|  |  |
| --- | --- |
| **Российский университет транспорта (МИИТ)**  **Институт транспортной техники и систем управления**  **Кафедра «Управление и защита информации»** | |
| **Задание №5**  **по теме «*SPWA via Node.js*»**  **по дисциплине «Web-программирование»** | |
|  | Выполнил:  Студент группы ТКИ-542  Пономарев А.Д.  Дроздов А.Д.  Проверил:  Доцент кафедры УиЗИ, к.т.н., с.н.с  Сафронов А.И. |
| Москва 2024 | |

Оглавление

[1. Цель работы 3](#_Toc181105810)

[2. ФОРМУЛИРОВАКА ЗАДАЧИ 3](#_Toc181105811)

[3. Пункт 1 (Web-страница 1) 4](#_Toc181105812)

[3.1. Формулировка задачи 4](#_Toc181105813)

[3.2. Код web-приложения 4](#_Toc181105814)

[3.3. Результат отображения в браузере 10](#_Toc181105815)

[3.4. Схема Сетей Петри 11](#_Toc181105816)

[4. Пункт 1 (Web-страница 1) 12](#_Toc181105817)

[4.1. Формулировка задачи 12](#_Toc181105818)

[4.2. Код web-приложения 12](#_Toc181105819)

[4.3. Результат отображения в браузере 14](#_Toc181105820)

[4.4. Схема Сетей Петри 15](#_Toc181105821)

[5. пункт 3 (Web-старница 3) 16](#_Toc181105822)

[5.1. Формулировка задачи 16](#_Toc181105823)

[5.2. Код web-приложения 16](#_Toc181105824)

[5.3. Результат отображения в браузере 17](#_Toc181105825)

[5.4. Схема Сетей Петри 18](#_Toc181105826)

[6. вывод по работе 19](#_Toc181105827)

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. ФОРМУЛИРОВАКА ЗАДАЧИ

Разработать фронтенд на Vue.js+Vite.js (порт 5173), бэкенд на базе фреймворка Express.js (порт 8080), базу данных PostgreSQL (порт 5432). Также требуется реализовать методологию CRUD, а именно:

* 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.

Дополнительно необходимо имплементировать дружественный к пользователю графический интерфейс, а именно:

* отдельные окна (шаблоны / templates) для решения отдельных задач по тематике – не решать все задачи в одном окне (шаблоне / template) – использовать компоненты Vue.js;
* разместить информацию от наиболее приоритетной к менее приоритетной;
* предусмотреть демонстрацию иллюстрации «ожидание» в момент ожидания пользователем откликов на запросы к базе данных;
* предусмотреть текстовую справку-инструкцию по работе с системой.

1. Описание предметной области
   1. Формулировка задачи

Разработать текстовую базу данных и веб-оболочку на тему «Транспортная компания». База данных создаётся для обслуживания работников и клиентов компании. Конечный вариант базы данных содержит в себе информацию о доступных видах транспорта для перевозки, цены на перевозки, а также данные о потенциальных покупателях. Готовые запросы:

* Выдавать список маршрутов, обслуживаемых компанией и цену на них;
* Выдавать список транспорта, занятого на данном маршруте;
* Выдавать список вариантов проезда по данному маршруту (морской и т. д.);
* Рассчитывать стоимость услуг, оказанных по перевозке данного груза. Постоянным клиентам предоставляется скидка;
* Показывать грузооборот по данному маршруту.
  1. ER-диаграмма данных

|  |
| --- |
| D:\Новая папка (5)\Институт\СУБД 4 курс\12.png |
| 1. – ER-диаграмма |

