addEventListener('click', () => {

        if (images.length === 0) return;

        currentIndex = (currentIndex + 1) % images.length; // Кольцевой переход

        showImage(currentIndex);

    });

    // Добавление лайка

    likeButton.addEventListener('click', async () => {

        const image = images[currentIndex];

        await fetch(`/api/images/${image.id}/like`, { method: 'POST' });

        image.likes += 1;

        showImage(currentIndex);

    });

    // Добавление дизлайка

    dislikeButton.addEventListener('click', async () => {

        const image = images[currentIndex];

        await fetch(`/api/images/${image.id}/dislike`, { method: 'POST' });

        image.dislikes += 1;

        showImage(currentIndex);

    });

});

Public/js/sckipts.js

// Обработчик добавления изображения

document.getElementById('addForm')?.addEventListener('submit', async (e) => {

    e.preventDefault(); // Отключаем стандартное поведение формы

    const path = document.getElementById('path').value; // Получаем значение пути к изображению

    if (!path) {

        alert('Пожалуйста, введите путь к изображению.');

        return;

    }

    try {

        // Отправляем запрос на сервер для добавления изображения

        const response = await fetch('/api/images', {

            method: 'POST',

            headers: { 'Content-Type': 'application/json' },

            body: JSON.stringify({ path }),

        });

        const result = await response.json(); // Парсим ответ от сервера

        if (response.ok) {

            alert(`Изображение добавлено с ID: ${result.id}`);

            document.getElementById('path').value = ''; // Очищаем поле ввода

        } else {

            alert(`Ошибка: ${result.error}`); // Показываем ошибку, если она есть

        }

    } catch (error) {

        console.error('Ошибка при добавлении изображения:', error);

        alert('Произошла ошибка. Пожалуйста, попробуйте снова.');

    }

});

// Обработчик удаления изображения

document.getElementById('deleteForm')?.addEventListener('submit', async (e) => {

    e.preventDefault(); // Отключаем стандартное поведение формы

    const id = document.getElementById('id').value; // Получаем ID изображения

    if (!id) {

        alert('Пожалуйста, введите ID изображения.');

        return;

    }

    try {

        // Отправляем запрос на сервер для удаления изображения

        const response = await fetch(`/api/images/${id}`, {

            method: 'DELETE',

        });

        const result = await response.json(); // Парсим ответ от сервера

        if (response.ok) {

            alert(result.message); // Показываем сообщение об успешном удалении

            document.getElementById('id').value = ''; // Очищаем поле ввода

        } else {

            alert(`Ошибка: ${result.error}`); // Показываем ошибку, если она есть

        }

    } catch (error) {

        console.error('Ошибка при удалении изображения:', error);

        alert('Произошла ошибка. Пожалуйста, попробуйте снова.');

    }

});

async function likeImage(id) {

    try {

        const response = await fetch(`/api/images/${id}/like`, { method: 'POST' });

        if (!response.ok) {

            const result = await response.json();

            alert(`Ошибка: ${result.error}`);

            return;

        }

        loadGallery(); // Обновляем галерею после успешного лайка

    } catch (error) {

        console.error('Ошибка при добавлении лайка:', error);

        alert('Не удалось поставить лайк. Попробуйте снова.');

    }

}

async function dislikeImage(id) {

    try {

        const response = await fetch(`/api/images/${id}/dislike`, { method: 'POST' });

        if (!response.ok) {

            const result = await response.json();

            alert(`Ошибка: ${result.error}`);

            return;

        }

        loadGallery(); // Обновляем галерею после успешного дизлайка

    } catch (error) {

        console.error('Ошибка при добавлении дизлайка:', error);

        alert('Не удалось поставить дизлайк. Попробуйте снова.');

    }

}

Public/add\_remove.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Добавить/Удалить картинки</title>

    <link rel="stylesheet" href="css/styles.css">

</head>

<body>

    <header>

        <h1>Добавить/Удалить картинки</h1>

        <nav>

            <ul>

                <li><a href="index.html">Главная</a></li>

                <li><a href="gallery.html">Галерея</a></li>

                <li><a href="rating.html">Рейтинг</a></li>

                <li><a href="contacts.html">Контакты</a></li>

            </ul>

        </nav>

    </header>

    <main>

        <h2>Добавить изображение</h2>

        <form id="addForm">

            <input type="text" id="path" placeholder="Введите путь к изображению">

            <button type="submit">Добавить</button>

        </form>

        <h2>Удалить изображение</h2>

        <form id="deleteForm">

            <input type="number" id="id" placeholder="Введите ID изображения">

            <button type="submit">Удалить</button>

        </form>

    </main>

    <script src="js/scripts.js"></script>

</body>

</html>

Public/contacts.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Контакты</title>

