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 №15

Выполнил: Дронов А. А.

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**

Лабораторная работа начинается с разработки интерфейса для управления информацией об университетах. Был создан компонент формы для добавления и редактирования данных о университетах. Форма включает поля для ввода названия, расположения, количества студентов и статуса (государственный или нет). При отправке формы данные отправляются на сервер через REST API. Компонент формы реализует возможность как добавления нового университета, так и редактирования существующего.

Для отображения списка университетов создан отдельный компонент. Он загружает данные с сервера и отображает их в виде списка. Каждая запись включает название, местоположение, количество студентов и статус университета. Пользователю предоставлены кнопки для редактирования и удаления записей. При редактировании данные выбранного университета загружаются в форму для дальнейшего изменения.

На стороне сервера был разработан API, используя Spring Boot. Контроллер предоставляет методы для добавления, обновления, получения и удаления данных. REST API обеспечивает обмен данными между фронтендом и сервером. Для настройки CORS использован конфигурационный класс, позволяющий фронтенду на другом порту взаимодействовать с сервером.

Взаимодействие между фронтендом и сервером реализовано через библиотеку axios. Были протестированы все CRUD-операции (создание, чтение, обновление, удаление) с использованием реальных данных. После отправки изменений форма очищается, а список университетов обновляется автоматически, что обеспечивает удобство работы.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

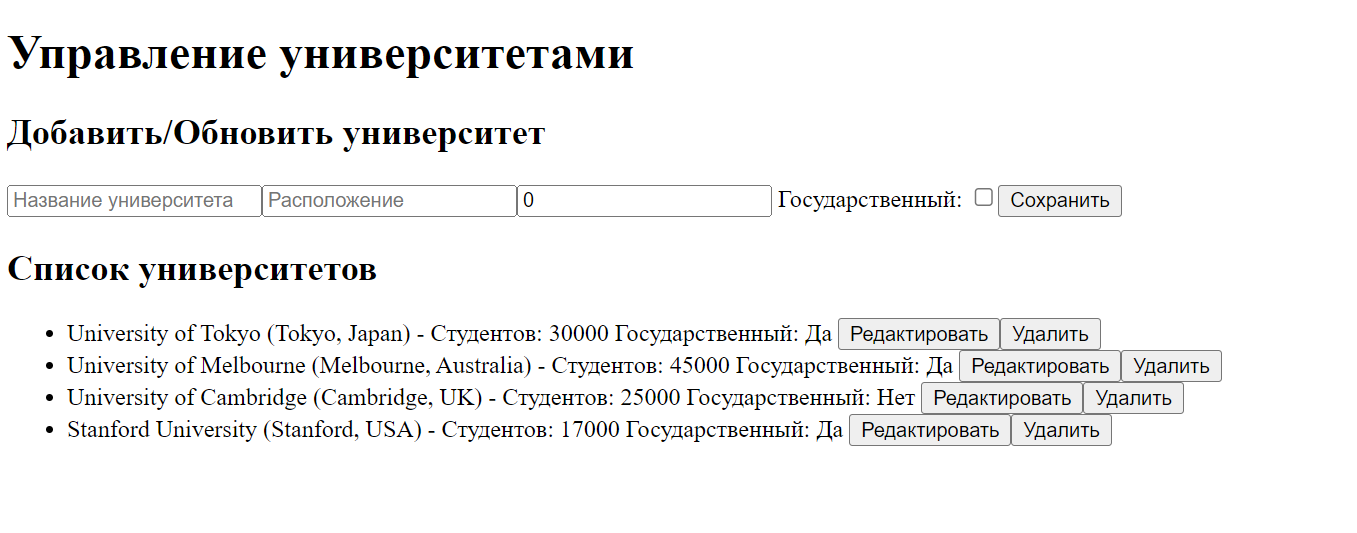


Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

<template>  
 <div>  
 <h2>Добавить/Обновить университет</h2>  
 <form @submit.prevent="submitUniversity">  
 <input v-model="university.name" placeholder="Название университета" required />  
 <input v-model="university.location" placeholder="Расположение" required />  
 <input  
 v-model.number="university.student\_count"  
 type="number"  
 placeholder="Количество студентов"  
 required  
 />  
 <label>  
 Государственный:  
 <input  
 type="checkbox"  
 v-model="university.is\_public"  
 @change="updateIsPublic"  
 />  
 </label>  
 <button type="submit">Сохранить</button>  
 </form>  
 </div>  
</template>  
  
