**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.**

Оглавление

[**1.** **Цель:** 3](#_Toc150883757)

[**2.** **Постановка задачи:** 3](#_Toc150883758)

[2.1 Требование по функционалу: 3](#_Toc150883759)

[2.2 Реализация методологии CRUD: 3](#_Toc150883760)

[2.3 Вариант №9. 4](#_Toc150883761)

[**3.** **Описание рабочей области:** 5](#_Toc150883762)

[3.1 ERD диаграмма БД «Аптека». 5](#_Toc150883763)

[3.2 Таблица соответствия переменных. 5](#_Toc150883764)

[3.3 Архитектура системы. 6](#_Toc150883765)

[3.4 Инструкция по развёртке системы: 8](#_Toc150883766)

[**4.** **Ход работы:** 8](#_Toc150883767)

[4.1 Код создания таблиц (SQL): 8](#_Toc150883768)

[4.2 Express.js (app.js) 9](#_Toc150883769)

[4.3 main.js 14](#_Toc150883770)

[4.4 router.js 14](#_Toc150883771)

[4.5 App.vue 15](#_Toc150883772)

[4.6 Medicines.vue 16](#_Toc150883773)

[4.7 Reservations.vue 24](#_Toc150883774)

[4.8 Sales.vue 27](#_Toc150883775)

[4.9 Search.vue 30](#_Toc150883776)

[4.10 Suppliers.vue 31](#_Toc150883777)

[**5. Сеть Петри 35**](#_Toc150883778)

[**6. Отображение в браузере 35**](#_Toc150883779)

[**7. Вывод 38**](#_Toc150883780)

# 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

# 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.

* 1. 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).

# Описание рабочей области:

## ERD диаграмма БД «Аптека».

Построим ERD диаграмму БД «Аптека»

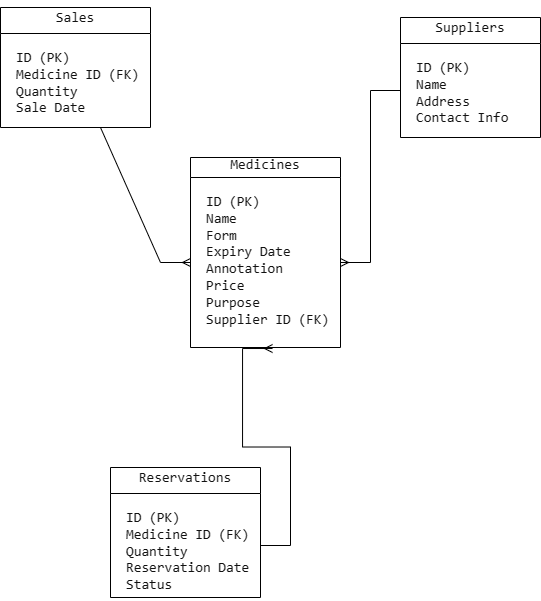


Рисунок . ERD диаграмма БД «Аптека»

## 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, редактирование, добавление, удаление, бронирование).
* Sales.vue: Управление продажами (просмотр, добавление, редактирование, удаление).
* Reservations.vue: Управление бронированиями (просмотр, редактирование, удаление).
* Suppliers.vue: Управление поставщиками (просмотр, добавление, редактирование, удаление).
* Home.vue Отображение приветственной страницы (просмотр)
* Above.vue Отображение приветственной страницы (просмотр)

**Функциональность:**

* CRUD операции: Создание, чтение, обновление и удаление данных.
* Навигация: Переключение между различными компонентами и представлениями.
* Формы: Интерфейсы для ввода данных пользователем.

**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.

# 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());

// Medicines

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

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

res.json(result.rows);

});

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

const { name, form, expiry\_date, annotation, price, purpose, supplier\_id } = req.body;

const result = await pool.query('INSERT INTO medicines (name, form, expiry\_date, annotation, price, purpose, supplier\_id) VALUES ($1, $2, $3, $4, $5, $6, $7) RETURNING \*', [name, form, expiry\_date, annotation, price, purpose, supplier\_id]);

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

});

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

const { id } = req.params;

const { name, form, expiry\_date, annotation, price, purpose, supplier\_id } = req.body;

await pool.query('UPDATE medicines SET name = $1, form = $2, expiry\_date = $3, annotation = $4, price = $5, purpose = $6, supplier\_id = $7 WHERE id = $8', [name, form, expiry\_date, annotation, price, purpose, supplier\_id, id]);

res.json({ message: 'Лекарство успешно обновлено' });

});

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

const { id } = req.params;

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

res.json({ message: 'Лекарство успешно удалено' });

});

// Suppliers

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

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

res.json(result.rows);

});

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

const { name, address, contact\_info } = req.body;

const result = await pool.query('INSERT INTO suppliers (name, address, contact\_info) VALUES ($1, $2, $3) RETURNING \*', [name, address, contact\_info]);

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

});

app.put('/api/suppliers/:id', async (req, res) => {

const { id } = req.params;

const { name, address, contact\_info } = req.body;

await pool.query('UPDATE suppliers SET name = $1, address = $2, contact\_info = $3 WHERE id = $4', [name, address, contact\_info, id]);

res.json({ message: 'Поставщик успешно обновлен' });

});

app.delete('/api/suppliers/:id', async (req, res) => {

const { id } = req.params;

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

res.json({ message: 'Поставщик успешно удален' });

});

// Sales

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

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

res.json(result.rows);

});

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

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

const result = await pool.query('INSERT INTO sales (medicine\_id, quantity, sale\_date) VALUES ($1, $2, $3) RETURNING \*', [medicine\_id, quantity, sale\_date]);

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

});

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

const { id } = req.params;

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

await pool.query('UPDATE sales SET medicine\_id = $1, quantity = $2, sale\_date = $3 WHERE id = $4', [medicine\_id, quantity, sale\_date, id]);

res.json({ message: 'Продажа успешно обновлена' });

});

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

const { id } = req.params;

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

res.json({ message: 'Продажа успешно удалена' });

});

// Reservations

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

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

res.json(result.rows);

});

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' });

});

app.put('/api/reservations/:id', async (req, res) => {

const { id } = req.params;

