МИНИСТЕРСТВО ТРАНСПОРТА РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)»

Кафедра ЦТУТП

**Отчёт**

По лабораторной работе №8  
по дисциплине «Корпоративные информационные системы»

Тема: «Написание приложение Vue»

Вариант №4

Выполнил: Иванов Н. С.

Группа: УИС-411

Преподаватель: доц. Кафедры ЦТУТП

Козьяков П. О.

­

Москва 2024 г.

**СОДЕРЖАНИЕ**

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc180325218)

[ХОД ВЫПОЛНЕНИЯ РАБОТЫ 4](#_Toc180325219)

[РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 5](#_Toc180325220)

[КОД ПРОГРАММЫ 6](#_Toc180325221)

[ВЫВОД 9](#_Toc180325222)

# **ПОСТАНОВКА ЗАДАЧИ**

Для контроллера, реализованного в лабораторной работе №7, создать приложение на Vue, выполняющее запросы к контроллеру и обрабатывающее ответы.

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

В рамках лабораторной работы было разработано клиентское приложение для управления парками с использованием Vue.js. Основные функции включают добавление, обновление, удаление и просмотр информации о парках. Приложение взаимодействует с REST API, работающим на сервере по адресу http://localhost:8081.

Первым этапом было создание двух компонентов: ParkForm для добавления и обновления данных и ParkList для отображения списка парков и их удаления. Компонент ParkForm предоставляет форму, которая позволяет пользователю вводить название парка, его местоположение, вместимость, а также статус (открыт/закрыт). Логика обработки данных реализована через сервис parkService, обеспечивающий отправку соответствующих запросов на сервер.

Компонент ParkList отвечает за отображение списка парков, предоставленных сервером. Он реализует метод fetchParks для получения данных и их отображения в виде списка. Пользователь может удалить любой из парков, воспользовавшись соответствующей кнопкой, после чего список автоматически обновляется.

Для взаимодействия с серверной частью создан сервис parkService, который реализует методы для получения, добавления, обновления и удаления данных о парках. Каждый метод сервиса отправляет HTTP-запрос с использованием библиотеки Axios, а базовый URL установлен на уровне конфигурации.

Основной компонент приложения, App, объединяет оба компонента (ParkForm и ParkList) в единую структуру. В корневом компоненте задается базовый интерфейс для управления парками, обеспечивающий интуитивно понятный пользовательский опыт.

В процессе разработки приложения были проведены тесты, чтобы убедиться в корректной работе форм, обработке данных, а также взаимодействии с серверной частью.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**



Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

<template>  
 <div>  
 <h2>Добавить/Обновить Парк</h2>  
 <form @submit.prevent="submitPark">  
 <input v-model="park.name" placeholder="Название парка" required />  
 <input v-model="park.location" placeholder="Локация" required />  
 <input type="number" v-model="park.capacity" placeholder="Вместимость" required />  
 <label>  
 Открыт:  
 <input type="checkbox" v-model="park.is\_open" />  
 </label>  
 <button type="submit">Сохранить</button>  
 </form>  
 </div>  
</template>  
  
<script>  
import parkService from '@/services/parkService';  
  
export default {  
 data() {  
 return {  
 park: {  
 id: null,  
 name: '',  
 location: '',  
 capacity: null,  
 is\_open: false,  
 },  
 };  
 },  
 methods: {  
 async submitPark() {  
 if (this.park.id) {  
 await parkService.updatePark(this.park);  
 } else {  
 await parkService.addPark(this.park);  
 }  
 this.clearForm();  
 },  
 clearForm() {  
 this.park = { id: null, name: '', location: '', capacity: null, is\_open: false };  
 },  
 },  
};  
</script>

<template>  
 <div>  
 <h2>Список парков</h2>  
 <ul>  
 <li v-for="park in parks" :key="park.id">  
 {{ park.name }} ({{ park.location }}) - Вместимость: {{ park.capacity }}  
 <span v-if="park.is\_open">[Открыт]</span>  
 <button @click="deletePark(park.id)">Удалить</button>  
 </li>  
 </ul>  
 </div>  
</template>  
  
<script>  
import parkService from '@/services/parkService';  
  
export default {  
 data() {  
 return {  
 parks: [],  
 };  
 },  
 methods: {  
 async fetchParks() {  
 const response = await parkService.getParks();  
 this.parks = response.data;  
 },  
 async deletePark(id) {  
 await parkService.deletePark(id);  
 this.fetchParks();  
 },  
 },  
 created() {  
 this.fetchParks();  
 },  
};  
</script>

import axios from 'axios';  
  
const API\_URL = 'http://localhost:8081/parks';  
  
export default {  
 getParks() {  
 return axios.get(API\_URL);  
 },  
 addPark(park) {  
 return axios.post(API\_URL, park);  
 },  
 updatePark(park) {  
 return axios.put(API\_URL, park);  
 },  
 deletePark(id) {  
 return axios.delete(`${API\_URL}/${id}`);  
 },  
};

<template>  
 <div id="app">  
 <h1>Управление парками</h1>  
 <ParkForm />  
 <ParkList />  
 </div>  
</template>  
  
<script>  
import ParkList from '@/components/ParkList.vue';  
import ParkForm from '@/components/ParkForm.vue';  
  
export default {  
 components: {  
 ParkList,  
 ParkForm,  
 },  
};  
</script>

import { createApp } from 'vue';  
import App from './App.vue';  
import axios from 'axios';  
  
axios.defaults.baseURL = 'http://localhost:8081';  
createApp(App).mount('#app');

# **ВЫВОД**

В результате выполнения лабораторной работы было создано функциональное клиентское приложение для управления парками. Пользователь может добавлять новые парки, редактировать их данные, удалять ненужные записи и просматривать список доступных парков. Работа продемонстрировала способность эффективно организовать клиент-серверное взаимодействие с использованием Vue.js и Axios. Полученные знания и навыки будут полезны для разработки более сложных приложений, использующих REST API.