* 1. Таблица соответствия переменных

Таблица 1 – Сущности Базы Данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование сущности** | **Содержание атрибута** | **Имя атрибута** | **Тип** | **Примечание** |
| Клиенты | Идентификатор клиента | id\_client | Целое число | Первичный ключ |
| Имя клиента | name | Строка (255) | Обязательное, уникальное  поле |
| Телефон | phone | Строка (255) | Обязательное, уникальное поле |
| Адрес | address | Строка (255) | Обязательное, уникальное поле |
| Скидка | discount | numeric (11,2) | Обязательное поле, целое число не может быть отрицательным |
| Транспорт | Идентификатор транспорта | id\_transport | Целое число | Первичный ключ |
| Тип транспорта | type | Строка (255) | Обязательное, уникальное  поле |
| Цена | cost | numeric (11,2) | Обязательное поле, целое число не может быть отрицательным |
| Заказы | Идентификатор заказа | id\_order | Целое число | Первичный ключ |
| Идентификатор транспорта | id\_transport | Целое число | Внешний ключ к отношению (сущности) транспорты |
| Идентификатор клиента | id\_client | Целое число | Внешний ключ к отношению (сущности) клиенты |
| Дата | date | Строка | Обязательное поле |
| Цена | cost | numeric (11,2) | Обязательное поле, целое число не может быть отрицательным |

* 1. Архитектура системы (связка: frontend, backend, database)

|  |
| --- |
|  |
| 1. – Архитектура системы |

1. Web-страница
   1. Инструкция по развёртке системы (подгрузка запрашиваемых (required) модулей, взаимная увязка технологий).
2. Установка Node.js и npm:

* sudo apt update
* sudo apt install nodejs npm

1. Установка PostgreSQL:

* sudo apt install postgresql postgresql-contrib

1. Установка Axios:

* npm install axios

1. Установка зависимостей для сервиса:

* npm install express cors pg
  1. Код web-приложения (server.js)

import express from 'express';

import cors from 'cors';

import pg from 'pg';

const pool = new pg.Pool({

user: 'admin',

host: 'localhost',

database: 'trcompany',

port: 5432

});

const app = express();

app.use(express.json());

app.use(cors());

class ClientsController {

// Получить список всех клиентов

async getClients(req, res) {

try {

const clients = await pool.query('SELECT \* FROM clients');

res.json(clients.rows);

} catch (error) {

console.error(error);

res.status(500).json({ error: 'Error fetching clients' });

}

}

// Добавить нового клиента

async addClient(req, res) {

const { name, address, phone, discount } = req.body;

try {

const newClient = await pool.query(

'INSERT INTO clients (name, address, phone, discount) VALUES ($1, $2, $3, $4) RETURNING \*',

[name, address, phone, discount]

);

res.json(newClient.rows[0]);

} catch (error) {

console.error(error);

res.status(500).json({ error: 'Error adding client' });

}

}

// Обновить существующего клиента

async updateClient(req, res) {

const { id } = req.params;

const { name, address, phone, discount } = req.body;

try {

const updatedClient = await pool.query(

'UPDATE clients SET name = $1, address = $2, phone = $3, discount = $4 WHERE id = $5 RETURNING \*',

[name, address, phone, discount, id]

);

if (updatedClient.rows.length > 0) {

res.json(updatedClient.rows[0]);

} else {

res.status(404).json({ error: 'Client not found' });

}

} catch (error) {

console.error(error);

res.status(500).json({ error: 'Error updating client' });

}

}

// Удалить клиента

async deleteClient(req, res) {

const { id } = req.params;

try {

const result = await pool.query('DELETE FROM clients WHERE id = $1', [id]);

if (result.rowCount > 0) {

res.json({ message: 'Client deleted successfully' });

} else {

res.status(404).json({ error: 'Client not found' });

}

} catch (error) {

console.error(error);

res.status(500).json({ error: 'Error deleting client' });

}

}

}

const clientsController = new ClientsController();

// Маршруты для методов

app.get('/clients', (req, res) => clientsController.getClients(req, res));

app.post('/clients', (req, res) => clientsController.addClient(req, res));

app.put('/clients/:id', (req, res) => clientsController.updateClient(req, res));

app.delete('/clients/:id', (req, res) => clientsController.deleteClient(req, res));

const PORT = process.env.PORT || 8080;

app.listen(PORT, () => {

console.log(`Server running on port ${PORT}`);

});

* 1. Код Web-приложения (Clients.vue)

<template>

<div>

<h2>Клиенты</h2>

<button @click="showAddForm = true">Добавить клиента</button>

<button @click="showUpdateForm = true" :disabled="clients.length === 0">Обновить клиента</button>