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

await pool.query('UPDATE reservations SET medicine\_id = $1, quantity = $2, reservation\_date = $3, status = $4 WHERE id = $5', [medicine\_id, quantity, reservation\_date, status, id]);

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

});

app.delete('/api/reservations/:id', async (req, res) => {

const { id } = req.params;

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

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

});

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

* 1. main.js

import { createApp } from 'vue';

import App from './App.vue';

import router from './router';

createApp(App).use(router).mount('#app');

## router.js

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

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

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

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

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

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

const routes = [

{ path: '/', component: Home },

{ path: '/about', component: About },

{ path: '/medicines', component: Medicines },

{ path: '/suppliers', component: Suppliers },

{ path: '/reservations', component: Reservations },

];

const router = createRouter({

history: createWebHistory(),

routes,

});

export default router;

## App.vue

<template>

<div id="app">

<nav>

<router-link to="/">Главная</router-link>

<router-link to="/about">О нас</router-link>

<router-link to="/medicines">Лекарства</router-link>

<router-link to="/suppliers">Производители</router-link>

<router-link to="/reservations">Моё бронирование</router-link>

</nav>

<router-view />

</div>

</template>

<script>

import { onMounted } from 'vue';

import { useStore } from 'vuex';

export default {

name: 'App',

setup() {

const store = useStore();

onMounted(() => {

store.dispatch('fetchMedicines');

store.dispatch('fetchSuppliers');

store.dispatch('fetchReservations');

});

},

};

</script>

<style>

nav {

display: flex;

justify-content: space-around;

padding: 1em;

background-color: #f8f9fa;

margin-bottom: 1em;

}

</style>

## Medicines.vue

<template>

<div>

<h1 style="text-align: center;">Лекарства</h1>

<Search v-if="!selectedMedicine" @search="searchMedicines" @filter="filterMedicines" />

<div v-if="!selectedMedicine">

<table>

<thead>

<tr>

<th>ID</th>

<th>Название</th>

<th>Форма</th>

<th>Срок годности</th>

<th>Аннотация</th>

<th>Цена</th>

<th>Назначение</th>

<th>ID поставщика</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

<tr v-for="medicine in filteredMedicines" :key="medicine.id" @click="selectMedicine(medicine)" class="clickable-row">

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

<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>

<td>

<button @click.stop="editMedicine(medicine)">Редактировать</button>

<button @click.stop="deleteMedicine(medicine.id)">Удалить</button>

</td>

</tr>

</tbody>

</table>

<button @click="addMedicine">Добавить лекарство</button>

<div v-if="medicineToEdit" class="modal">

<h2>Редактировать лекарство</h2>

<label>ID: <input v-model.number="medicineToEdit.id" /></label>

<label>Название: <input v-model="medicineToEdit.name" /></label>

<label>Форма: <input v-model="medicineToEdit.form" /></label>

<label>Срок годности: <input v-model="medicineToEdit.expiry\_date" /></label>

<label>Аннотация: <input v-model="medicineToEdit.annotation" /></label>

<label>Цена: <input v-model.number="medicineToEdit.price" /></label>

<label>Назначение: <input v-model="medicineToEdit.purpose" /></label>

<label>ID поставщика: <input v-model.number="medicineToEdit.supplier\_id" /></label>

<button @click="updateMedicine">Сохранить</button>

<button @click="medicineToEdit = null">Отмена</button>

</div>

<div v-if="addingNewMedicine" class="modal">

<h2>Добавить лекарство</h2>

<label>ID: <input v-model.number="newMedicine.id" /></label>

<label>Название: <input v-model="newMedicine.name" /></label>

<label>Форма: <input v-model="newMedicine.form" /></label>

<label>Срок годности: <input v-model="newMedicine.expiry\_date" /></label>

<label>Аннотация: <input v-model="newMedicine.annotation" /></label>

<label>Цена: <input v-model.number="newMedicine.price" /></label>

<label>Назначение: <input v-model="newMedicine.purpose" /></label>

<label>ID поставщика: <input v-model.number="newMedicine.supplier\_id" /></label>

<button @click="createMedicine">Добавить</button>

<button @click="addingNewMedicine = false">Отмена</button>

</div>

</div>

<div v-else>

<h2 style="text-align: center;">{{ selectedMedicine.name }}</h2>

<table class="info-table">

<tr><th>Форма</th><td>{{ selectedMedicine.form }}</td></tr>

<tr><th>Срок годности</th><td>{{ selectedMedicine.expiry\_date }}</td></tr>

<tr><th>Аннотация</th><td>{{ selectedMedicine.annotation }}</td></tr>

<tr><th>Цена</th><td>{{ selectedMedicine.price }}</td></tr>

<tr><th>Назначение</th><td>{{ selectedMedicine.purpose }}</td></tr>

</table>

<h3 style="text-align: center;">Поставщик</h3>

<table class="info-table" v-if="getSupplier(selectedMedicine.supplier\_id)">

<tr><th>Имя</th><td>{{ getSupplier(selectedMedicine.supplier\_id).name }}</td></tr>

<tr><th>Адрес</th><td>{{ getSupplier(selectedMedicine.supplier\_id).address }}</td></tr>

<tr><th>Контактная информация</th><td>{{ getSupplier(selectedMedicine.supplier\_id).contact\_info }}</td></tr>

</table>

<h3 style="text-align: center;">Продажи</h3>

<table class="info-table">

<thead>

<tr>

<th>Количество</th>

<th>Дата продажи</th>

</tr>

</thead>

<tbody>

<tr v-for="sale in getSales(selectedMedicine.id)" :key="sale.id">

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

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

</tr>

</tbody>

</table>

<div style="text-align: center; margin-top: 20px;">

<button @click="selectedMedicine = null">Вернуться к списку</button>

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

</div>

</div>

</div>

</template>

<script>

import axios from 'axios';

import Search from './Search.vue';

export default {

components: {

Search,

},

data() {

return {

medicines: [],

newMedicine: {},

medicineToEdit: null,

addingNewMedicine: false,

selectedMedicine: null,

filteredMedicines: [],

suppliers: [],

sales: [],

};

