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 №12

Выполнил: Бугаева А. Л.

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 6](#_Toc180325220)

[КОД ПРОГРАММЫ 7](#_Toc180325221)

[ВЫВОД 11](#_Toc180325222)

# **ПОСТАНОВКА ЗАДАЧИ**

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

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

Ход лабораторной работы начинается с проектирования веб-приложения, которое позволяет управлять данными о компьютерах. Для этого были созданы два Vue.js-компонента: форма для добавления и редактирования данных о компьютерах и список для отображения всех записей. В форме предусмотрены поля для ввода информации о бренде, модели, процессоре, оперативной памяти, объёме диска и доступности. Данные из формы отправляются на сервер через REST API. После успешной отправки форма очищается, а список компьютеров обновляется.

Список компьютеров отображает данные, полученные с сервера, в структурированном виде. Пользователь может редактировать или удалять записи. При редактировании выбранный объект загружается в форму, где пользователь может внести изменения. При удалении запись мгновенно удаляется из списка благодаря вызову метода обновления данных.

На стороне сервера был разработан API на базе Spring Boot. Контроллер ComputerController обрабатывает запросы на добавление, обновление, получение и удаление данных. Методы контроллера тесно связаны с бизнес-логикой, реализованной в сервисе ComputerService. API спроектирован так, чтобы взаимодействие с клиентом было безопасным и эффективным.

Особое внимание уделено настройке CORS в конфигурации Spring. Используется класс WebConfig, который позволяет фронтенду взаимодействовать с сервером, запущенным на другом порту. Это обеспечивает полную совместимость между клиентом и сервером.

В процессе интеграции был настроен компонент axios для взаимодействия с сервером. Базовый URL был установлен для корректной работы с REST API. После этого функционал был протестирован с использованием реальных данных. Все операции — добавление, обновление и удаление — отработали корректно.

Результатом работы стало приложение, позволяющее эффективно управлять данными о компьютерах. Оно включает полный цикл операций CRUD (Create, Read, Update, Delete) и предоставляет пользователю удобный интерфейс для управления данными. Интеграция фронтенда и бэкенда была выполнена успешно, а тестирование подтвердило стабильную работу приложения.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

