Министерство образования и науки Республики Казахстан

Колледж ТОО «Astana IT University»

Циклическая комиссия «Специальные дисциплины»

ПМ03 - Программирование модулей программного обеспечения

**КУРСОВОЙ ПРОЕКТ**

Приложение для управления и бронирования мест в спортивных мероприятиях и секциях

Выполнил:

ПО 2307 студент группы

Руководитель:

г.Астана 2025

**Содержание**

[**Введение 4**](#_2gqlb6rcubwr)

[**1.“ Знакомство с приложением «SportFlow» 8**](#_l9lkf6b20csm)

[Основное меню: 8](#)

[Принцип работы: 9](#)

[Особенности: 9](#)

[**2. SportFlow - Основные характеристики 9**](#_dgsl5lgbq1oq)

[Основные характеристики SportFlow: 10](#)

[1. Интерфейс на русском языке 10](#_xsb8415omoos)

[2. Система регистрации и входа 10](#_xtum2os3rurp)

[3. Поиск мероприятий и секций 10](#_83ca0j7esife)

[4. Бронирование спортивных мероприятий 11](#_xzmjf95t4n98)

[5. Управление бронированиями 11](#_v0cav6y370o5)

[6. Безопасность и защита данных 11](#_iym5meoz11it)

[7. Работа с базой данных 11](#_o8djy43jye86)

[**3. Документация SportFlow 12**](#_43ju1dxaxibn)

[1. React 12](#)

[2. Python 12](#)

[3. SQLite 13](#)

[4. HTML/CSS 13](#)

[5. Swiper 14](#_vbhpz8145bqg)

[6. Bcrypt.js 14](#)

[Структура проекта 17](#_w0nuvbl1vv90)

[**4. Разработка конструкции приложения 18**](#_4cdim0c444nw)

[4.1 Файл App.js 18](#)

[1. Импортированные модули 19](#)

[2. Функция приложения 20](#)

[2.1. Элемент < Router > 22](#)

[Принцип работы проекта 22](#_9nviu3a7mj1)

[4.2 Header.jsx 23](#_srqp4c7bw2o8)

[Краткое описание: файл Header.jsx 23](#_gn8x95urq9jg)

[1. Основной функционал 26](#)

[2. Управление состоянием авторизации 26](#)

[3. Стилизация 26](#)

[4. Элементы интерфейса 27](#_khk94tz67ogh)

[**5. База данных 28**](#_rcgwbkusj9js)

[**Заключение 31**](#_krmqoqi0j13w)

[1. Основные функции проекта 31](#_2clwr2d2lqwt)

[2. Технологии и инструменты 31](#_h2scoqpjuu5s)

[3. Структура базы данных 32](#_hg6o85x87g05)

[4. Структура проекта 33](#_cua858l26l75)

[4.1. Клиентский раздел (Frontend) 33](#_51sb59tyspu0)

[4.2. Серверный раздел (Backend) 33](#_yxnabypiwfyh)

[5. База данных 34](#_hkh3z3h8b1q6)

[**Список использованной литературы 34**](#_c93rl4jwin65)

[**ПРИЛОЖЕНИЕ 1 3**](#_ndu75ofnk34)**6**

[**ПРИЛОЖЕНИЕ 2 39**](#_zfwek9vf783e)

# **Введение**

В современном мире, наряду с развитием информационных технологий, особое значение приобретают вопросы эффективного управления данными и их использования во всех сферах человеческой жизни. Участие в спортивных мероприятиях и секциях является важной частью жизни активных граждан современного общества. В этом курсовом проекте поставлена цель создать систему бронирования мест для спортивных мероприятий и секций через создание веб-интерфейса с использованием фреймворка React и управление базой данных SQLite через Python. Основная цель данного проекта - предоставление пользователям решения, позволяющего оптимизировать процесс участия в спортивных мероприятиях и бронирования мест в секциях. Основная особенность проекта - это доступное, функциональное и общедоступное веб-приложение. С помощью этой системы пользователи смогут эффективно записываться на спортивные мероприятия и экономить время.

**Целями проекта являются:**

**1. Поиск и бронирование спортивных объектов:** Возможность пользователей искать и подписываться на спортивные мероприятия.

**2. Управление бронированием:** Возможность бронирования мест пользователями, их осмотр и управление..

**3. Удобный интерфейс:** Создание доступного и наглядного интерфейса для пользователей через React.

**4. Реализация системы авторизации:** Автоматизация процессов входа и регистрации пользователей.

**Актуальность проекта:**

Цифровизация процесса участия в спортивных мероприятиях и подписки на секции - это важный вопрос современного общества. Многие люди живут трудоголиком и стараются эффективно использовать свое время. Поэтому очень важна онлайн-подписка на спортивные секции и возможность предварительного бронирования мест. Данный проект направлен на решение этих проблем.

**Применяемые технологии:**

1. React: библиотека JavaScript, используемая для создания пользовательского интерфейса. Его компонентная архитектура и высокая производительность являются эффективным решением для этого проекта.
2. Python: язык программирования, используемый для создания серверной части. Простота и функциональность Python позволяют быстро и эффективно реализовывать проект.
3. SQLite: база данных, используемая для хранения и управления данными. Его легкость и продуктивность являются важным фактором реализации проекта.
4. HTML/CSS: Технологии, используемые для создания структуры и дизайна пользовательского интерфейса.