<button @click="showDeleteForm = true" :disabled="clients.length === 0">Удалить клиента</button>

<!-- Форма для добавления клиента -->

<div v-if="showAddForm">

<h3>Добавить клиента</h3>

<form @submit.prevent="addClient">

<input v-model="newClient.name" placeholder="Имя" required />

<input v-model="newClient.address" placeholder="Адрес" required />

<input v-model="newClient.phone" placeholder="Телефон" required />

<input v-model="newClient.discount" type="number" placeholder="Скидка" required />

<button type="submit">Добавить</button>

<button type="button" @click="showAddForm = false">Отмена</button>

</form>

</div>

<!-- Форма для обновления клиента -->

<div v-if="showUpdateForm">

<h3>Обновить клиента</h3>

<form @submit.prevent="updateClient">

<select v-model="selectedClient" required>

<option disabled value="">Выберите клиента</option>

<option v-for="client in clients" :value="client" :key="client.id">

{{ client.name }}

</option>

</select>

<input v-model="selectedClient.name" placeholder="Имя" required v-if="selectedClient" />

<input v-model="selectedClient.address" placeholder="Адрес" required v-if="selectedClient" />

<input v-model="selectedClient.phone" placeholder="Телефон" required v-if="selectedClient" />

<input v-model="selectedClient.discount" type="number" placeholder="Скидка" required v-if="selectedClient" />

<button type="submit">Обновить</button>

<button type="button" @click="showUpdateForm = false">Отмена</button>

</form>

</div>

<!-- Форма для удаления клиента -->

<div v-if="showDeleteForm">

<h3>Удалить клиента</h3>

<form @submit.prevent="deleteClient">

<select v-model="selectedClient" required>

<option disabled value="">Выберите клиента</option>

<option v-for="client in clients" :value="client" :key="client.id">

{{ client.name }}

</option>

</select>

<button type="submit">Удалить</button>

<button type="button" @click="showDeleteForm = false">Отмена</button>

</form>

</div>

<ul>

<li v-for="client in clients" :key="client.id">

{{ client.name }} - {{ client.address }} - {{ client.phone }} - {{ client.discount }}

</li>

</ul>

</div>

</template>

<script>

export default {

data() {

return {

clients: [], // Массив клиентов

newClient: {

name: '',

address: '',

phone: '',

discount: ''

},

selectedClient: null,

showAddForm: false,

showUpdateForm: false,

showDeleteForm: false,

};

},

methods: {

addClient() {

const client = {

id: Date.now(), // Генерация уникального идентификатора

name: this.newClient.name,

address: this.newClient.address,

phone: this.newClient.phone,

discount: this.newClient.discount,

};

this.clients.push(client);

this.resetNewClient();

this.showAddForm = false; // Скрываем форму

},

updateClient() {

const index = this.clients.findIndex(client => client.id === this.selectedClient.id);

if (index !== -1) {

this.clients[index] = { ...this.selectedClient }; // Обновление данных клиента

}

this.selectedClient = null; // Сбрасываем выбранного клиента

this.showUpdateForm = false; // Скрываем форму

},

deleteClient() {

this.clients = this.clients.filter(client => client.id !== this.selectedClient.id); // Удаление клиента

this.selectedClient = null; // Сбрасываем выбранного клиента

this.showDeleteForm = false; // Скрываем форму

},

resetNewClient() {

this.newClient.name = '';

this.newClient.address = '';

this.newClient.phone = '';

this.newClient.discount = '';

}

}

};

</script>

<style scoped>

button {

margin: 5px;

}

form {

margin-bottom: 20px;

}

input, select {

display: block;

margin: 5px 0;

padding: 8px;

width: 200px;

}

ul {

margin-top: 20px;

}

</style>

* 1. Результат отображения в браузере

|  |
| --- |
|  |
| 1. – Результат |
|  |
| 1. – Результат |
|  |
| 1. – Результат |
|  |
| 1. – Результат |

* 1. Схема Сетей Петри

|  |
| --- |
|  |
| 1. – Сети Петри |

1. вывод по работе

В результате проведенной работы было создано 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, а именно была разработана текстовая база данных и веб-оболочка на тему «Библиотека» с соответствующими запросами.