**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(МИИТ)**

|  |
| --- |
| **ИНСТИТУТ ТРАНСПОРТНОЙ ТЕХНИКИ И СИСТЕМ УПРАВЛЕНИЯ**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Кафедра «Управление и защита информации»

Лабораторная работа №8

**«Масштабируемое одностраничное веб-приложение на базе фреймворка Vue.js»**

по дисциплине

**«Web программирование»**

|  |  |
| --- | --- |
|  | Выполнил: студент группы ТКИ-541  Куминов В. П.  Проверил: к.т.н., доц.  Сафронов А. И. |

**Москва – 2023 г.**

**1. Цель:**

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

**2. Постановка задачи:**

Создать масштабируемое одностраничное веб-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 (https://elibrary.ru/item.asp?id=46268632 стр. 81-93).

**Требование по функционалу:**

**Хранение данных:**

база данных PostgreSQL.

**Backend:**

На базе фреймворка Express.js (см. Приложение 1).

**Frontend:**

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 ();

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;

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

Предусмотреть текстовую справку-инструкцию по работе с системой.

Рекомендация: обучающимся, прошедшим и / или проходящим курсы «Информационное обеспечение систем управления», «Системы управления базами данных», «Основы построения защищённых баз данных», взять в качестве тематики базы данных выданную в рамках этих курсов.

Подсказка: в методическом пособии рассмотрен пример для «Отдела кадров» в простейшем проявлении – реализовать аналог под выбранную тематику.

**Вариант №9.**Разработка текстовой базы данных и веб-оболочки на тему: «Аптека».

Описание предметной области. База данных создаётся для

информационного обслуживания посетителей аптеки. В аптеку города поступает ассортимент лекарств со склада каждые 7 дней. Аптека предлагает услуги по продаже лекарств и их бронированию. Срок бронирования лекарств – 3 дня. В справочной системе аптеки можно получить информацию о лекарствах, находящихся в самой аптеке: название, форма выпуска, срок годности, аннотация, цена, изготовитель.

**Готовые запросы:**

Выдавать сведения о лекарствах; предоставлять покупателям возможность бронирования лекарств, сроком на 3 дня; выдавать информацию о поступлении лекарства в аптеку, исходя из ассортимента на складе; выдавать информацию о продажах указанного лекарства за неделю (месяц, год); выполнять поиск лекарства по названию, форме выпуска, изготовителю; выдавать список лекарств, применяемых для указанной болезни (легких недугах).

**3. Описание рабочей области:**

**ERD диаграмма БД «Аптека».**

Построим ERD диаграмму БД «Аптека»

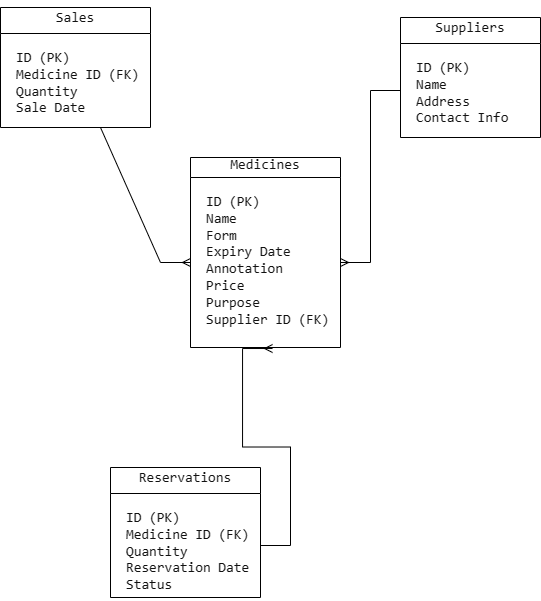


Рисунок 1. ERD диаграмма БД «Аптека»