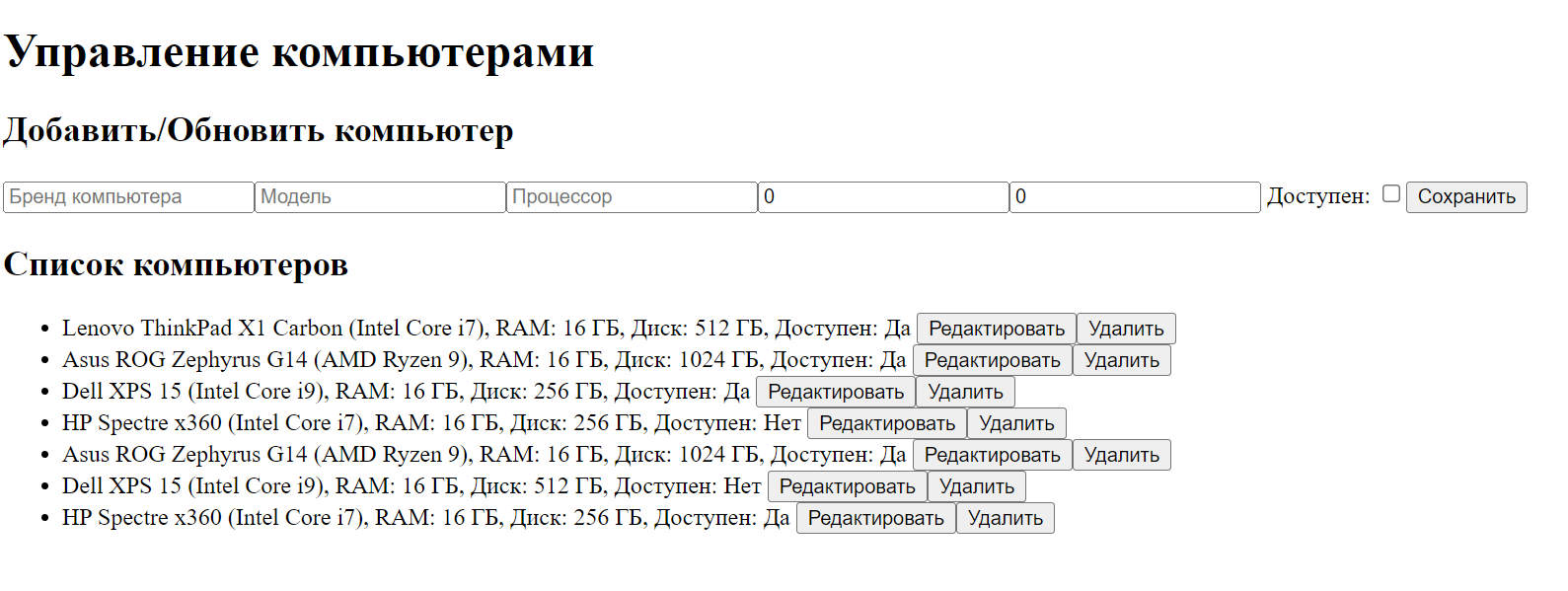


Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

<template>  
 <div>  
 <h2>Добавить/Обновить компьютер</h2>  
 <form @submit.prevent="submitComputer">  
 <input  
 v-model="computer.brand"  
 placeholder="Бренд компьютера"  
 required  
 />  
 <input  
 v-model="computer.model"  
 placeholder="Модель"  
 required  
 />  
 <input  
 v-model="computer.processor"  
 placeholder="Процессор"  
 required  
 />  
 <input  
 v-model.number="computer.ram"  
 type="number"  
 placeholder="RAM (ГБ)"  
 required  
 />  
 <input  
 v-model.number="computer.storage"  
 type="number"  
 placeholder="Объём диска (ГБ)"  
 required  
 />  
 <label>  
 Доступен:  
 <input  
 type="checkbox"  
 v-model="computer.is\_available"  
 @change="updateIsAvailable"  
 />  
 </label>  
 <button type="submit">Сохранить</button>  
 </form>  
 </div>  
</template>  
  
<script>  
import computerService from '@/services/computerService';  
  
export default {  
 data() {  
 return {  
 computer: {  
 id: null,  
 brand: '',  
 model: '',  
 processor: '',  
 ram: 0,  
 storage: 0,  
 is\_available: false,  
 },  
 };  
 },  
 methods: {  
 updateIsAvailable(event) {  
 this.computer.is\_available = event.target.checked || false;  
 },  
 async submitComputer() {  
 // Переводим is\_available в Boolean, чтобы убедиться в правильном типе.  
 this.computer.is\_available = !!this.computer.is\_available;  
 if (this.computer.id) {  
 await computerService.updateComputer(this.computer);  
 } else {  
 await computerService.addComputer(this.computer);  
 }  
 this.clearForm();  
 this.$emit('refresh');  
 },  
 clearForm() {  
 this.computer = {  
 id: null,  
 brand: '',  
 model: '',  
 processor: '',  
 ram: 0,  
 storage: 0,  
 is\_available: false,  
 };  
 },  
 editComputer(computer) {  
 this.computer = { ...computer };  
 },  
 },  
};  
</script>

<template>  
 <div>  
 <h2>Список компьютеров</h2>  
 <ul>  
 <li v-for="computer in computers" :key="computer.id">  
 {{ computer.brand }} {{ computer.model }} ({{ computer.processor }}),  
 RAM: {{ computer.ram }} ГБ, Диск: {{ computer.storage }} ГБ,  
 Доступен: {{ computer.is\_available ? 'Да' : 'Нет' }}  
 <button @click="$emit('edit', computer)">Редактировать</button>  
 <button @click="deleteComputer(computer.id)">Удалить</button>  
 </li>  
 </ul>  
 </div>  
</template>  
  
<script>  
import computerService from '@/services/computerService';  
  
export default {  
 data() {  
 return {  
 computers: [],  
 };  
 },  
 methods: {  
 async fetchComputers() {  
 // Предполагаем, что у нас есть .data в ответе  
 this.computers = (await computerService.getComputers()).data;  
 },  
 async deleteComputer(id) {  
 await computerService.deleteComputer(id);  
 this.fetchComputers();  
 },  
 },  
 mounted() {  
 this.fetchComputers();  
 },  
};  
</script>

import axios from 'axios';  
  
const API\_URL = 'http://localhost:8081/computers';  
  
export default {  
 getComputers() {  
 return axios.get(API\_URL);  
 },  
 addComputer(computer) {  
 return axios.post(API\_URL, computer);  
 },  
 updateComputer(computer) {  
 return axios.put(API\_URL, computer);  
 },  
 deleteComputer(id) {  
 return axios.delete(`${API\_URL}/${id}`);  
 },  
};

<template>  
 <div id="app">  
 <h1>Управление компьютерами</h1>  
 <ComputerForm @refresh="refreshList" ref="form" />  
 <ComputerList @edit="editComputer" ref="list" />  
 </div>  
</template>  
  
<script>  
import ComputerForm from '@/components/ComputerForm.vue';  
import ComputerList from '@/components/ComputerList.vue';  
  
export default {  
 name: 'App',  
 components: {  
 ComputerForm,  
 ComputerList,  
 },  
 methods: {  
 refreshList() {  
 this.$refs.list.fetchComputers();  
 },  
 editComputer(computer) {  
 this.$refs.form.editComputer(computer);  
 },  
 },  
};  
</script>

import { createApp } from 'vue';  
import App from './App.vue';  
import axios from 'axios';  
  
axios.defaults.baseURL = 'http://localhost:8081'; // Убедитесь, что адрес соответствует вашему бэкенду  
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 о компьютерах и является хорошим примером применения паттернов проектирования клиент-серверной архитектуры.