},

methods: {

async searchMedicines(query) {

this.filteredMedicines = this.medicines.filter(medicine => medicine.name.toLowerCase().includes(query.toLowerCase()));

},

filterMedicines(disease) {

if (disease) {

this.filteredMedicines = this.medicines.filter(medicine => medicine.purpose === disease);

} else {

this.filteredMedicines = this.medicines;

}

},

selectMedicine(medicine) {

this.selectedMedicine = medicine;

},

editMedicine(medicine) {

this.medicineToEdit = { ...medicine };

},

async updateMedicine() {

if (this.medicineToEdit) {

await axios.put(`http://localhost:3000/api/medicines/${this.medicineToEdit.id}`, this.medicineToEdit);

const index = this.medicines.findIndex(m => m.id === this.medicineToEdit.id);

this.medicines.splice(index, 1, this.medicineToEdit);

this.medicineToEdit = null;

}

},

async deleteMedicine(id) {

await axios.delete(`http://localhost:3000/api/medicines/${id}`);

this.medicines = this.medicines.filter(medicine => medicine.id !== id);

},

addMedicine() {

this.addingNewMedicine = true;

},

async createMedicine() {

const response = await axios.post('http://localhost:3000/api/medicines', this.newMedicine);

this.medicines.push(response.data);

this.newMedicine = {};

this.addingNewMedicine = false;

},

async reserveMedicine() {

await axios.post('http://localhost:3000/api/reservations', {

medicine\_id: this.selectedMedicine.id,

quantity: 1,

});

alert('Лекарство успешно забронировано!');

},

getSupplier(id) {

return this.suppliers.find(supplier => supplier.id === id);

},

getSales(id) {

return this.sales.filter(sale => sale.medicine\_id === id);

},

},

async created() {

const [medicinesResponse, suppliersResponse, salesResponse] = await Promise.all([

axios.get('http://localhost:3000/api/medicines'),

axios.get('http://localhost:3000/api/suppliers'),

axios.get('http://localhost:3000/api/sales'),

]);

this.medicines = medicinesResponse.data;

this.suppliers = suppliersResponse.data;

this.sales = salesResponse.data;

this.filteredMedicines = this.medicines;

},

};

</script>

<style scoped>

table {

width: 100%;

border-collapse: collapse;

margin-bottom: 1em;

}

th, td {

border: 1px solid #ddd;

padding: 8px;

text-align: left;

}

tr:nth-child(even) {

background-color: #f2f2f2;

}

th {

padding-top: 12px;

padding-bottom: 12px;

text-align: left;

background-color: #4CAF50;

color: white;

}

.clickable-row {

cursor: pointer;

}

.modal {

position: fixed;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

background: #fff;

padding: 20px;

border: 1px solid #ccc;

z-index: 1000;

}

.info-table {

width: 50%;

margin: 0 auto;

}

.info-table th,

.info-table td {

width: 50%;

}

</style>

## Reservations.vue

<template>

<div>

<h1 style="text-align: center;">Моё бронирование</h1>

<table>

<thead>

<tr>

<th>Название лекарства</th>

<th>Количество</th>

<th>Дата бронирования</th>

<th>Статус</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

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

<td>{{ getMedicineName(reservation.medicine\_id) }}</td>

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

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

<td>{{ reservation.status }}</td>

<td>

<button @click="deleteReservation(reservation.id)">Отменить бронирование</button>

</td>

</tr>

</tbody>

</table>

</div>

</template>

<script>

import axios from 'axios';

export default {

data() {

return {

reservations: [],

medicines: [],

};

},

methods: {

getMedicineName(id) {

const medicine = this.medicines.find(m => m.id === id);

return medicine ? medicine.name : 'Неизвестное лекарство';

},

async deleteReservation(id) {

await axios.delete(`http://localhost:3000/api/reservations/${id}`);

this.reservations = this.reservations.filter(reservation => reservation.id !== id);

},

},

async created() {

const [reservationsResponse, medicinesResponse] = await Promise.all([

axios.get('http://localhost:3000/api/reservations'),

axios.get('http://localhost:3000/api/medicines'),

]);

this.reservations = reservationsResponse.data;

this.medicines = medicinesResponse.data;

},

};

</script>

<style scoped>

table {

width: 100%;

border-collapse: collapse;

margin-bottom: 1em;

}

th, td {

border: 1px solid #ddd;

padding: 8px;

text-align: left;

}

tr:nth-child(even) {

background-color: #f2f2f2;

}

th {

padding-top: 12px;

padding-bottom: 12px;

text-align: left;

background-color: #4CAF50;

color: white;

}

</style>

## Sales.vue

<template>

<div>

<h1>Продажи</h1>

<ul>

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

{{ sale.medicine\_id }} - {{ sale.quantity }} - {{ sale.sale\_date }}

<button @click="editSale(sale)">Редактировать</button>

<button @click="deleteSale(sale.id)">Удалить</button>

</li>

</ul>

<button @click="addSale">Добавить продажу</button>

<div v-if="saleToEdit" class="modal">

<h2>Редактировать продажу</h2>

<label>ID: <input v-model.number="saleToEdit.id" /></label>

<label>ID лекарства: <input v-model="saleToEdit.medicine\_id" /></label>

<label>Количество: <input v-model="saleToEdit.quantity" /></label>

<label>Дата продажи: <input v-model="saleToEdit.sale\_date" /></label>

<button @click="updateSale">Сохранить</button>

<button @click="saleToEdit = null">Отмена</button>

</div>

<div v-if="addingNewSale" class="modal">

<h2>Добавить продажу</h2>

<label>ID: <input v-model.number="newSale.id" /></label>

<label>ID лекарства: <input v-model="newSale.medicine\_id" /></label>

<label>Количество: <input v-model="newSale.quantity" /></label>

<label>Дата продажи: <input v-model="newSale.sale\_date" /></label>

<button @click="createSale">Добавить</button>

<button @click="addingNewSale = false">Отмена</button>

</div>

</div>

</template>

<script>

import axios from 'axios';

export default {

data() {

return {

sales: [],

newSale: {},

saleToEdit: null,

addingNewSale: false,

};