**Таблица соответствия переменных.**

|  |  |  |
| --- | --- | --- |
| **Переменная в коде** | **Описание в предметной области** | **Тип данных** |
| medicines | Список лекарств в аптеке | Array |
| selectedMedicine | Выбранное лекарство для просмотра деталей | Object |
| searchTerm | Поисковый запрос для фильтрации списка лекарств | String |
| selectedDisease | Выбранное заболевание для фильтрации списка лекарств | String |
| diseases | Список заболеваний для фильтрации списка лекарств | Array |
| quantity | Количество лекарства для бронирования | Number |
| status | Статус бронирования | String |
| reservation\_date | Дата бронирования | Date |
| form | Форма лекарства | String |
| expiry\_date | Срок годности лекарства | Date |
| annotation | Аннотация к лекарству | String |
| price | Цена лекарства | Decimal(10, 2) |
| purpose | Назначение лекарства | String |
| supplier\_id | ID поставщика лекарства | Integer |
| id | ID лекарства | Integer |
| name | Название лекарства | String |

**Архитектура системы.**

**Frontend:**

**Технологии:**

* Vue.js: Фреймворк, используемый для создания пользовательского интерфейса.
* Axios: Библиотека для выполнения HTTP-запросов к серверу из браузера.

**Компоненты:**

* Search.vue: Компонент для поиска и фильтрации лекарств.
* Medicines.vue: Управление лекарствами (просмотр).
* MedicineDetails.vue: Управление деталями лекарства (просмотр, бронирование).
* Sales.vue: Управление продажами (просмотр, добавление, редактирование, удаление).
* Reservations.vue: Управление бронированиями (просмотр, добавление, редактирование, удаление).
* Suppliers.vue: Управление поставщиками (просмотр, добавление, редактирование, удаление).

**Функциональность:**

* CRUD операции: Создание, чтение, обновление и удаление данных.
* Навигация: Переключение между различными компонентами и представлениями.
* Формы: Интерфейсы для ввода данных пользователем.
* Валидация: Проверка корректности введенных данных перед отправкой на сервер.

**Backend:**

**Технологии:**

* Node.js/Express: Серверная платформа и фреймворк для обработки HTTP-запросов.

**API Endpoints:**

* /api/medicines: Эндпоинты для управления лекарствами.
* /api/reservations: Эндпоинты для управления бронированиями.
* /api/sales: Эндпоинты для управления продажами.
* /api/suppliers: Эндпоинты для управления поставщиками.

**Функциональность:**

* Обработка запросов: Получение и обработка запросов от клиента.
* Бизнес-логика: Правила и процедуры, связанные с обработкой данных аптеки.
* Взаимодействие с базой данных: Создание запросов к базе данных для выполнения операций CRUD.

**Database**

**Технологии:**

* SQL база данных: Система управления базами данных, которая хранит данные аптеки.

**Схема данных:**

* Medicines: Таблица или коллекция, содержащая информацию о лекарствах.
* Reservations: Таблица или коллекция, содержащая информацию о бронированиях.
* Sales: Таблица или коллекция, содержащая информацию о продажах.
* Suppliers: Таблица или коллекция, содержащая информацию о поставщиках.

**Функциональность:**

* Хранение данных: Постоянное хранение данных системы.
* Индексация: Ускорение поиска данных.
* Транзакции: Обеспечение целостности данных при выполнении операций.

**Общая связь между компонентами:**

* Пользовательский интерфейс (Frontend) отправляет запросы к серверу через HTTP-запросы, используя Axios.
* Сервер (Backend) принимает запросы, обрабатывает их с помощью бизнес-логики и взаимодействует с базой данных.
* База данных (Database) выполняет операции CRUD на основе запросов от сервера и возвращает результаты.
* Сервер отправляет ответы обратно в пользовательский интерфейс.
* Пользовательский интерфейс обновляется на основе полученных данных.

**Инструкция по развёртке системы:**

**Подготовка окружения:**

* Установка Node.js и npm;
* Установка базы данных PostgreSQL;
* Установка зависимостей npm install express cors pg;
* Создание базы данных с помощью SQL.

