|  |  |
| --- | --- |
| **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](#_Toc182836273)

[2. ФОРМУЛИРОВАКА ЗАДАЧИ 4](#_Toc182836274)

[3. Описание предметной области 5](#_Toc182836275)

[3.1. Формулировка задачи 5](#_Toc182836276)

[3.2. ER-диаграмма данных 5](#_Toc182836277)

[3.3. Таблица соответствия переменных 6](#_Toc182836278)

[3.4. Архитектура системы (связка: frontend, backend, database) 7](#_Toc182836279)

[4. Web-страница 8](#_Toc182836280)

[4.1. Инструкция по развёртке системы (подгрузка запрашиваемых (required) модулей, взаимная увязка технологий). 8](#_Toc182836281)

[4.2. Код web-приложения (server.js) 8](#_Toc182836282)

[4.3. Код Web-приложения (ClientList.vue) 13](#_Toc182836283)

[4.4. Код Web-приложения (TransportList.vue) 17](#_Toc182836284)

[4.5. Результат отображения в браузере 21](#_Toc182836285)

[4.6. Схема Сетей Петри 24](#_Toc182836286)

[5. вывод по работе 24](#_Toc182836287)

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: 'postgres',

host: 'localhost',

database: 'trcompany',

port: 5432

});

const app = express();

app.use(express.json());

app.use(cors());

class ClientController {

// Получить список всех клиентов

async getClient(req, res) {

try {

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

res.json(client.rows);

} catch (error) {

console.error(error);

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

}

}

// Добавить нового клиента

async addClient(req, res) {

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

if (!name || !address || !phone || !discount) {

return res.status(400).json({ error: 'Name, address, and phone are required' });

}

try {

const newClient = await pool.query(

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

[name, address, phone, discount]

);

res.status(201).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;

if (!name || !address || !phone || !discount) {

return res.status(400).json({ error: 'Name, address, and phone are required' });

}

const clientId = parseInt(id, 10);

if (isNaN(clientId)) {

return res.status(400).json({ error: 'Invalid client ID' });

}

try {

const updatedClient = await pool.query(

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

[name, address, phone, discount, clientId]

);

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;

if (!id || isNaN(id)) {

return res.status(400).json({ error: 'Invalid client ID' });

}

const clientId = parseInt(id, 10);

console.log(`Deleting client with ID: ${clientId}`); // Отладочное сообщение

try {

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

if (result.rowCount > 0) {

console.log(`Client with ID ${clientId} deleted successfully`); // Отладочное сообщение

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

} else {

console.log(`Client with ID ${clientId} not found`); // Отладочное сообщение

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

}

} catch (error) {

console.error(error);

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

}

}

}

class TransportController {

// Получить список всего транспорта

async getTransport(req, res) {

try {

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

res.json(transport.rows);

} catch (error) {

console.error(error);

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

}

}

// Добавить новый транспорт

async addTransport(req, res) {

const { type, cost } = req.body;

if (!type || !cost) {

return res.status(400).json({ error: 'Type and cost are required' });

}

try {

const newTransport = await pool.query(

'INSERT INTO transport (type, cost) VALUES ($1, $2) RETURNING \*',

[type, cost]

);

res.status(201).json(newTransport.rows[0]);

} catch (error) {

console.error(error);

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

}

}

// Обновить существующий транспорт

async updateTransport(req, res) {

const { id } = req.params;

const { type, cost } = req.body;

if (!type || !cost) {

return res.status(400).json({ error: 'Type and cost are required' });

}

const transportId = parseInt(id, 10);

if (isNaN(transportId)) {

return res.status(400).json({ error: 'Invalid transport ID' });

}

try {

const updatedTransport = await pool.query(

'UPDATE transport SET type = $1, cost = $2 WHERE id = $3 RETURNING \*',

[type, cost, transportId]

);

if (updatedTransport.rows.length > 0) {

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

} else {

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

}

} catch (error) {

console.error(error);

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

}

}

// Удалить транспорт

async deleteTransport(req, res) {

const { id } = req.params;

if (!id || isNaN(id)) {

return res.status(400).json({ error: 'Invalid transport ID' });