},

methods: {

editSale(sale) {

this.saleToEdit = { ...sale };

},

async updateSale() {

if (this.saleToEdit) {

await axios.put(`http://localhost:3000/api/sales/${this.saleToEdit.id}`, this.saleToEdit);

const index = this.sales.findIndex(s => s.id === this.saleToEdit.id);

this.sales.splice(index, 1, this.saleToEdit);

this.saleToEdit = null;

}

},

async deleteSale(id) {

await axios.delete(`http://localhost:3000/api/sales/${id}`);

this.sales = this.sales.filter(sale => sale.id !== id);

},

addSale() {

this.addingNewSale = true;

},

async createSale() {

const response = await axios.post('http://localhost:3000/api/sales', this.newSale);

this.sales.push(response.data);

this.newSale = {};

this.addingNewSale = false;

},

},

async created() {

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

this.sales = response.data;

},

};

</script>

<style scoped>

.modal {

position: fixed;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

background: #fff;

padding: 20px;

border: 1px solid #ccc;

z-index: 1000;

}

</style>

## Search.vue

<template>

<div>

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

<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 style="text-align: center;">Поставщики</h1>

<table>

<thead>

<tr>

<th>ID</th>

<th>Название</th>

<th>Адрес</th>

<th>Контактная информация</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

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

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

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

<td>{{ supplier.address }}</td>

<td>{{ supplier.contact\_info }}</td>

<td>

<button @click="editSupplier(supplier)">Редактировать</button>

<button @click="deleteSupplier(supplier.id)">Удалить</button>

</td>

</tr>

</tbody>

</table>

<button @click="addSupplier">Добавить поставщика</button>

<div v-if="supplierToEdit" class="modal">

<h2>Редактировать поставщика</h2>

<label>ID: <input v-model.number="supplierToEdit.id" /></label>

<label>Название: <input v-model="supplierToEdit.name" /></label>

<label>Адрес: <input v-model="supplierToEdit.address" /></label>

<label>Контактная информация: <input v-model="supplierToEdit.contact\_info" /></label>

<button @click="updateSupplier">Сохранить</button>

<button @click="supplierToEdit = null">Отмена</button>

</div>

<div v-if="addingNewSupplier" class="modal">

<h2>Добавить поставщика</h2>

<label>ID: <input v-model.number="newSupplier.id" /></label>

<label>Название: <input v-model="newSupplier.name" /></label>

<label>Адрес: <input v-model="newSupplier.address" /></label>

<label>Контактная информация: <input v-model="newSupplier.contact\_info" /></label>

<button @click="createSupplier">Добавить</button>

<button @click="addingNewSupplier = false">Отмена</button>

</div>

</div>

</template>

<script>

import axios from 'axios';

export default {

data() {

return {

suppliers: [],

newSupplier: {},

supplierToEdit: null,

addingNewSupplier: false,

};

},

methods: {

editSupplier(supplier) {

this.supplierToEdit = { ...supplier };

},

async updateSupplier() {

if (this.supplierToEdit) {

await axios.put(`http://localhost:3000/api/suppliers/${this.supplierToEdit.id}`, this.supplierToEdit);

const index = this.suppliers.findIndex(s => s.id === this.supplierToEdit.id);

this.suppliers.splice(index, 1, this.supplierToEdit);

this.supplierToEdit = null;

}

},

async deleteSupplier(id) {

await axios.delete(`http://localhost:3000/api/suppliers/${id}`);

this.suppliers = this.suppliers.filter(supplier => supplier.id !== id);

},

addSupplier() {

this.addingNewSupplier = true;

},

async createSupplier() {

const response = await axios.post('http://localhost:3000/api/suppliers', this.newSupplier);

this.suppliers.push(response.data);

this.newSupplier = {};

this.addingNewSupplier = false;

},

},

async created() {

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

this.suppliers = response.data;

},

};

</script>

<style scoped>

table {

width: 100%;

border-collapse: collapse;

margin-bottom: 1em;

}

th, td {

border: 1px solid #ddd;

padding: 8px;

text-align: left;

}

tr:nth-child(even) {

background-color: #f2f2f2;

}

th {

padding-top: 12px;

padding-bottom: 12px;

text-align: left;