**Запуск приложения:**

В директории backend выполним команду node app.js.

**Запуск клиента frontend:**

В директории frontend выполните команду npm run dev для запуска клиентской части приложения.

**4. Ход работы:**

**Код создания таблиц (SQL):**

CREATE TABLE medicines (

id SERIAL PRIMARY KEY,

name VARCHAR(255),

form VARCHAR(255),

expiry\_date DATE,

annotation TEXT,

price DECIMAL(10, 2),

purpose VARCHAR(255),

supplier\_id INT

);

CREATE TABLE suppliers (

id SERIAL PRIMARY KEY,

name VARCHAR(255),

address VARCHAR(255),

contact\_info VARCHAR(255)

);

CREATE TABLE sales (

id SERIAL PRIMARY KEY,

medicine\_id INT,

quantity INT,

sale\_date DATE

);

CREATE TABLE reservations (

id SERIAL PRIMARY KEY,

medicine\_id INT,

quantity INT,

reservation\_date DATE,

status VARCHAR(255)

);

**Express.js (app.js)**

const express = require('express');

const cors = require('cors');

const { Pool } = require('pg');

const pool = new Pool({

user: 'postgres',

host: 'localhost',

database: 'pharmacy',

password: '1',

port: 5432,

});

const app = express();

app.use(cors());

app.use(express.json());

app.get('/api/medicines', async (req, res) => {

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

res.json(result.rows);

});

app.get('/api/suppliers', async (req, res) => {

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

res.json(result.rows);

});

app.get('/api/sales', async (req, res) => {

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

res.json(result.rows);

});

app.get('/api/reservations', async (req, res) => {

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

res.json(result.rows);

});

// Предоставление покупателям возможности бронирования лекарств, сроком на 3 дня

app.post('/api/reservations', async (req, res) => {

const { medicine\_id, quantity } = req.body;

const reservation\_date = new Date();

await pool.query('INSERT INTO reservations (medicine\_id, quantity, reservation\_date, status) VALUES ($1, $2, $3, $4)', [medicine\_id, quantity, reservation\_date, 'Забронировано']);

res.json({ message: 'Бронирование успешно создано' });

});

// Выдача информации о поступлении лекарства в аптеку, исходя из ассортимента на складе

app.get('/api/medicines/:id/sales', async (req, res) => {

const { id } = req.params;

const result = await pool.query('SELECT \* FROM sales WHERE medicine\_id = $1', [id]);

res.json(result.rows);

});

// Выдача информации о продажах указанного лекарства за неделю (месяц, год)

app.get('/api/medicines/:id/sales/:period', async (req, res) => {

const { id, period } = req.params;

const now = new Date();

let startDate;

if (period === 'week') {

startDate = new Date(now.getFullYear(), now.getMonth(), now.getDate() - 7);

} else if (period === 'month') {

startDate = new Date(now.getFullYear(), now.getMonth() - 1, now.getDate());

} else if (period === 'year') {

startDate = new Date(now.getFullYear() - 1, now.getMonth(), now.getDate());

}

const result = await pool.query('SELECT \* FROM sales WHERE medicine\_id = $1 AND sale\_date >= $2', [id, startDate]);

res.json(result.rows);

});

app.listen(3000, () => console.log('Server running on port 3000'));

**main.js**  
import './assets/main.css'

import { createApp } from 'vue'

import App from './App.vue'

createApp(App).mount('#app')

**App.vue**

<template>

<div id="app">

<Medicines @select="selectMedicine" />

<MedicineDetails :selectedMedicine="selectedMedicine" />

<Sales v-if="selectedMedicine" :medicineId="selectedMedicine.id" />

</div>

</template>

<script>

import Medicines from './components/Medicines.vue';

import MedicineDetails from './components/MedicineDetails.vue';

import Sales from './components/Sales.vue';

export default {

name: 'App',

components: {

Medicines,

MedicineDetails,

Sales,

},

data() {

return {

selectedMedicine: null,

};

