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 №13

Выполнил: Веселков М.Ю.

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: ShopForm и ShopList. ShopForm отвечает за добавление и обновление магазинов, а ShopList — за отображение списка магазинов и их удаление. Это позволяет разделить функционал для удобства работы и масштабирования.

В компоненте ShopForm реализована форма с полями для ввода названия и адреса магазина. Метод submitShop отправляет данные на сервер, проверяя наличие id у объекта. Если id существует, происходит обновление магазина через метод updateShop сервиса api, если нет — добавление через метод addShop. После успешного выполнения этих операций форма очищается вызовом метода clearForm.

Компонент ShopList отвечает за отображение списка магазинов, получаемого с сервера. При загрузке компонента вызывается метод fetchShops, который получает данные о магазинах и сохраняет их в состоянии компонента. Для удаления магазина реализован метод deleteShop, который отправляет запрос на сервер и обновляет список магазинов.

Сервис api.js инкапсулирует логику работы с API. Он содержит методы для получения списка магазинов (getShops), добавления нового магазина (addShop), обновления существующего (updateShop) и удаления магазина по его id (deleteShop). Это позволяет централизовать работу с сервером и упростить дальнейшую поддержку кода.

Основное приложение подключено к axios с базовым URL http://localhost:5173/. Это обеспечивает единый адрес для всех запросов и позволяет удобно управлять настройками API.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

<template>  
 <div>  
 <h1>Магазины</h1>  
  
 <!-- Форма для добавления/обновления магазина -->  
 <div>  
 <h3>{{ editing ? "Редактировать магазин" : "Добавить магазин" }}</h3>  
 <input v-model="shopForm.title" placeholder="Название" />  
 <input v-model="shopForm.adress" placeholder="Адрес" />  
 <input v-model="shopForm.phone" placeholder="номер" />  
 <button @click="editing ? updateShop() : addShop()">  
 {{ editing ? "Сохранить изменения" : "Добавить" }}  
 </button>  
 <button v-if="editing" @click="cancelEdit">Отмена</button>  
 </div>  
  
 <!-- Список магазинов -->  
 <ul>  
 <li v-for="shop in shops" :key="shop.id">  
 {{ shop.title }} - {{ shop.adress }} - {{ shop.phone }}  
 <button @click="editShop(shop)">Изменить</button>  
 <button @click="deleteShop(shop.id)">Удалить</button>  
 </li>  
 </ul>  
 </div>  
</template>  
  
<script>  
import { ref, onMounted } from 'vue';  
import { getAllShops, addShop, updateShop, deleteShop } from '@/services/api';  
  
export default {  
 setup() {  
 const shops = ref([]);  
 const shopForm = ref({ title: "", adress: "" });  
 const editing = ref(false);  
 const currentShopId = ref(null);  
  
 // Получение списка магазинов  
 const fetchShops = async () => {  
 try {  
 shops.value = await getAllShops();  
 } catch (error) {  
 *console*.error("Ошибка получения магазинов:", error);  
 }  
 };  
  
 // Добавление нового магазина  
 const addShopHandler = async () => {  
 try {  
 await addShop(shopForm.value);  
 shopForm.value = { title: "", adress: "" };  
 fetchShops(); // Обновить список  
 } catch (error) {  
 *console*.error("Ошибка добавления магазина:", error);  
 }  
 };  
  
 // Удаление магазина  
 const deleteShopHandler = async (shopId) => {  
 try {  
 await deleteShop(shopId);  
 fetchShops(); // Обновить список  
 } catch (error) {  
 *console*.error("Ошибка удаления магазина:", error);  
 }  
 };  
  
 // Начало редактирования  
 const editShop = (shop) => {  
 shopForm.value = { ...shop };  
 currentShopId.value = shop.id;  
 editing.value = true;  
 };  
  
 // Обновление магазина  
 const updateShopHandler = async () => {  
 try {  
 await updateShop({ id: currentShopId.value, ...shopForm.value });  
 shopForm.value = { title: "", adress: "" };  
 editing.value = false;  
 currentShopId.value = null;  
 fetchShops();  
 } catch (error) {  
 *console*.error("Ошибка обновления магазина:", error);  
 }  
 };  
  
 // Отмена редактирования  
 const cancelEdit = () => {  
 shopForm.value = { title: "", adress: "" };  
 editing.value = false;  
 currentShopId.value = null;  
 };  
  
 onMounted(fetchShops);  
  
 return {  
 shops,  
 shopForm,  
 editing,  
 fetchShops,  
 addShop: addShopHandler,  
 deleteShop: deleteShopHandler,  
 editShop,  
 updateShop: updateShopHandler,  
 cancelEdit  
 };  
 }  
};  
</script>