}

const transportId = parseInt(id, 10);

console.log(`Deleting transport with ID: ${transportId}`); // Отладочное сообщение

try {

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

if (result.rowCount > 0) {

console.log(`Transport with ID ${transportId} deleted successfully`); // Отладочное сообщение

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

} else {

console.log(`Transport with ID ${transportId} not found`); // Отладочное сообщение

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

}

} catch (error) {

console.error(error);

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

}

}

}

const clientController = new ClientController();

const transportController = new TransportController();

// Маршруты для методов клиентов

app.get('/client', (req, res) => clientController.getClient(req, res));

app.post('/client', (req, res) => clientController.addClient(req, res));

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

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

// Маршруты для методов транспорта

app.get('/transport', (req, res) => transportController.getTransport(req, res));

app.post('/transport', (req, res) => transportController.addTransport(req, res));

app.put('/transport/:id', (req, res) => transportController.updateTransport(req, res));

app.delete('/transport/:id', (req, res) => transportController.deleteTransport(req, res));

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

app.listen(PORT, () => {

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

});

* 1. Код Web-приложения (ClientList.vue)

<template>

<div>

<h1>Список клиентов</h1>

<div>

<label for="showClient">Показать клиентов в БД:</label>

<select id="showClient" v-model="selectedClient">

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

{{ client.name }}

</option>

</select>

</div>

<div>

<button @click="toggleAddClientForm">Добавить клиента</button>

</div>

<div>

<button @click="toggleUpdateClientForm">Обновить клиента</button>

<select v-if="showUpdateClientForm" v-model="selectedClientForUpdate">

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

{{ client.name }}

</option>

</select>

</div>

<div>

<button @click="toggleDeleteClientForm">Удалить клиента</button>

<select v-if="showDeleteClientForm" v-model="selectedClientForDelete">

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

{{ client.name }}

</option>

</select>

</div>

<!-- Форма для добавления нового клиента -->

<div v-if="showAddClientForm">

<h2>Добавить нового клиента</h2>

<p>Имя: <input v-model="newClient.name"></p>

<p>Адрес: <input v-model="newClient.address"></p>

<p>Телефон: <input v-model="newClient.phone"></p>

<p>Скидка: <input v-model="newClient.discount"></p>

<button @click="addClient">Сохранить клиента</button>

</div>

<!-- Форма для обновления информации о клиенте -->

<div v-if="showUpdateClientForm && selectedClientForUpdate">

<h2>Обновить информацию о клиенте</h2>

<p>Имя: <input v-model="selectedClientForUpdate.name"></p>

<p>Адрес: <input v-model="selectedClientForUpdate.address"></p>

<p>Телефон: <input v-model="selectedClientForUpdate.phone"></p>

<p>Скидка: <input v-model="selectedClientForUpdate.discount"></p>

<button @click="saveUpdatedClient">Сохранить изменения</button>

</div>

<!-- Форма для удаления клиента -->

<div v-if="showDeleteClientForm && selectedClientForDelete">

<h2>Удалить клиента</h2>

<p>Вы уверены, что хотите удалить клиента {{ selectedClientForDelete.name }}?</p>

<button @click="confirmDeleteClient">Да, удалить</button>

</div>

<!-- Информация о выбранном клиенте -->

<div v-if="selectedClient">

<h2>Информация о клиенте</h2>

<p>Имя: <input v-model="selectedClient.name"></p>

<p>Адрес: <input v-model="selectedClient.address"></p>

<p>Телефон: <input v-model="selectedClient.phone"></p>

<p>Скидка: <input v-model="selectedClient.discount"></p>

</div>

</div>

</template>

<script>

import axios from 'axios';

export default {

data() {

return {

lstClient: [],

selectedClient: null,

selectedClientForUpdate: null,

selectedClientForDelete: null,

newClient: {

name: '',

address: '',

phone: '',

discount: ''

},

showAddClientForm: false,

showUpdateClientForm: false,

showDeleteClientForm: false,

host: 'http://localhost:8080'

};

},

mounted() {

this.getListByReference('/client');

},

methods: {

getListByReference: async function(refer) {

try {

const res = await axios.get(this.host + refer);

const data = res.data;

this.lstClient = data;