},

methods: {

selectMedicine(medicine) {

this.selectedMedicine = medicine;

},

},

};

</script>

**Medicines.vue**

<template>

<div>

<Search @search="searchMedicines" @filter="filterMedicines" />

<table border="1">

<thead>

<tr>

<th>Название</th>

<th>Форма</th>

<th>Срок годности</th>

<th>Аннотация</th>

<th>Цена</th>

<th>Назначение</th>

<th>ID поставщика</th>

</tr>

</thead>

<tbody>

<tr v-for="medicine in medicines" :key="medicine.id" @click="selectMedicine(medicine)">

<td>{{ medicine.name }}</td>

<td>{{ medicine.form }}</td>

<td>{{ medicine.expiry\_date }}</td>

<td>{{ medicine.annotation }}</td>

<td>{{ medicine.price }}</td>

<td>{{ medicine.purpose }}</td>

<td>{{ medicine.supplier\_id }}</td>

</tr>

</tbody>

</table>

</div>

</template>

<script>

import axios from 'axios';

import Search from './Search.vue';

export default {

components: {

Search,

},

data() {

return {

medicines: [],

allMedicines: [],

};

},

methods: {

selectMedicine(medicine) {

this.$emit('select', medicine);

},

searchMedicines(term) {

this.medicines = this.allMedicines.filter((medicine) =>

medicine.name.toLowerCase().includes(term.toLowerCase())

);

},

filterMedicines(purpose) {

if (purpose) {

this.medicines = this.allMedicines.filter((medicine) =>

medicine.purpose.toLowerCase() === purpose.toLowerCase()

);

} else {

this.medicines = this.allMedicines;

}

},

},

async created() {

const response = await axios.get('http://localhost:3000/api/medicines');

this.allMedicines = response.data;

this.medicines = this.allMedicines;

},

};

</script>

<style scoped>

table {

width: 100%;

text-align: center;

}

tr {

cursor: pointer;

}

</style>

**MedicinesDetails.vue**

<template>

<div v-if="selectedMedicine" class="medicine-details">

<h1>{{ selectedMedicine.name }}</h1>

<p>Форма: {{ selectedMedicine.form }}</p>

<p>Срок годности: {{ selectedMedicine.expiry\_date }}</p>

<p>Аннотация: {{ selectedMedicine.annotation }}</p>

<p>Цена: {{ selectedMedicine.price }}</p>

<p>Назначение: {{ selectedMedicine.purpose }}</p>

<p>ID поставщика: {{ selectedMedicine.supplier\_id }}</p>

<button @click="reserveMedicine">Забронировать</button>

</div>

</template>

<script>

import axios from 'axios';

export default {

props: ['selectedMedicine'],

methods: {

async reserveMedicine() {

const quantity = prompt('Введите количество для бронирования:');

if (quantity) {

await axios.post('http://localhost:3000/api/reservations', { medicine\_id: this.selectedMedicine.id, quantity });

alert('Бронирование успешно создано');

}

},

},

};

</script>

<style scoped>

.medicine-details {

margin-top: 20px;

}

</style>

**Reservations.vue**

<template>

<div>

<h1>Список бронирований</h1>

<ul>

<li v-for="reservation in reservations" :key="reservation.id">

{{ reservation.medicine\_id }} - {{ reservation.quantity }} - {{ reservation.reservation\_date }} - {{ reservation.status }}

</li>

</ul>

</div>

</template>

<script>

import axios from 'axios';

export default {

data() {

return {

reservations: [],

};

},

async created() {

const response = await axios.get('http://localhost:3000/api/reservations');

this.reservations = response.data;

},

};

