|  |  |
| --- | --- |
| **Российский университет транспорта (МИИТ)**  **Институт транспортной техники и систем управления**  **Кафедра «Управление и защита информации»** | |
| **Отчет**  **по лабораторной работе №5**  **по дисциплине «Web-программирование» Вариант 12** | |
|  | Выполнили:  Студенты группы ТКИ-541  Кузнецов К. К.  Степанов Д. Е. |
|  | Проверил:  Доцент кафедры УиЗИ к.т.н. с.н.с.  Сафронов А. И. |
| Москва 2024 | |

**Оглавление**

[1 Цель работы 3](#_Toc185165364)

[2 Формулировка задачи 4](#_Toc185165365)

[3 ER-диаграмма данных 6](#_Toc185165366)

[4 Таблица соответствия переменных и методов, используемых в приложениях 7](#_Toc185165367)

[5 Архитектура системы 8](#_Toc185165368)

[6 Инструкция по развертке системы 9](#_Toc185165369)

[7 Содержательная часть 10](#_Toc185165370)

[7.1 Код ProductsController.rb 10](#_Toc185165371)

[7.2 Код main.js 11](#_Toc185165372)

[7.3 Код App.vue 12](#_Toc185165373)

[7.4 Код index.js 12](#_Toc185165374)

[7.5 Код ProductView.vue 13](#_Toc185165375)

[7.6 Код Products.vue 13](#_Toc185165376)

[7.7 Код ProductFormView.vue 18](#_Toc185165377)

[7.8 Код ProductForm.vue 18](#_Toc185165378)

[7.9 Код main.css 23](#_Toc185165379)

[7.10 Отображение в браузере 27](#_Toc185165380)

[7.11 Сеть Петри 29](#_Toc185165381)

[8 Вывод 30](#_Toc185165382)

1. Цель работы

Создать масштабируемое одностраничное *web*-приложение на базе фреймворка *Vue.js* (по шаблону *Vite.js*, подгружаемому командой *npm init vue@latest*), отражающее требуемый функционал и тематику согласно варианту индивидуального задания.

1. Формулировка задачи

**Хранение данных (порт 5432):**

– база данных *PostgreSQL*.

**Backend (порт 8080):**

*Вариант 1:* на базе фреймворка *Express.js* (см. **Приложение 1**).

*Вариант 2:* предварительно согласованная с преподавателем информационная технология (выбор может быть сделан и направлении *Java*-класса при необходимости).

**Frontend (порт 5173):**

– на базе фреймворка *Vue.js* (при подключении шаблона *Vite.js*). Пример шаблона для встраивания в компонент *App.vue* в **Приложении 2**. Пример настройки экземпляра по аналогии с *LSPWA* в **Приложении 3**. **Приложение 3** также содержит пример асинхронных методов получения данных от *backend*. **Приложение 4** содержит пример аналогичного получения данных в обход асинхронных методов. **Приложение 5** содержит пример добавления новой сущности в отношение базы данных.

**Реализация методологии *CRUD*:**

– отображать сущности основного отношения выбранной тематики в функциональных списках (<*select>*</*select>*);

– предоставлять пользователю возможность редактирования атрибутов выбранной / выбранных из списка сущностей;

– предоставлять пользователю возможность для добавления новых сущностей;

– предоставлять возможность удаления избыточных сущностей.

**Дружественный к пользователю графический интерфейс:**

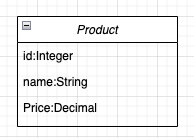
– отдельные окна (шаблоны / *templates*) для решения отдельных задач по тематике – не решать все задачи в одном окне (шаблоне / *template*);

– разместить информацию от наиболее приоритетной к менее приоритетной (эргономичность ГПИ – [*https://elibrary.ru/item.asp?id=68533620*](https://elibrary.ru/item.asp?id=68533620));

– предусмотреть демонстрацию иллюстрации «ожидание» в момент ожидания пользователем откликов на запросы к базе данных;

– предусмотреть текстовую справку-инструкцию по работе с системой.

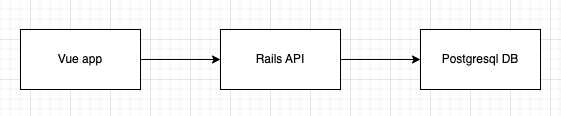
1. ER-диаграмма данных



1. Таблица соответствия переменных и методов, используемых в приложениях

|  |  |
| --- | --- |
| Переменная/метод | Описание |
| products | Массив продуктов |
| productForm | Данные формы создания/обновления продукта |
| isEditing | Тип формы |
| errors | Объект ошибок |
| loading | Ожидание откликов на запросы к базе данных |
| apiUrl | Ссылка на API |
| fetchProducts() | Извлечение существующих продуктов |
| createProduct() | Создание продукта |
| updateProduct() | Обновление продукта |
| deleteProduct() | Удаление продукта |
| editProduct() | Подстановка данных изменияемого продукта в форму |
| resetForm() | Установка значений по умолчанию в форму |