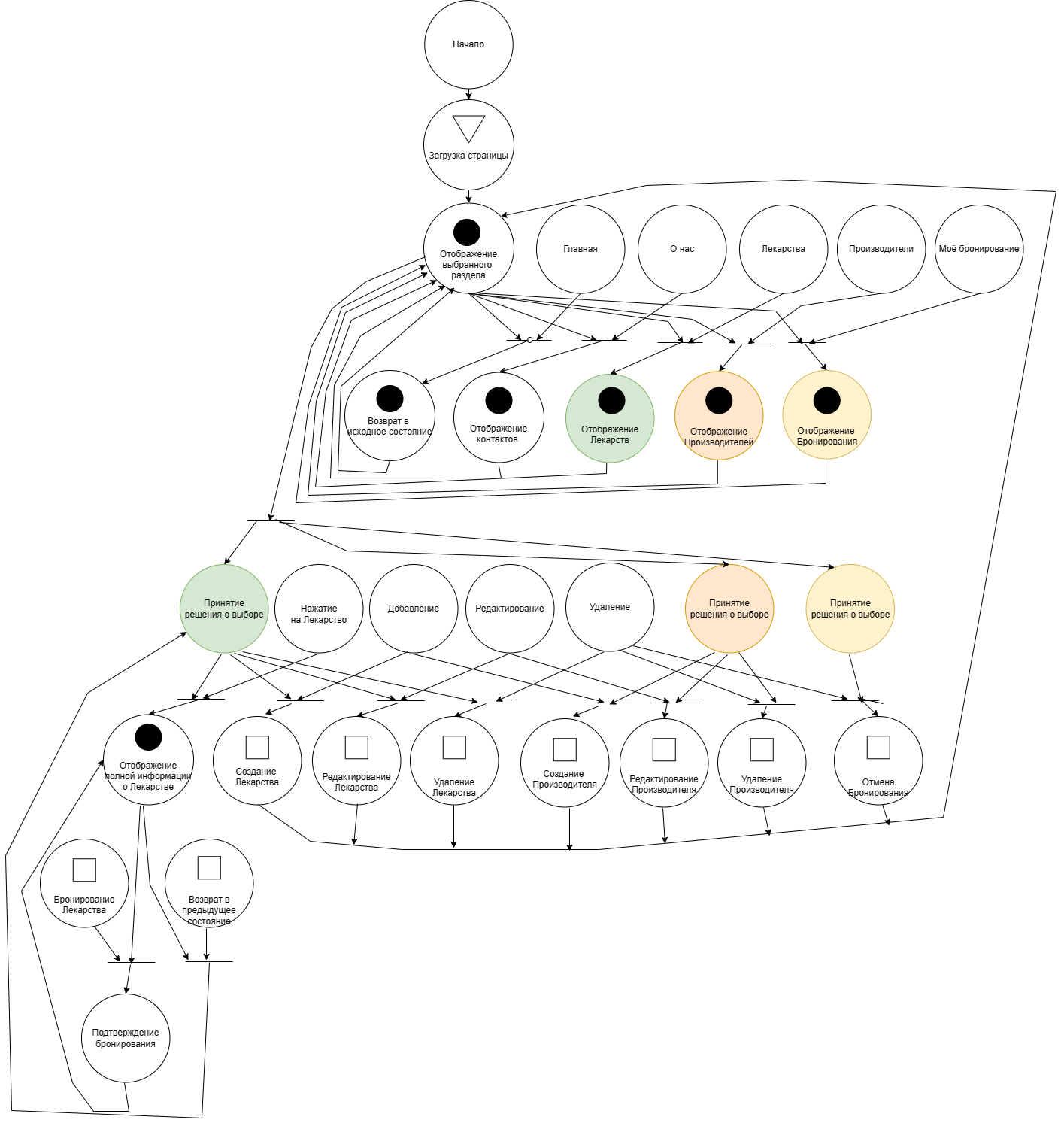
background-color: #4CAF50;

color: white;