</script>

**Sales.vue**

<template>

<div>

<h1>Продажи</h1>

<select v-model="period" @change="getSales">

<option value="">Выберите период</option>

<option value="week">Неделя</option>

<option value="month">Месяц</option>

<option value="year">Год</option>

</select>

<table v-if="sales.length">

<thead>

<tr>

<th>ID продажи</th>

<th>ID лекарства</th>

<th>Количество</th>

<th>Дата продажи</th>

</tr>

</thead>

<tbody>

<tr v-for="sale in sales" :key="sale.id">

<td>{{ sale.id }}</td>

<td>{{ sale.medicine\_id }}</td>

<td>{{ sale.quantity }}</td>

<td>{{ sale.sale\_date }}</td>

</tr>

</tbody>

</table>

</div>

</template>

<script>

import axios from 'axios';

export default {

props: ['medicineId'],

data() {

return {

period: '',

sales: [],

};

},

methods: {

async getSales() {

if (this.period) {

const response = await axios.get(`http://localhost:3000/api/medicines/${this.medicineId}/sales/${this.period}`);

this.sales = response.data;

}

},

},

};

</script>

**Search.vue**

<template>

<div>

<input type="text" v-model="searchTerm" placeholder="Поиск лекарства..." />

<select v-model="selectedDisease">

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

<option value="">Все лекарства</option>

<option v-for="disease in diseases" :key="disease">{{ disease }}</option>

</select>

</div>

</template>

<script>

export default {

data() {

return {

searchTerm: '',

selectedDisease: '',

diseases: ['Головная боль', 'Воспаление', 'Гипертония', 'Инфекция'],

};

},

watch: {

searchTerm(newTerm) {

this.$emit('search', newTerm);

},

selectedDisease(newDisease) {

this.$emit('filter', newDisease);

},

},

};

</script>

**Suppliers.vue**

<template>

<div>

<h1>Список поставщиков</h1>

<ul>

<li v-for="supplier in suppliers" :key="supplier.id">

{{ supplier.name }} - {{ supplier.address }} - {{ supplier.contact\_info }}

</li>

</ul>

</div>

</template>

<script>

import axios from 'axios';

export default {

data() {

return {

suppliers: [],

};

},

async created() {

const response = await axios.get('http://localhost:3000/api/suppliers');