    <link rel="stylesheet" href="css/styles.css">

</head>

<body>

    <header>

        <h1>Контакты</h1>

        <nav>

            <ul>

                <li><a href="index.html">Главная</a></li>

                <li><a href="add\_remove.html">Добавить/Удалить картинки</a></li>

                <li><a href="gallery.html">Галерея</a></li>

                <li><a href="rating.html">Рейтинг</a></li>

            </ul>

        </nav>

    </header>

    <main>

        <h2>Свяжитесь с нами</h2>

        <p>Email: example@example.com</p>

        <p>Телефон: +7 (123) 456-7890</p>

    </main>

</body>

</html>

Public/gallery.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Галерея</title>

    <link rel="stylesheet" href="css/styles.css">

</head>

<body>

    <header>

        <h1>Галерея</h1>

        <nav>

            <ul>

                <li><a href="index.html">Главная</a></li>

                <li><a href="add\_remove.html">Добавить/Удалить картинки</a></li>

                <li><a href="rating.html">Рейтинг</a></li>

                <li><a href="contacts.html">Контакты</a></li>

            </ul>

        </nav>

    </header>

    <main>

        <div class="slider-container">

            <!-- Кнопка для перехода к предыдущему изображению -->

            <button class="nav-button" id="prevButton">&lt;</button>

            <!-- Слайдер -->

            <div class="slider">

                <img id="sliderImage">

                <div class="image-info">

                    <p>ID: <span id="imageId">1</span></p>

                    <p>Лайки: <span id="likes">0</span></p>

                    <p>Дизлайки: <span id="dislikes">0</span></p>

                </div>

                <div class="like-buttons">

                    <button class="like-button" id="likeButton">👍 Лайк</button>

                    <button class="like-button" id="dislikeButton">👎 Дизлайк</button>

                </div>

            </div>

            <!-- Кнопка для перехода к следующему изображению -->

            <button class="nav-button" id="nextButton">&gt;</button>

        </div>

    </main>

    <!-- Подключение скрипта для управления галереей -->

    <script src="js/gallery.js"></script>

</body>

</html>

Public/index.html

<!-- <!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Image Service</title>

    <link rel="stylesheet" href="css/styles.css">

</head>

<body>

    <h1>Добро пожаловать в Image Service</h1>

    <header>

        <nav>

            <ul>

                <li><a href="#" onclick="loadPage('home')">Главная</a></li>

                <li><a href="#" onclick="loadPage('add\_remove')">Добавить/Удалить</a></li>

                <li><a href="#" onclick="loadPage('gallery')">Галерея</a></li>

                <li><a href="#" onclick="loadPage('contacts')">Контакты</a></li>

            </ul>

        </nav>

    </header>

    <main id="content">

    </main>

    <script src="js/scripts.js"></script>

</body>

</html> -->

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Image Service</title>

    <link rel="stylesheet" href="css/styles.css">

</head>

<body>

    <header>

        <h1>Image Service</h1>

        <nav>

            <ul>

                <li><a href="index.html">Главная</a></li>

                <li><a href="add\_remove.html">Добавить/Удалить картинки</a></li>

                <li><a href="gallery.html">Галерея</a></li>

                <li><a href="rating.html">Рейтинг</a></li>

                <li><a href="contacts.html">Контакты</a></li>

            </ul>

        </nav>

    </header>

    <main>

        <h2>Добро пожаловать в Image Service!</h2>

        <p>Используйте меню, чтобы перейти к нужному разделу.</p>

    </main>

</body>

</html>

Public/rating.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Рейтинг</title>

    <link rel="stylesheet" href="css/styles.css">

</head>

<body>

    <header>

        <h1>Рейтинг изображений</h1>

        <nav>

            <ul>

                <li><a href="index.html">Главная</a></li>

                <li><a href="add\_remove.html">Добавить/Удалить картинки</a></li>

                <li><a href="gallery.html">Галерея</a></li>

                <li><a href="contacts.html">Контакты</a></li>

            </ul>

        </nav>

    </header>

    <main>

        <h2>Топ изображений</h2>

        <ul id="ratingList"></ul>

    </main>

    <script>

        window.onload = async () => {

            const list = document.getElementById('ratingList');

            const response = await fetch('/api/images');

            const images = await response.json();

            const sortedImages = images.sort((a, b) => b.likes - a.likes);

            sortedImages.forEach((image) => {

                const li = document.createElement('li');

                li.innerHTML = `ID: ${image.id}, Лайки: ${image.likes}`;

                list.appendChild(li);

            });

        };

    </script>

</body>

</html>

Routes/images.js

const express = require('express'); // Подключаем Express.

const router = express.Router(); // Создаём новый роутер.

const imageService = require('../services/imageService'); // Подключаем сервис изображений.