<template>  
 <div>  
 <h1>Добро пожаловать!</h1>  
 <router-link to="/shops">Список магазинов</router-link>  
 </div>  
</template>

import { createRouter, createWebHistory } from 'vue-router';  
import HomeView from '@/views/HomeView.vue';  
import *ShopView* from '@/views/ShopView.vue';  
  
const routes = [  
 { path: '/', component: HomeView },  
 { path: '/shops', component: *ShopView* }  
];  
  
const *router* = createRouter({  
 history: createWebHistory(),  
 routes  
});  
  
export default *router*;

<template>  
 <div>  
 <h1>Магазины</h1>  
  
 <div>  
 <input v-model="shopId" placeholder="Введите ID магазина" />  
 <button @click="fetchShop">Найти</button>  
 </div>  
  
 <div v-if="shop">  
 <h2>{{ shop.title }}</h2>  
 <p>Адрес: {{ shop.adress }}</p>  
 <p>Телефон: {{ shop.phone }}</p>  
 </div>  
  
 <h2>Добавить магазин</h2>  
 <input v-model="newShop.title" placeholder="Название" />  
 <input v-model="newShop.adress" placeholder="Адрес" />  
 <input v-model="newShop.phone" placeholder="Телефон" />  
 <button @click="addShop">Добавить</button>  
 </div>  
</template>  
  
<script>  
import *shopService* from "@/services/shopService";  
  
export default {  
 data() {  
 return {  
 shopId: "",  
 shop: null,  
 newShop: {  
 title: "",  
 adress: "",  
 phone: ""  
 }  
 };  
 },  
 methods: {  
 async fetchShop() {  
 try {  
 const response = await *shopService*.getShopById(this.shopId);  
 this.shop = response.data.data;  
 } catch (error) {  
 *console*.error("Ошибка при получении магазина", error);  
 }  
 },  
 async addShop() {  
 try {  
 await *shopService*.addShop(this.newShop);  
 alert("Магазин добавлен!");  
 this.newShop = { title: "", adress: "", phone: "" };  
 } catch (error) {  
 *console*.error("Ошибка при добавлении", error);  
 }  
 }  
 }  
};  
</script>

import *axios* from "axios";  
  
const API\_URL = "http://localhost:8080/shop/op";  
  
export default {  
 getShopById(shopId) {  
 return axios.post(`${API\_URL}/id`, { shopId });  
 },  
  
 addShop(shop) {  
 return axios.post(`${API\_URL}/add`, shop);  
 },  
  
 updateShop(shop) {  
 return axios.put(`${API\_URL}/update`, shop);  
 },  
  
 deleteShop(shopId) {  
 return axios.delete(`${API\_URL}/delete/${shopId}`);  
 }  
};

<template>  
 <div>  
 <nav>  
 <router-link to="/">Главная</router-link> |  
 <router-link to="/shops">Магазины</router-link>  
 </nav>  
 <router-view />  
 </div>  
</template>  
  
  
<style scoped>  
header {  
 line-height: 1.5;  
}  
  
.logo {  
 display: block;  
 margin: 0 auto 2rem;  
}  
  
@media (min-width: 1024px) {  
 header {  
 display: flex;  
 place-items: center;  
 padding-right: calc(var(--section-gap) / 2);  
 }  
  
 .logo {  
 margin: 0 2rem 0 0;  
 }  
  
 header .wrapper {  
 display: flex;  
 place-items: flex-start;  
 flex-wrap: wrap;  
 }  
}  
</style>

import './assets/main.css'  
  
import { createApp } from 'vue'  
import App from './App.vue'  
import *router* from './router';  
  
const app = createApp(App);  
app.use(*router*);  
app.mount('#app');

import { defineConfig } from 'vite';  
import vue from '@vitejs/plugin-vue';  
import path from 'path';  
  
export default defineConfig({  
 plugins: [vue()],  
 resolve: {  
 alias: {  
 '@': path.resolve(\_\_dirname, 'src'), // Алиас для `@/`  
 },  
 },  
 server: {  
 proxy: {  
 '/api': {  
 target: 'http://localhost:8080',  
 changeOrigin: true,  
 rewrite: (path) => path.replace(/^\/api/, ''),  
 },  
 },  
 },  
});

# **ВЫВОД**

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

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

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