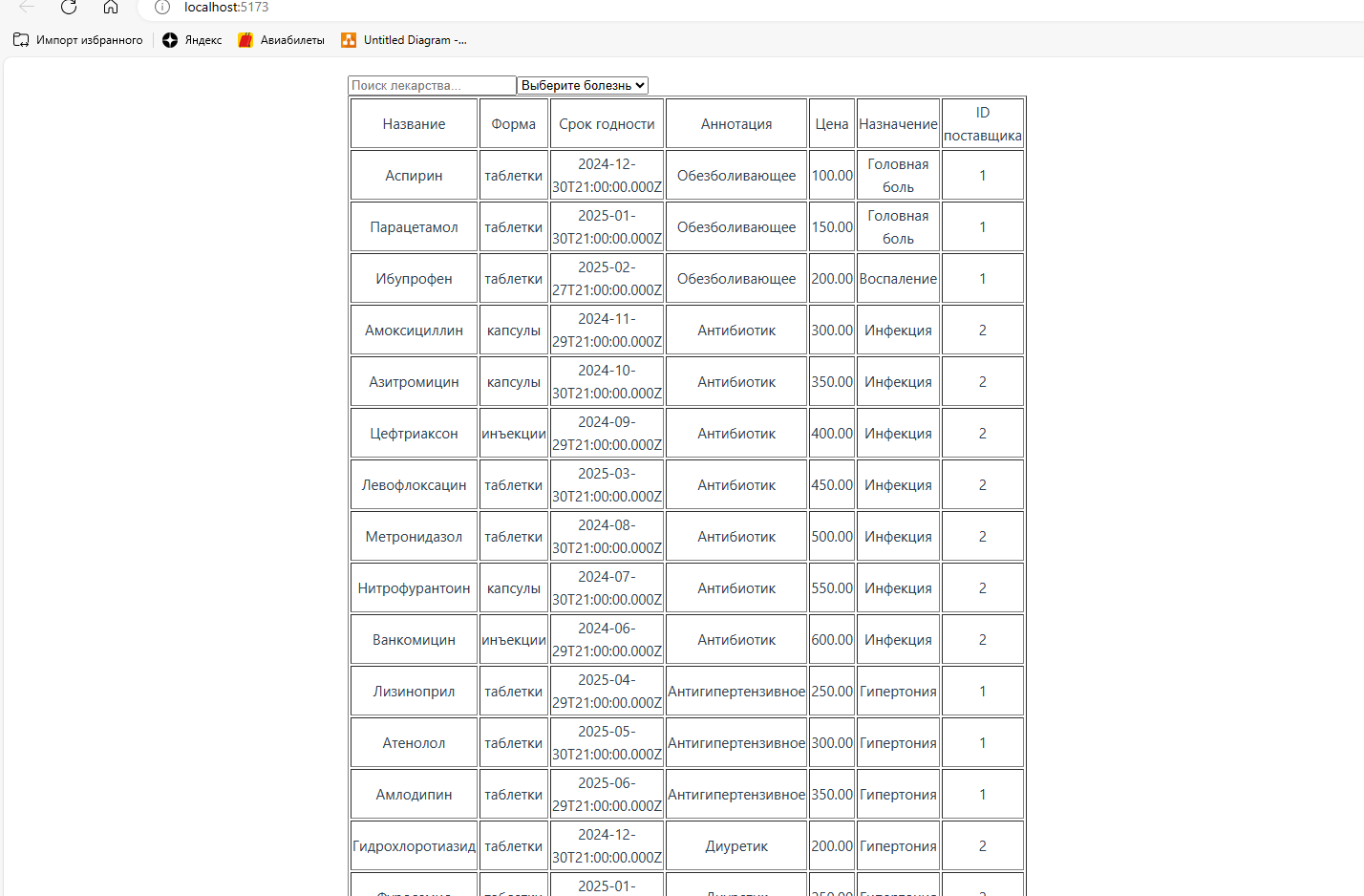
this.suppliers = response.data;

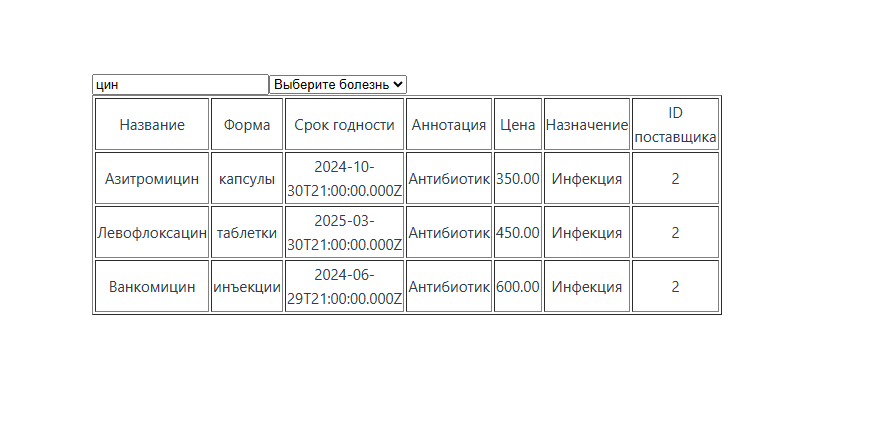
},

};