} catch (e) {

console.log('Ошибка: ', e);

}

},

toggleAddClientForm: function() {

this.showAddClientForm = !this.showAddClientForm;

if (!this.showAddClientForm) {

this.newClient = {

name: '',

address: '',

phone: '',

discount: ''

};

}

},

toggleUpdateClientForm: function() {

this.showUpdateClientForm = !this.showUpdateClientForm;

if (!this.showUpdateClientForm) {

this.selectedClientForUpdate = null;

}

},

toggleDeleteClientForm: function() {

this.showDeleteClientForm = !this.showDeleteClientForm;

if (!this.showDeleteClientForm) {

this.selectedClientForDelete = null;

}

},

addClient: async function() {

if (!this.newClient.name || !this.newClient.address || !this.newClient.phone) {

console.log('Введите имя, адрес и телефон');

return;

}

try {

const res = await axios.post(this.host + '/client', this.newClient);

this.lstClient.push(res.data);

this.toggleAddClientForm();

} catch (e) {

console.log('Ошибка: ', e);

}

},

saveUpdatedClient: async function() {

if (!this.selectedClientForUpdate) {

console.log('Выберите клиента для обновления');

return;

}

const updatedClient = {

id: this.selectedClientForUpdate.id,

name: this.selectedClientForUpdate.name,

address: this.selectedClientForUpdate.address,

phone: this.selectedClientForUpdate.phone,

discont: this.selectedClientForUpdate.discount

};

try {

const res = await axios.put(this.host + '/client/' + updatedClient.id, updatedClient);

const index = this.lstClient.findIndex(client => client.id === res.data.id);

this.lstClient.splice(index, 1, res.data);

this.toggleUpdateClientForm();

} catch (e) {

console.log('Ошибка: ', e);

}

},

confirmDeleteClient: async function() {

if (!this.selectedClientForDelete) {

console.log('Выберите клиента для удаления');

return;

}

try {

const res = await axios.delete(this.host + '/client/' + this.selectedClientForDelete.id);

const index = this.lstClient.findIndex(client => client.id === res.data.id);

this.lstClient.splice(index, 1);

this.toggleDeleteClientForm();

} catch (e) {

console.log('Ошибка: ', e);

}

}

}

};

</script>

* 1. Код Web-приложения (TransportList.vue)

<template>

<div>

<h1>Список транспорта</h1>

<div>

<label for="showTransport">Показать транспорт в БД:</label>

<select id="showTransport" v-model="selectedTransport">

<option v-for="transport in lstTransport" :value="transport" :key="transport.id">

{{ transport.type }}

</option>

</select>

</div>

<div>

<button @click="toggleAddTransportForm">Добавить транспорт</button>

</div>

<div>

<button @click="toggleUpdateTransportForm">Обновить транспорт</button>

<select v-if="showUpdateTransportForm" v-model="selectedTransportForUpdate">

<option v-for="transport in lstTransport" :value="transport" :key="transport.id">

{{ transport.type }}

</option>

</select>

</div>

<div>

<button @click="toggleDeleteTransportForm">Удалить транспорт</button>

<select v-if="showDeleteTransportForm" v-model="selectedTransportForDelete">

<option v-for="transport in lstTransport" :value="transport" :key="transport.id">

{{ transport.type }}

</option>

</select>

</div>

<!-- Форма для добавления нового транспорта -->

<div v-if="showAddTransportForm">

<h2>Добавить новый транспорт</h2>

<p>Тип: <input v-model="newTransport.type"></p>

<p>Стоимость: <input v-model="newTransport.cost"></p>

<button @click="addTransport">Сохранить транспорт</button>

</div>

<!-- Форма для обновления информации о транспорте -->

<div v-if="showUpdateTransportForm && selectedTransportForUpdate">

<h2>Обновить информацию о транспорте</h2>

<p>Тип: <input v-model="selectedTransportForUpdate.type"></p>

<p>Стоимость: <input v-model="selectedTransportForUpdate.cost"></p>

<button @click="saveUpdatedTransport">Сохранить изменения</button>

</div>

<!-- Форма для удаления транспорта -->

<div v-if="showDeleteTransportForm && selectedTransportForDelete">

<h2>Удалить транспорт</h2>

<p>Вы уверены, что хотите удалить транспорт {{ selectedTransportForDelete.type }}?</p>