1. Архитектура системы



1. Инструкция по развертке системы

**Backend:**

Ruby 3.0.3

Rails 7.1.5

**Frontend:**

Node: 22.12.0

Vite: 6.0.3

**Db:**

PostgreSQL: 14.8

Запуск rails приложения (в папке bakery):

bundle install

bundle exec rails db:setup – создание базы данных

bundle exec rails db:migrate – создание таблицы

bundle exec rails s – запуск сервера

Запуск vue приложения (в папке bakery\_vue):

npm install – установка зависимостей

npm run dev – запуск сервера

1. Содержательная часть
   1. Код ProductsController.rb

class ProductsController < ApplicationController

skip\_before\_action :verify\_authenticity\_token

def index

products = Product.all

render json: products

end

def create

product = Product.new(product\_params)

if product.save

render json: product, status: :created

else

render json: product.errors, status: :unprocessable\_entity

end

end

def update

product = Product.find(params[:id])

if product.update(product\_params)

render json: product, status: :ok

else

render json: product.errors, status: :unprocessable\_entity

end

end

def destroy

product = Product.find(params[:id])

product.destroy

head :no\_content

end

def edit

product = Product.find(params[:id])

render json: product

end

private

def product\_params

params.require(:product).permit(:name, :price)

end

end

* 1. Код main.js

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')

* 1. Код App.vue

<script setup>

import { RouterLink, RouterView } from 'vue-router'

</script>

<template>

<RouterView />

</template>

* 1. Код index.js

import { createRouter, createWebHistory } from 'vue-router'

import ProductView from '../views/ProductView.vue'

import ProductFormView from '../views/ProductFormView.vue'

const router = createRouter({

history: createWebHistory(import.meta.env.BASE\_URL),

routes: [

{

path: '/products',

name: 'products',

component: ProductView,

},

{

path: '/products/:id/edit',

name: 'edit product',

component: ProductFormView,

},

{

path: '/products/new',

name: 'new product',

component: ProductFormView,

}

],

})

export default router

* 1. Код ProductView.vue

<script setup>

import Products from '../components/Products.vue'

</script>

<template>

<main>

<Products />

</main>

</template>

* 1. Код Products.vue

<template>

<div>

<h1>Product List</h1>

<div v-if="loading" class="loading-spinner">

<span class="loader"></span>

</div>

<div v-else>

<div>

<ul>

<li v-for="product in products" :key="product.id">

{{ product.name }} - ${{ product.price }}

<router-link :to="`/products/${product.id}/edit`">Edit</router-link>

<button @click="deleteProduct(product.id)">Delete</button>

</li>

</ul>

<router-link :to="`/products/new`">Add Product</router-link>

</div>

</div>

</div>

</template>

<script>

export default {

data() {

return {

products: [],

productForm: {

name: '',

price: ''

},

isEditing: false,

errors: {},

loading: false,

apiUrl: 'http://localhost:3000/products' // Adjust the API URL as needed

};

},

mounted() {

this.fetchProducts();

},

methods: {

async fetchProducts() {

this.loading = true;

try {

const response = await fetch(this.apiUrl);

this.products = await response.json();

} catch (error) {

console.error('Error fetching products:', error);

} finally {

this.loading = false;

}

},

async createProduct() {

this.errors = {};

this.loading = true;

try {

const response = await fetch(this.apiUrl, {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(this.productForm)

});

if (!response.ok) {

const errorData = await response.json();

this.errors = errorData || {};

return;

}

const newProduct = await response.json();

this.products.push(newProduct);

this.resetForm();

} catch (error) {

console.error('Error creating product:', error);

} finally {

this.loading = false;

}

},

async updateProduct() {

this.errors = {};

this.loading = true;

try {

const response = await fetch(`${this.apiUrl}/${this.productForm.id}`, {

method: 'PUT',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(this.productForm)

});

if (!response.ok) {

const errorData = await response.json();

this.errors = errorData || {};

return;

}

const updatedProduct = await response.json();

const index = this.products.findIndex(product => product.id === updatedProduct.id);

if (index !== -1) {

this.products.splice(index, 1, updatedProduct); // Directly modify the array

}

this.resetForm();

} catch (error) {

console.error('Error updating product:', error);

} finally {

this.loading = false;

}

},

async deleteProduct(id) {

this.loading = true;

try {

await fetch(`${this.apiUrl}/${id}`, {

method: 'DELETE'

});

this.products = this.products.filter(product => product.id !== id);

} catch (error) {

console.error('Error deleting product:', error);

} finally {

this.loading = false;

}

},

}

};

</script>

* 1. Код ProductFormView.vue

<script setup>

import ProductForm from '../components/ProductForm.vue'

</script>

<template>

<main>

<ProductForm />

</main>

</template>

* 1. Код ProductForm.vue

<template>

<div>

<h1>Product Management</h1>

<div v-if="loading" class="loading-spinner">

<span class="loader"></span>

</div>

<div v-else>

<div>

<h2>{{ isEditing ? 'Edit Product' : 'Add Product' }}</h2>

<form @submit.prevent="isEditing ? updateProduct() : createProduct()">

<div>

<label for="name">Name:</label>

<input type="text" v-model="productForm.name" required>

<span v-if="errors.name" class="error-message">{{ errors.name[0] }}</span>

</div>

<div>

<label for="price">Price:</label>

<input type="number" v-model="productForm.price" required>

<span v-if="errors.price" class="error-message">{{ errors.price[0] }}</span>

</div>

<button type="submit">{{ isEditing ? 'Update' : 'Add' }}</button>

<button type="button" @click="resetForm">Cancel</button>

</form>

</div>

</div>

</div>

</template>

<script>

export default {

data() {

return {

products: [],

productForm: {

name: '',

price: ''