// Получить все изображения.

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

    res.json(imageService.getAllImages()); // Возвращаем JSON с изображениями.

});

// Добавить изображение.

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

    const { path } = req.body; // Получаем путь к изображению из тела запроса.

    try {

        const newImage = imageService.createImage(path); // Создаём изображение.

        res.status(201).json(newImage); // Возвращаем созданное изображение.

    } catch (error) {

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

    }

});

// Поставить лайк/дизлайк.

router.post('/:id/:action', (req, res) => {

    const { id, action } = req.params; // Получаем ID и действие (like/dislike).

    const isLike = action === 'like'; // Определяем, это лайк или дизлайк.

    try {

        imageService.likeImage(parseInt(id), isLike); // Обновляем данные.

        res.status(200).json({ message: `${action} added` }); // Возвращаем успех.

    } catch (error) {

        res.status(404).json({ error: error.message }); // Обрабатываем ошибку.

    }

});

// Удалить изображение.

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

    const { id } = req.params; // Получаем ID изображения.

    try {

        imageService.removeImage(parseInt(id)); // Удаляем изображение.

        res.status(200).json({ message: 'Image deleted' }); // Возвращаем успех.

    } catch (error) {

        res.status(404).json({ error: error.message }); // Обрабатываем ошибку.

    }

});

module.exports = router; // Экспортируем маршруты.

Service/imageservice.js

const storage = require('../storage/memoryStorage'); // Подключаем хранилище.

const Image = require('../models/image'); // Подключаем модель Image.

// Создать изображение.

const createImage = (path) => {

    const id = storage.getImages().length + 1; // Генерируем новый ID.

    const newImage = new Image(id, path); // Создаём новый объект изображения.

    if (!storage.addImage(newImage)) { // Пытаемся добавить изображение в хранилище.

        throw new Error('Storage limit reached'); // Если не удалось, выбрасываем ошибку.

    }

    return newImage; // Возвращаем созданное изображение.

};

// Получить все изображения.

const getAllImages = () => storage.getImages();

// Поставить лайк/дизлайк изображению.

const likeImage = (id, isLike) => {

    if (!storage.updateLikes(id, isLike)) { // Если обновить лайки не удалось.

        throw new Error('Image not found'); // Выбрасываем ошибку.

    }

};

// Удалить изображение.

const removeImage = (id) => {

    if (!storage.deleteImage(id)) { // Если удалить изображение не удалось.

        throw new Error('Image not found'); // Выбрасываем ошибку.

    }

};

module.exports = {

    createImage,

    getAllImages,

    likeImage,

    removeImage,

};

Storage/memoryStorege.js

const images = []; // Список для хранения всех изображений в памяти.

const MAX\_IMAGES = 50; // Максимальное количество изображений в хранилище.

// Добавить изображение.

const addImage = (image) => {

    if (images.length < MAX\_IMAGES) { // Если хранилище не заполнено.

        images.push(image); // Добавляем изображение в список.

        return true; // Возвращаем успех.

    }

    return false; // Если хранилище заполнено, возвращаем ошибку.

};

// Получить все изображения.

const getImages = () => images;

// Найти изображение по ID.

const getImageById = (id) => images.find((img) => img.id === id);

// Удалить изображение по ID.

const deleteImage = (id) => {

    const index = images.findIndex((img) => img.id === id); // Ищем индекс изображения.

    if (index !== -1) { // Если изображение найдено.

        images.splice(index, 1); // Удаляем его из списка.

        return true;

    }

    return false; // Если изображение не найдено, возвращаем ошибку.

};

// Обновить количество лайков/дизлайков.

const updateLikes = (id, isLike) => {

    const image = getImageById(id); // Находим изображение по ID.

    if (image) {

        isLike ? image.likes++ : image.dislikes++; // Увеличиваем лайки или дизлайки.

        return true;

    }

    return false; // Если изображение не найдено, возвращаем ошибку.

};

module.exports = {

    addImage,

    getImages,

    getImageById,

    deleteImage,

    updateLikes,

};

Server.js

const express = require('express'); // Подключаем фреймворк Express для создания сервера.

const bodyParser = require('body-parser'); // Подключаем middleware для обработки JSON-запросов.

const swaggerUi = require('swagger-ui-express'); // Подключаем Swagger UI для документирования API.

const swaggerDocument = require('./swagger/swagger.json'); // Загружаем Swagger-документацию.

const imageRoutes = require('./routes/images'); // Подключаем маршруты для работы с изображениями.

const app = express(); // Создаём экземпляр Express-приложения.

app.use(bodyParser.json()); // Добавляем middleware для обработки входящих JSON-запросов.