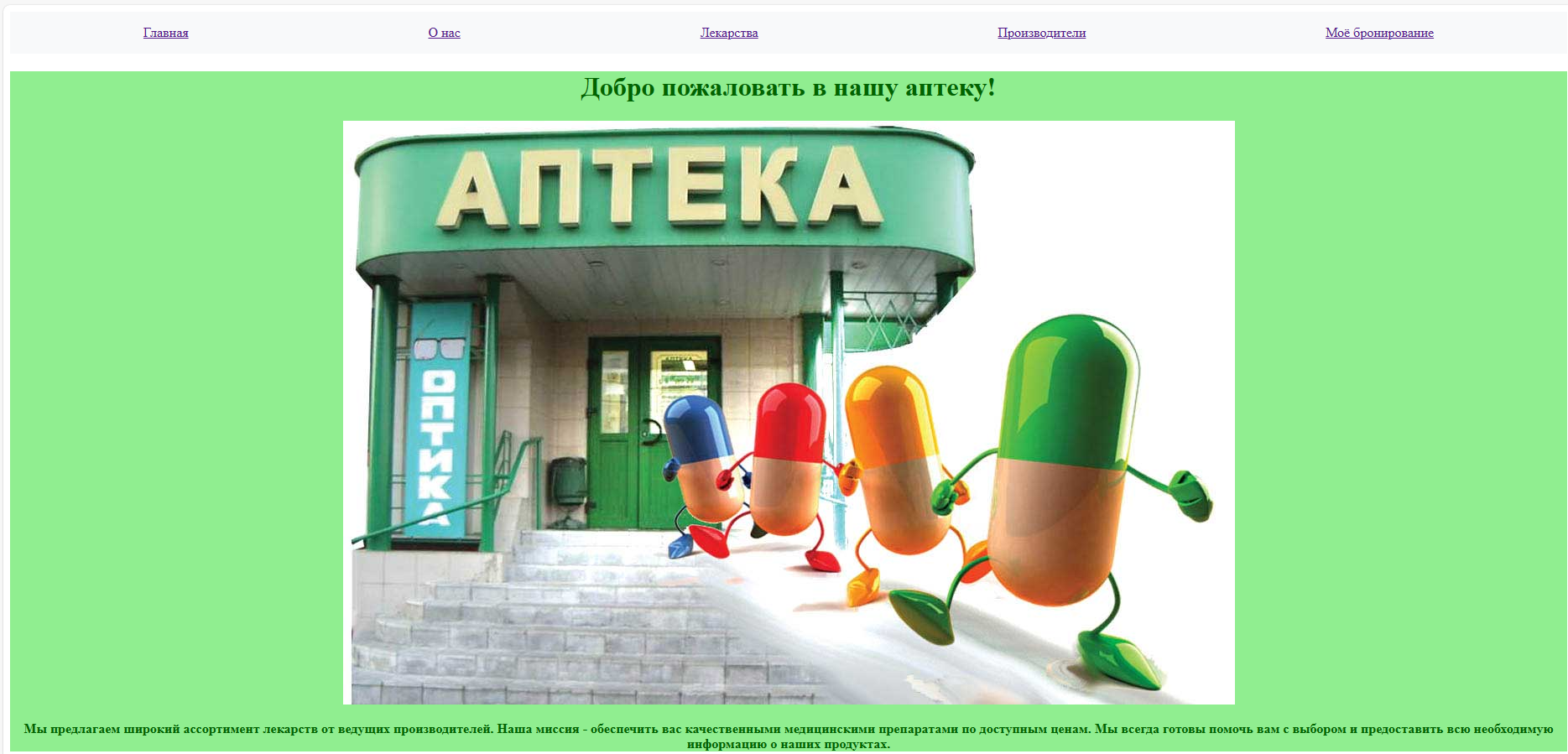
}

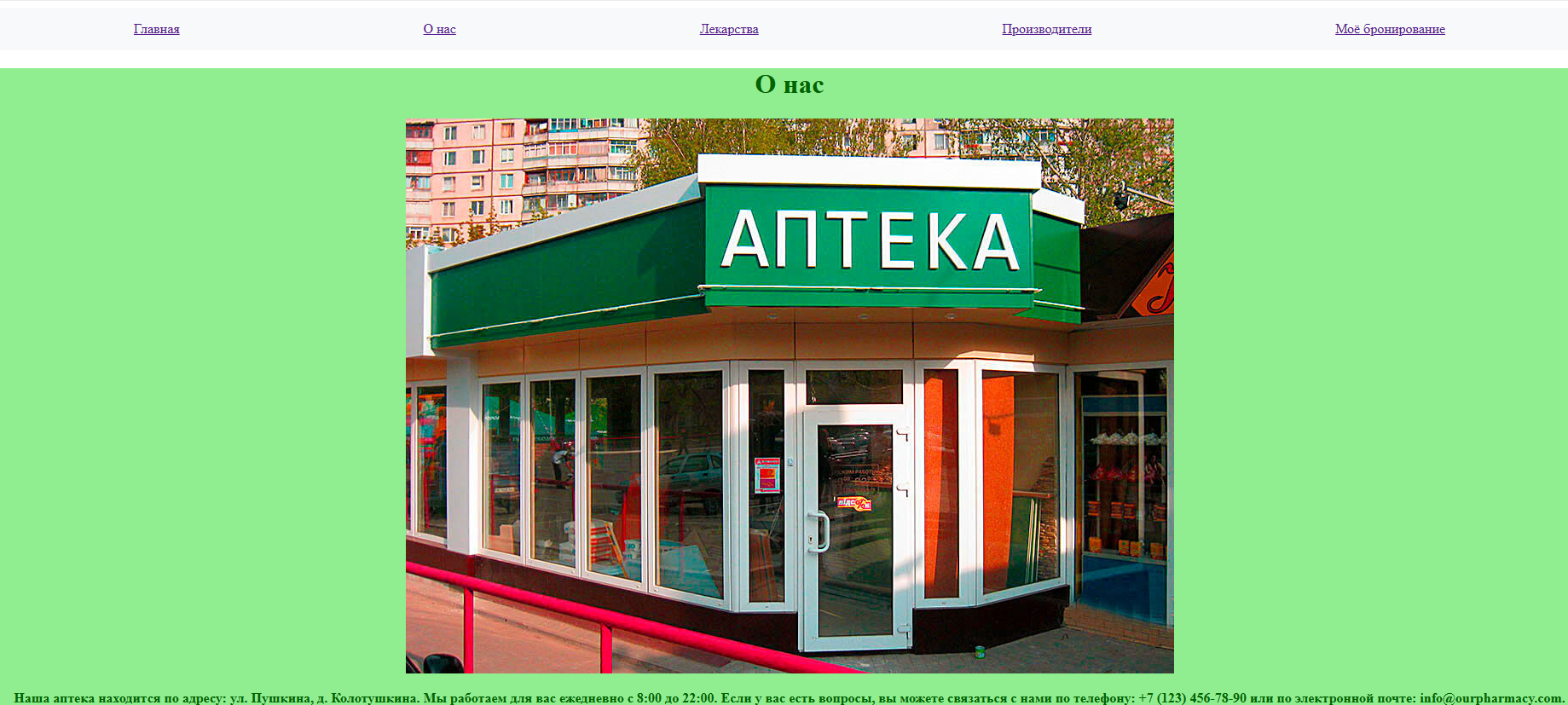
</style>

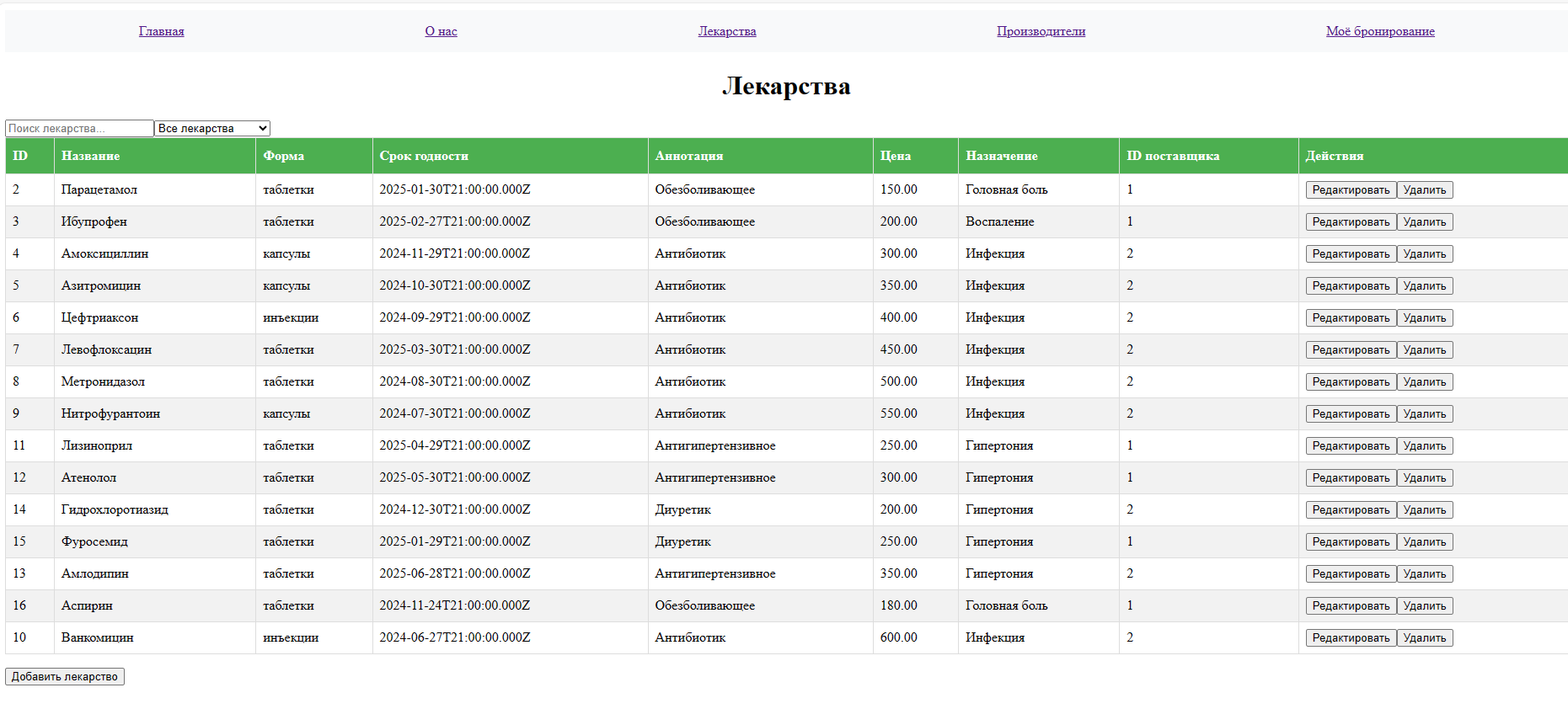
# Сеть Петри



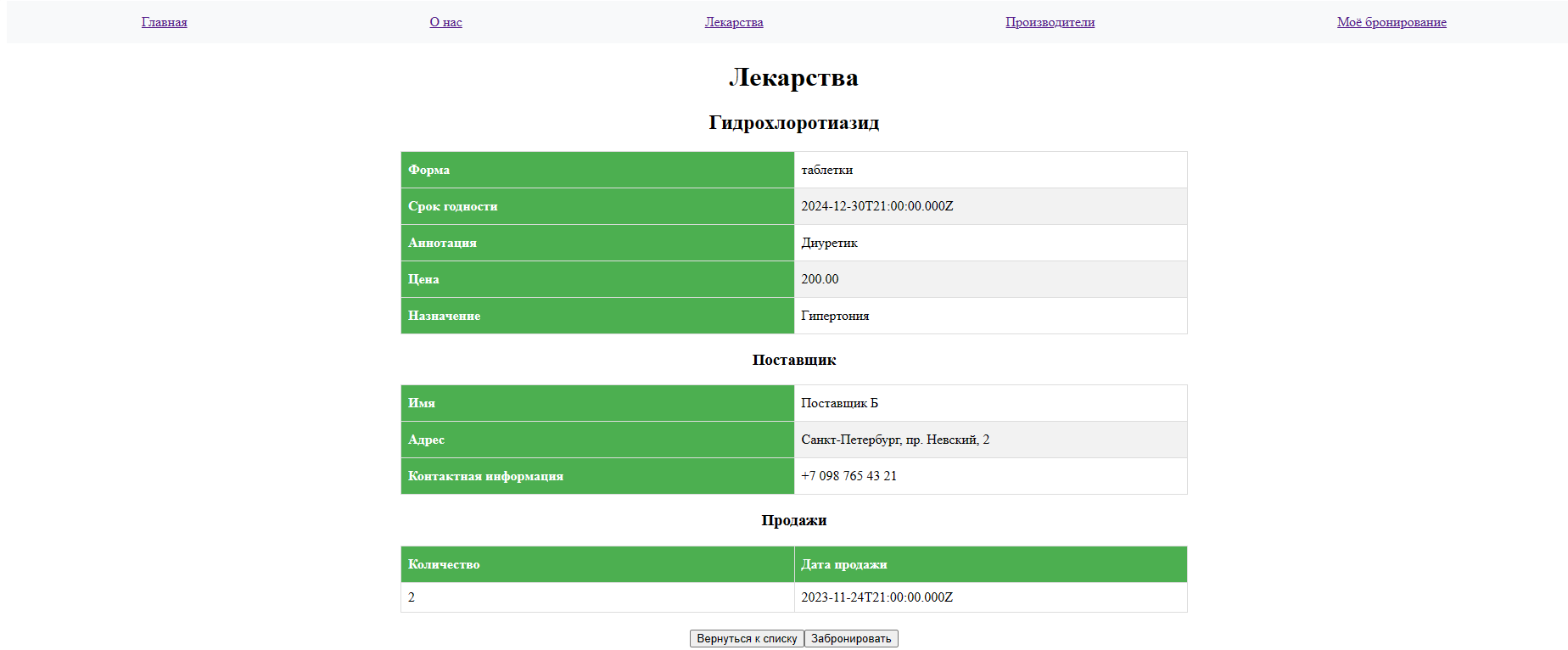
# Отображение в браузере

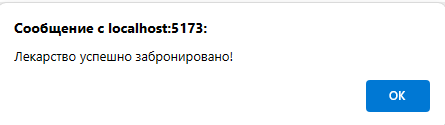


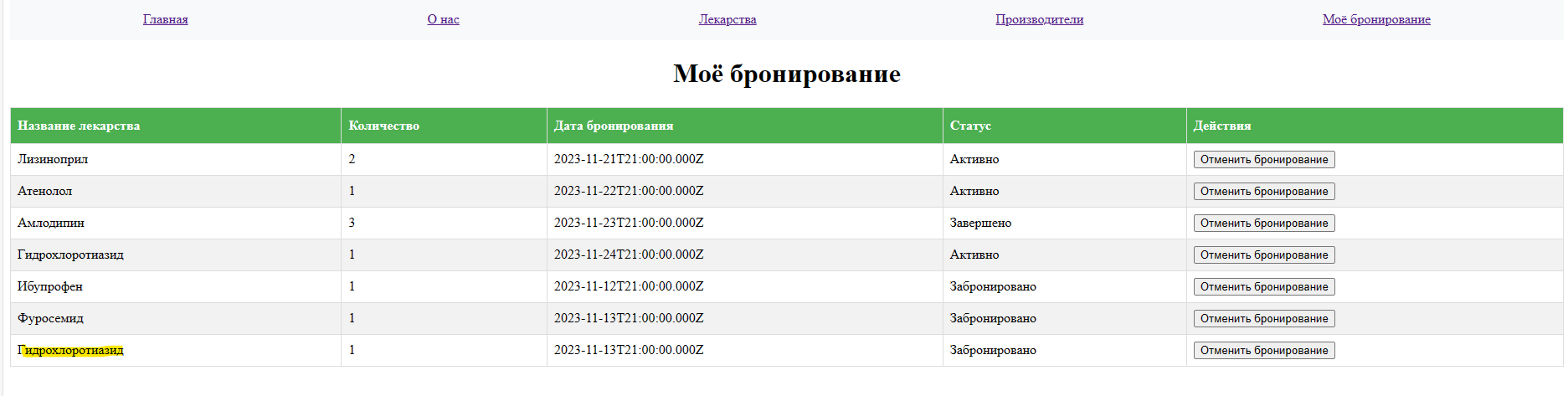