},

isEditing: false,

errors: {},

loading: false,

apiUrl: 'http://localhost:3000/products' // Adjust the API URL as needed

};

},

mounted() {

this.isEditing = this.$route.path.includes('/edit');

if (this.isEditing) {

this.fetchProduct();

}

},

methods: {

async fetchProduct() {

this.loading = true;

try {

const response = await fetch(`${this.apiUrl}/${this.$route.params.id}/edit`);

const parsed\_response = await response.json();

this.productForm = {

id: parsed\_response.id,

name: parsed\_response.name,

price: parsed\_response.price

}

console.log(this.productForm);

} catch (error) {

console.error('Error fetching products:', error);

} finally {

this.loading = false;

}

},

async createProduct() {

this.errors = {};

this.loading = true;

try {

const response = await fetch(this.apiUrl, {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(this.productForm)

});

if (!response.ok) {

const errorData = await response.json();

this.errors = errorData || {};

return;

}

const newProduct = await response.json();

this.products.push(newProduct);

this.resetForm();

this.$router.push('/products');

} catch (error) {

console.error('Error creating product:', error);

} finally {

this.loading = false;

}

},

async updateProduct() {

this.errors = {};

this.loading = true;

try {

const response = await fetch(`${this.apiUrl}/${this.productForm.id}`, {

method: 'PUT',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(this.productForm)

});

if (!response.ok) {

const errorData = await response.json();

this.errors = errorData || {};

return;

}

const updatedProduct = await response.json();

const index = this.products.findIndex(product => product.id === updatedProduct.id);

if (index !== -1) {

this.products.splice(index, 1, updatedProduct); // Directly modify the array

}

this.resetForm();

this.$router.push('/products');

} catch (error) {

console.error('Error updating product:', error);

} finally {

this.loading = false;

}

},

resetForm() {

this.productForm = {

name: '',

price: ''

};

this.errors = {};

}

}

};

</script>

* 1. Код main.css

@import './base.css';

#app {

max-width: 1280px;

margin: 0 auto;

padding: 2rem;

font-weight: normal;

}

a,

.green {

text-decoration: none;

color: hsla(160, 100%, 37%, 1);

transition: 0.4s;

padding: 3px;

}

@media (hover: hover) {

a:hover {

background-color: hsla(160, 100%, 37%, 0.2);

}

}

@media (min-width: 1024px) {

body {

display: flex;

place-items: center;

}

#app {

display: grid;

grid-template-columns: 1fr 1fr;

padding: 0 2rem;

}

}

.loader {

transform: rotateZ(45deg);

perspective: 1000px;

border-radius: 50%;

width: 48px;

height: 48px;

color: #fff;

position: fixed;

top: 50%;

left: 50%;

}

.loader:before,

.loader:after {

content: '';

display: block;

position: absolute;

top: 0;

left: 0;

width: inherit;

height: inherit;

border-radius: 50%;

transform: rotateX(70deg);

animation: 1s spin linear infinite;

}

.loader:after {

color: #FF3D00;

transform: rotateY(70deg);

animation-delay: .4s;

}

@keyframes rotate {

0% {

transform: translate(-50%, -50%) rotateZ(0deg);

}

100% {

transform: translate(-50%, -50%) rotateZ(360deg);

}

}

@keyframes rotateccw {

0% {

transform: translate(-50%, -50%) rotate(0deg);

}

100% {

transform: translate(-50%, -50%) rotate(-360deg);

}

}

@keyframes spin {

0%,

100% {

box-shadow: .2em 0px 0 0px currentcolor;

}

12% {

box-shadow: .2em .2em 0 0 currentcolor;

}

25% {

box-shadow: 0 .2em 0 0px currentcolor;

}

37% {

box-shadow: -.2em .2em 0 0 currentcolor;

}

50% {

box-shadow: -.2em 0 0 0 currentcolor;

}

62% {

box-shadow: -.2em -.2em 0 0 currentcolor;

}

75% {

box-shadow: 0px -.2em 0 0 currentcolor;

}

87% {

box-shadow: .2em -.2em 0 0 currentcolor;

}

}

* 1. Отображение в браузере

|  |
| --- |
|  |
| 1. – Начальное состояние |

|  |
| --- |
|  |
| 1. – Обновление продукта |

|  |  |
| --- | --- |
|  |  |
| 1. – Создание продукта | 1. – Начальное состояние |

* 1. Сеть Петри

|  |
| --- |
|  |
| 1. – Сеть Петри |

1. Вывод

В результате проведенной работы были разработаны четыре простых локальных одностраничных web-приложение (LSPWA) под управлением фреймворка Vue.js на языке JavaScript.