// Подключаем Swagger UI по маршруту /api-docs.

app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerDocument));

// Подключаем маршруты для работы с API изображений.

app.use('/api/images', imageRoutes);

// Указываем папку public для раздачи статических файлов (HTML, CSS, JS).

app.use(express.static('public'));

// Настраиваем сервер для прослушивания на порту 3000.

const PORT = 3000;

app.listen(PORT, () => {

    console.log(`Server running at http://localhost:${PORT}`); // Выводим сообщение, что сервер запущен.

    console.log(`Swagger available at http://localhost:${PORT}/api-docs`); // Выводим ссылку на Swagger UI.

});

Swagger/swagger.json

{

    "swagger": "2.0",

    "info": {

        "version": "1.0.0",

        "title": "Image Management API",

        "description": "API для управления изображениями в галерее. Поддерживает добавление, удаление, лайки и дизлайки изображений."

    },

    "host": "localhost:3000",

    "basePath": "/api",

    "paths": {

        "/images": {

            "get": {

                "summary": "Получить список всех изображений",

                "description": "Возвращает список всех изображений с информацией о них, включая лайки и дизлайки.",

                "responses": {

                    "200": {

                        "description": "Список изображений успешно получен",

                        "schema": {

                            "type": "array",

                            "items": {

                                "$ref": "#/definitions/Image"

                            }

                        }

                    },

                    "500": {

                        "description": "Ошибка сервера при получении списка изображений"

                    }

                }

            },

            "post": {

                "summary": "Добавить новое изображение",

                "description": "Добавляет новое изображение в галерею. Требуется указать путь к изображению.",

                "parameters": [

                    {

                        "in": "body",

                        "name": "body",

                        "description": "Путь к изображению",

                        "required": true,

                        "schema": {

                            "type": "object",

                            "properties": {

                                "path": {

                                    "type": "string",

                                    "example": "/images/photo.jpg"

                                }

                            },

                            "required": [

                                "path"

                            ]

                        }

                    }

                ],

                "responses": {

                    "201": {

                        "description": "Изображение успешно добавлено",

                        "schema": {

                            "$ref": "#/definitions/Image"

                        }

                    },

                    "400": {

                        "description": "Ошибка в запросе (например, путь не указан)"

                    }

                }

            }

        },

        "/images/{id}/{action}": {

            "post": {

                "summary": "Поставить лайк или дизлайк изображению",

                "description": "Позволяет поставить лайк или дизлайк изображению. Укажите ID изображения и действие (`like` или `dislike`).",

                "parameters": [

                    {

                        "name": "id",

                        "in": "path",

                        "required": true,

                        "type": "integer",

                        "description": "ID изображения"

                    },

                    {

                        "name": "action",

                        "in": "path",

                        "required": true,

                        "type": "string",

                        "enum": [

                            "like",

                            "dislike"

                        ],

                        "description": "Действие: 'like' для лайка или 'dislike' для дизлайка"

                    }

                ],

                "responses": {

                    "200": {

                        "description": "Лайк или дизлайк успешно добавлен",

                        "schema": {

                            "type": "object",

                            "properties": {

                                "message": {

                                    "type": "string",

                                    "example": "like added"

                                }

                            }

                        }

                    },

                    "400": {

                        "description": "Некорректное действие (например, не 'like' или 'dislike')"

                    },

                    "404": {

                        "description": "Изображение не найдено"

                    }

                }

            }

        },

        "/images/{id}": {

            "delete": {

                "summary": "Удалить изображение по ID",

                "description": "Удаляет изображение с указанным ID.",

                "parameters": [

                    {

                        "name": "id",

                        "in": "path",

                        "required": true,

                        "type": "integer",

                        "description": "ID изображения"

                    }

                ],

                "responses": {

                    "200": {

                        "description": "Изображение успешно удалено",

                        "schema": {

                            "type": "object",

                            "properties": {

                                "message": {

                                    "type": "string",

                                    "example": "Image deleted"

                                }

                            }

                        }

                    },

                    "404": {

                        "description": "Изображение с указанным ID не найдено"

                    },

                    "500": {

                        "description": "Ошибка сервера при удалении изображения"

                    }

                }

            }

        }

    },

    "definitions": {

        "Image": {

            "type": "object",

            "properties": {

                "id": {

                    "type": "integer",

                    "example": 1

                },

                "path": {

                    "type": "string",

                    "example": "/images/photo.jpg"

                },

                "likes": {

                    "type": "integer",

                    "example": 10

                },

                "dislikes": {

                    "type": "integer",

                    "example": 2

                }

            }

        }

    }

}

Запуск кода:

 

   

   