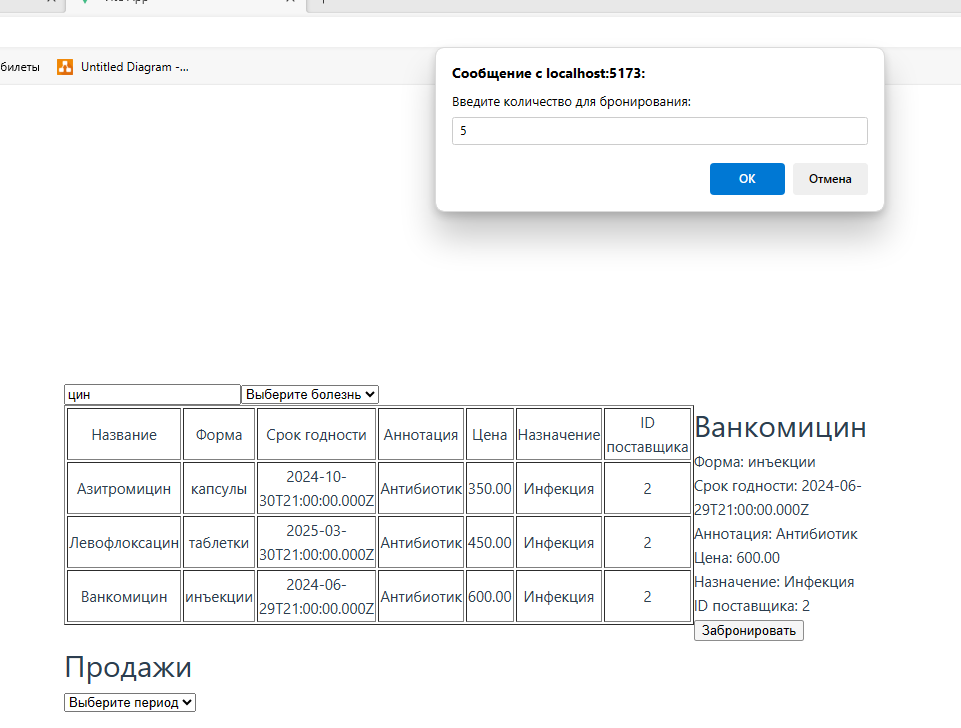
</script>

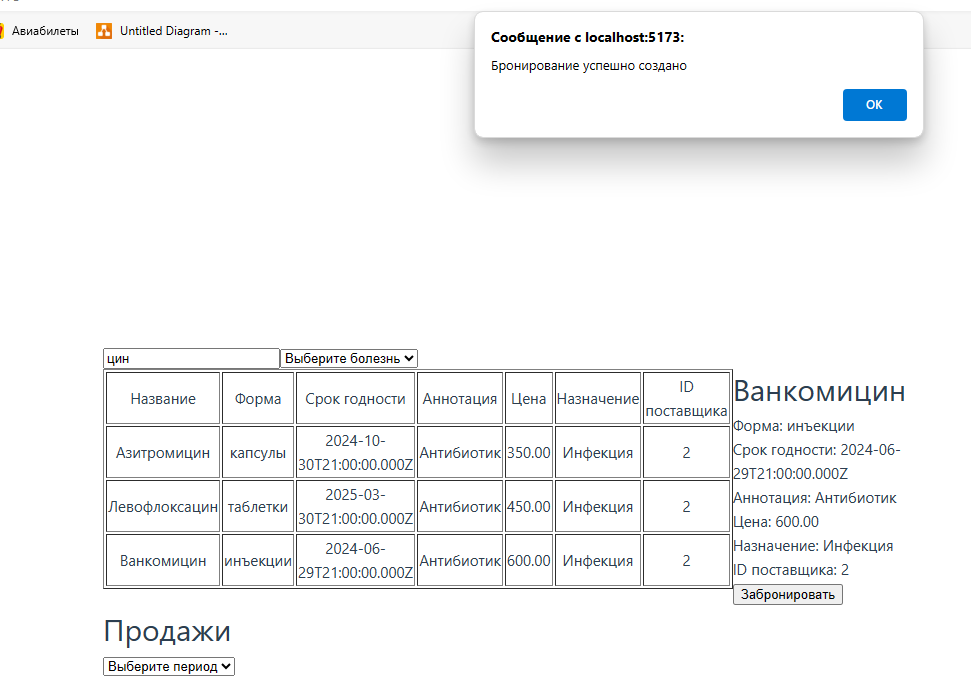
**Отображение в браузере**













**4. Вывод**