<button @click="confirmDeleteTransport">Да, удалить</button>

</div>

<!-- Информация о выбранном транспорте -->

<div v-if="selectedTransport">

<h2>Информация о транспорте</h2>

<p>Тип: <input v-model="selectedTransport.type"></p>

<p>Стоимость: <input v-model="selectedTransport.cost"></p>

</div>

</div>

</template>

<script>

export default {

data() {

return {

lstTransport: [],

selectedTransport: null,

selectedTransportForUpdate: null,

selectedTransportForDelete: null,

newTransport: {

type: '',

cost: ''

},

showAddTransportForm: false,

showUpdateTransportForm: false,

showDeleteTransportForm: false,

host: 'http://localhost:8080'

};

},

mounted() {

this.getListByReference('/transport');

},

methods: {

getListByReference: async function(refer) {

try {

const res = await fetch(this.host + refer);

const data = await res.json();

this.lstTransport = data;

} catch (e) {

console.log('Ошибка: ', e);

}

},

toggleAddTransportForm: function() {

this.showAddTransportForm = !this.showAddTransportForm;

if (!this.showAddTransportForm) {

this.newTransport = {

type: '',

cost: ''

};

}

},

toggleUpdateTransportForm: function() {

this.showUpdateTransportForm = !this.showUpdateTransportForm;

if (!this.showUpdateTransportForm) {

this.selectedTransportForUpdate = null;

}

},

toggleDeleteTransportForm: function() {

this.showDeleteTransportForm = !this.showDeleteTransportForm;

if (!this.showDeleteTransportForm) {

this.selectedTransportForDelete = null;

}

},

addTransport: async function() {

if (!this.newTransport.type || !this.newTransport.cost) {

console.log('Введите тип и стоимость транспорта');

return;

}

try {

const res = await fetch(this.host + '/transport', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(this.newTransport)

});

const data = await res.json();

this.lstTransport.push(data);

this.toggleAddTransportForm();

} catch (e) {

console.log('Ошибка: ', e);

}

},

saveUpdatedTransport: async function() {

if (!this.selectedTransportForUpdate) {

console.log('Выберите транспорт для обновления');

return;

}

const updatedTransport = {

id: this.selectedTransportForUpdate.id,

type: this.selectedTransportForUpdate.type,

cost: this.selectedTransportForUpdate.cost

};

try {

const res = await fetch(this.host + '/transport/' + updatedTransport.id, {

method: 'PUT',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(updatedTransport)

});

const data = await res.json();

const index = this.lstTransport.findIndex(transport => transport.id === data.id);

this.lstTransport.splice(index, 1, data);

this.toggleUpdateTransportForm();

} catch (e) {

console.log('Ошибка: ', e);

}

},

confirmDeleteTransport: async function() {

if (!this.selectedTransportForDelete) {

console.log('Выберите транспорт для удаления');

return;

}

try {

const res = await fetch(this.host + '/transport/' + this.selectedTransportForDelete.id, {

method: 'DELETE'

});

const data = await res.json();

const index = this.lstTransport.findIndex(transport => transport.id === data.id);

this.lstTransport.splice(index, 1);

this.toggleDeleteTransportForm();

} catch (e) {

console.log('Ошибка: ', e);

}

}

}

};

</script>

* 1. Результат отображения в браузере

|  |
| --- |
|  |
| 1. – Запуск backend/frontend |
|  |
| 1. – Отображение начальной (пустой) БД |
|  |
| 1. – Начальная страница |
|  |
|  |
| 1. – Результат добавления клиента в БД |
|  |
| 1. – Результат удаления клиента из БД |
|  |
| 1. – Результат добавления транспорта в БД |
|  |
| 1. – Результат удаления транспорта из БД |

* 1. Схема Сетей Петри

|  |
| --- |
|  |
| 1. – Сети Петри для «Клиента» |
|  |
| 1. – Сети Петри для «Транспорта» |

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.