<script>  
import universityService from "@/services/universityService";  
  
export default {  
 data() {  
 return {  
 university: {  
 id: null,  
 name: "",  
 location: "",  
 student\_count: 0,  
 is\_public: false,  
 },  
 };  
 },  
 methods: {  
 updateIsPublic(event) {  
 this.university.is\_public = event.target.checked || false;  
 },  
 async submitUniversity() {  
 this.university.is\_public = !!this.university.is\_public;  
 if (this.university.id) {  
 await universityService.updateUniversity(this.university);  
 } else {  
 await universityService.addUniversity(this.university);  
 }  
 this.clearForm();  
 this.$emit("refresh");  
 },  
 clearForm() {  
 this.university = {  
 id: null,  
 name: "",  
 location: "",  
 student\_count: 0,  
 is\_public: false,  
 };  
 },  
 editUniversity(university) {  
 this.university = { ...university };  
 },  
 },  
};  
</script>

<template>  
 <div>  
 <h2>Список университетов</h2>  
 <ul>  
 <li v-for="university in universities" :key="university.id">  
 {{ university.name }} ({{ university.location }}) - Студентов: {{ university.student\_count }}  
 Государственный: {{ university.is\_public ? "Да" : "Нет" }}  
 <button @click="$emit('edit', university)">Редактировать</button>  
 <button @click="deleteUniversity(university.id)">Удалить</button>  
 </li>  
 </ul>  
 </div>  
</template>  
  
<script>  
import universityService from "@/services/universityService";  
  
export default {  
 data() {  
 return {  
 universities: [],  
 };  
 },  
 methods: {  
 async fetchUniversities() {  
 this.universities = (await universityService.getUniversities()).data;  
 },  
 async deleteUniversity(id) {  
 await universityService.deleteUniversity(id);  
 this.fetchUniversities();  
 },  
 },  
 mounted() {  
 this.fetchUniversities();  
 },  
};  
</script>

import axios from "axios";  
  
const API\_URL = "http://localhost:8081/universities";  
  
export default {  
 getUniversities() {  
 return axios.get(API\_URL);  
 },  
 addUniversity(university) {  
 return axios.post(API\_URL, university);  
 },  
 updateUniversity(university) {  
 return axios.put(API\_URL, university);  
 },  
 deleteUniversity(id) {  
 return axios.delete(`${API\_URL}/${id}`);  
 },  
};

<template>  
 <div id="app">  
 <h1>Управление университетами</h1>  
 <UniversityForm @refresh="refreshList" ref="form" />  
 <UniversityList @edit="editUniversity" ref="list" />  
 </div>  
</template>  
  
<script>  
import UniversityForm from "@/components/UniversityForm.vue";  
import UniversityList from "@/components/UniversityList.vue";  
  
export default {  
 components: {  
 UniversityForm,  
 UniversityList,  
 },  
 methods: {  
 refreshList() {  
 this.$refs.list.fetchUniversities();  
 },  
 editUniversity(university) {  
 this.$refs.form.editUniversity(university);  
 },  
 },  
};  
</script>

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

# **ВЫВОД**

Лабораторная работа продемонстрировала принципы создания полнофункционального веб-приложения. В ходе работы были использованы современные технологии, такие как Vue.js для фронтенда и Spring Boot для бэкенда. Полученное приложение эффективно решает задачу управления данными об университетах и является хорошим примером применения паттернов проектирования клиент-серверной архитектуры.