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 №7

Выполнил: Шедания В. М.

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**

Создан основной компонент App.vue, который объединяет два компонента: TrainForm и TrainList. TrainForm отвечает за добавление и обновление поездов, а TrainList — за отображение списка поездов и их удаление. Это позволяет разделить функционал для удобства работы и масштабирования.

В компоненте TrainForm реализована форма с полями для ввода названия, номера и статуса поезда. Метод submitTrain отправляет данные на сервер, проверяя наличие id у объекта. Если id существует, происходит обновление поезда через метод updateTrain сервиса trainService, если нет — добавление через метод addTrain. После успешного выполнения данных операций форма очищается вызовом метода clearForm.

Компонент TrainList отвечает за отображение списка поездов, получаемого с сервера. При загрузке компонента вызывается метод fetchTrains, который получает данные о поездах и сохраняет их в состоянии компонента. Для удаления поезда реализован метод deleteTrain, который отправляет запрос на сервер и обновляет список поездов.

Сервис trainService инкапсулирует логику работы с API. Он содержит методы для получения списка поездов (getTrains), добавления нового поезда (addTrain), обновления существующего (updateTrain) и удаления поезда по его id (deleteTrain). Это позволяет централизовать работу с сервером и упростить дальнейшую поддержку кода.

Основное приложение подключено к axios с базовым URL http://localhost:8081. Это обеспечивает единый адрес для всех запросов и позволяет удобно управлять настройками API.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

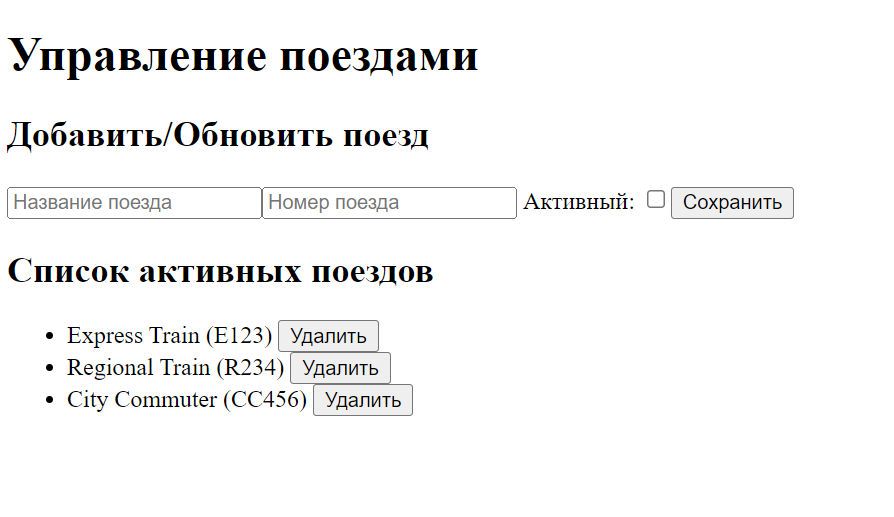


Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

<template>  
 <div>  
 <h2>Добавить/Обновить поезд</h2>  
 <form @submit.prevent="submitTrain">  
 <input v-model="train.name" placeholder="Название поезда" required/>  
 <input v-model="train.number" placeholder="Номер поезда" required/>  
 <label>  
 Активный:  
 <input type="checkbox" v-model="train.isActive"/>  
 </label>  
 <button type="submit">Сохранить</button>  
 </form>  
 </div>  
</template>  
  
<script>  
import trainService from '@/services/trainService';  
  
export default {  
 data() {  
 return {  
 train: {  
 id: null,  
 name: '',  
 number: '',  
 isActive: false,  
 },  
 };  
 },  
 methods: {  
 async submitTrain() {  
 if (this.train.id) {  
 await trainService.updateTrain(this.train);  
 } else {  
 await trainService.addTrain(this.train);  
 }  
 this.clearForm();  
 },  
 clearForm() {  
 this.train = {id: null, name: '', number: '', isActive: false};  
 },  
 },  
};  
</script>

<template>  
 <div>  
 <h2>Список активных поездов</h2>  
 <ul>  
 <li v-for="train in trains" :key="train.id">  
 {{ train.name }} ({{ train.number }})  
 <span v-if="train.isActive">[Активный]</span>  
 <button @click="deleteTrain(train.id)">Удалить</button>  
 </li>  
 </ul>  
 </div>  
</template>  
<script>  
import trainService from '@/services/trainService';  
  
export default {  
 data() {  
 return {  
 trains: [],  
 };  
 },  
 methods: {  
 async fetchTrains() {  
 const response = await trainService.getTrains();  
 this.trains = response.data;  
 },  
 async deleteTrain(id) {  
 await trainService.deleteTrain(id);  
 this.fetchTrains();  
 },  
 },  
 created() {  
 this.fetchTrains();  
 },  
};  
</script>

import axios from 'axios';  
  
const API\_URL = 'http://localhost:8081/trains';  
  
export default {  
 getTrains() {  
 return axios.get(API\_URL);  
 },  
 addTrain(train) {  
 return axios.post(API\_URL, train);  
 },  
 updateTrain(train) {  
 return axios.put(API\_URL, train);  
 },  
 deleteTrain(id) {  
 return axios.delete(`${API\_URL}/${id}`);  
 },  
};

<template>  
 <div id="app">  
 <h1>Управление поездами</h1>  
 <TrainForm />  
 <TrainList />  
 </div>  
</template>  
  
<script>  
import TrainList from '@/components/TrainList.vue';  
import TrainForm from '@/components/TrainForm.vue';  
  
export default {  
 components: {  
 TrainList,  
 TrainForm,  
 },  
};  
</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**

В результате работы было создано приложение для управления списком поездов, включающее добавление, обновление, отображение и удаление поездов.

Приложение успешно разделено на независимые компоненты TrainForm и TrainList, что облегчает его поддержку и масштабирование.

Сервис trainService обеспечивает централизованную обработку всех запросов к API, упрощая взаимодействие с серверной частью.

Функционал протестирован, и все операции — добавление, обновление, удаление и отображение поездов — работают корректно, обеспечивая удобный пользовательский опыт.