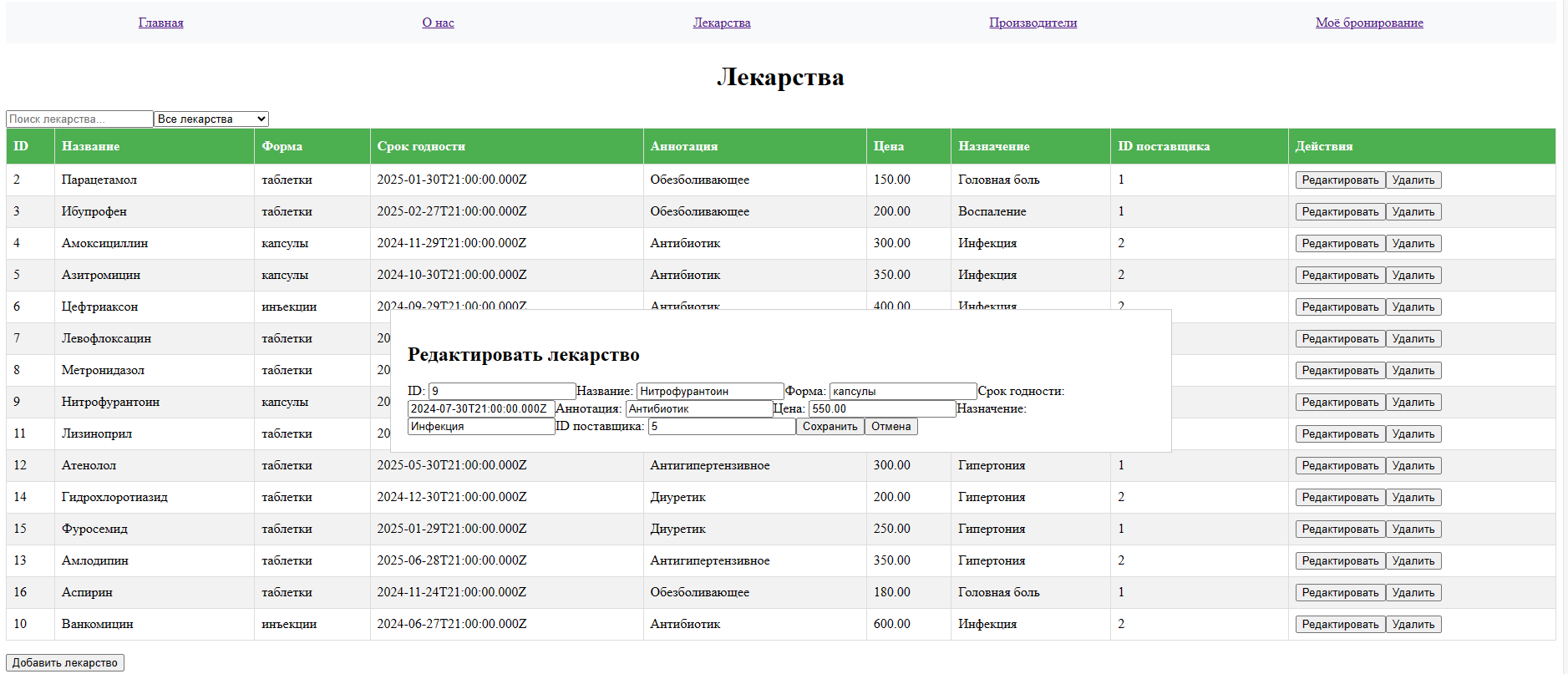


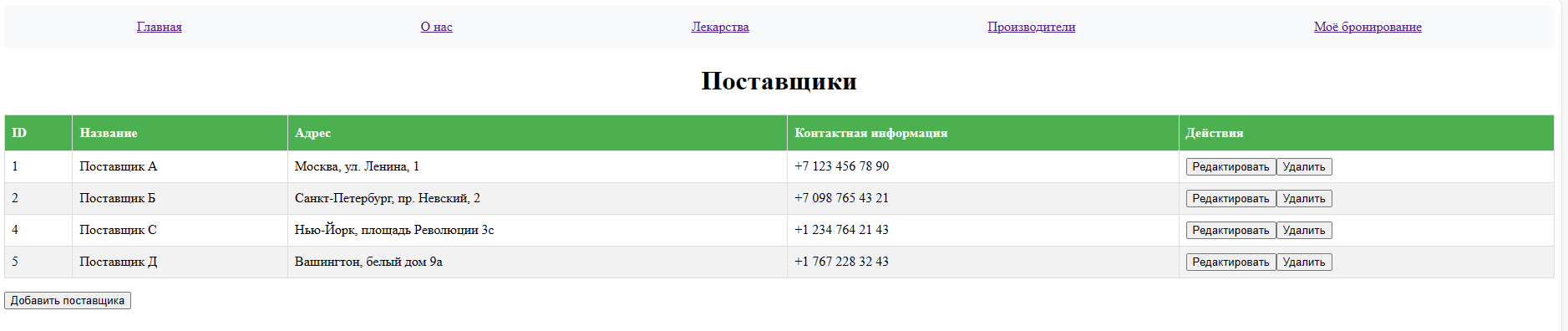












# Вывод

Проект был направлен на создание интернет-решения для БД «Аптека», обеспечивающего доступ к данным о медикаментах, поставщиках, транзакциях и резервированиях. Главная задача заключалась в разработке простого в использовании и многофункционального интерфейса, подходящего для разнообразных устройств и платформ.

По завершении проекта было установлено, что цель достигнута. Веб-приложение готово к внедрению и может быть использовано аптекой для повышения качества обслуживания клиентов и эффективности управления бизнес-процессами.