**Структура проекта:**

1. Форма регистрации: Форма для регистрации пользователей в системе. В форме пользователи вводят свое имя и пароль.
2. Форма входа: Форма для входа пользователей в систему. В форме пользователи вводят свое имя и пароль.
3. Список мероприятий: Список всех доступных спортивных мероприятий и секций.
4. Форма бронирования: Форма, предназначенная для бронирования мест на мероприятия или секции пользователями.
5. База данных: модель пользователей, мероприятий и бронирований, хранящаяся в базе данных SQLite.

В результате проекта пользователи получат возможность зарегистрироваться на спортивные мероприятия и записаться в секции. Через эту систему пользователям предоставляется доступный, легкий и эффективный интерфейс, который вносит вклад в развитие и применение технологий. SportFlow - это приложение позволяет пользователям легко бронировать и управлять местами для спортивных мероприятий и секций. Приложение выполнено на казахском языке, что обеспечивает удобный интерфейс для местных пользователей. Кроме того, в системе есть разделы регистрации и входа, что позволяет пользователям сохранять собственные бронирования и управлять данными в личном кабинете. Дизайн приложения прост и интуитивно понятен, что помогает пользователям легко ориентироваться в системе.

Данные о проекте:

1. Предоставление пользователям доступа к мероприятиям: Возможность поиска и бронирования спортивных мероприятий и секций..
2. Интерфейс на русском языке: Упрощение понимания и работы с интерфейсом пользователями.
3. Удобный дизайн: Дизайн приложения удобен и интуитивно понятен пользователю.
4. Реализация системы регистрации и доступа: Сохранение данных пользователей и предоставление им возможности работы в личном кабинете.
5. Управление бронированиями: Возможность просмотра, изменения и отмены собственных бронирований пользователями.

В результате проекта пользователи упростят процесс подписки на спортивные мероприятия и секции, что внесет вклад в развитие и применение технологий. Наличие приложения на русском языке и удобный дизайн дают пользователям хороший опыт.

**Негізгі бөлім**

# **1.“ Знакомство с приложением «SportFlow»**

**«SportFlow»** - это приложение для бронирования инновационных спортивных мероприятий с использованием современных технологий. Это приложение позволяет пользователям легко бронировать места для спортивных мероприятий и секций. Основная цель приложения - облегчить посещение спортивных мероприятий через удобный, интуитивно понятный и функциональный интерфейс.

#### **Основное меню:**

Интерфейс приложения «SportFlow» прост и удобен. Меню содержит следующие разделы:

1. **Главная:** В этом разделе пользователи могут ознакомиться с основными функциями приложения. На главной странице расположено поле поиска, в котором пользователь может просматривать доступные места, введя имя события или секции.
2. **Мероприятия**: В этом разделе пользователи могут увидеть список всех доступных спортивных мероприятий. О каждом мероприятии дается подробная информация с указанием даты, времени, места и количества вакансий.
3. **Мои бронирования:** Пользователи могут видеть свои собственные бронирования. В этом разделе пользователи могут управлять своими бронированиями (изменять или отменять).
4. **Выход:**

Пользователь использует эту кнопку для выхода из системы. После нажатия кнопки «Выход» сессия пользователя закрывается, а система автоматически авторизуется в целях безопасности.

#### **Принцип работы:**

1. **Регистрация или вход:**

Для полноценного использования приложения «SportFlow» пользователям необходимо сначала зарегистрироваться или войти в систему. При регистрации пользователи вводят свои имена и пароль. В разделе входа пользователи вводят свой логин и пароль.

1. **Выбор мероприятия:**

Пользователи выбирают любое событие из результатов поиска. Указывается полная информация о выбранном мероприятии: дата, время, место, цена и количество свободных мест.

1. **Забронировать место:**

Пользователи нажимают кнопку «Заполнить», чтобы забронировать место для выбранного мероприятия. В форме бронирования пользователь выбирает количество мест и подтверждает бронирование.

1. **Управление бронированиями:**

Пользователи могут просматривать свои собственные бронирования в разделе «Мои бронирования» и управлять ими (изменять или отменять).

#### **Особенности:**

* **Удобный дизайн:**Дизайн приложения удобен для пользователя и интуитивно понятен. Цветовые схемы и элементы навигации помогают пользователям легко работать с приложением.
* **Эффективная система бронирования:**Процесс бронирования прост и быстр, что экономит время пользователей.

# **2. SportFlow - Основные характеристики**

**«SportFlow»** - это приложение для бронирования инновационных спортивных мероприятий с использованием современных технологий. Его основная цель - облегчить посещение спорта, предоставив пользователям возможность легко бронировать места на спортивных мероприятиях и секциях..

#### **Основные характеристики SportFlow:**

#### Интерфейс на русском языке

* Интерфейс приложения полностью выполнен на русском языке.
* Это позволяет местным пользователям легко понимать и использовать приложение.

#### **Система регистрации и входа**

* Регистрация: Пользователи регистрируются в системе путем ввода своего имени и пароля.
* Вход: Зарегистрированные пользователи могут войти в систему, введя логин и пароль.
* Профиль: Пользователи могут управлять своими данными и бронированиями в личном кабинете..

#### **Поиск мероприятий и секций**

* Поле поиска: Пользователи могут искать спортивные события или секции через поле поиска, расположенное на главной странице.
* Фильтрация: Пользователи могут фильтровать мероприятия по видам спорта.
  + Параметры мероприятия:
  + Имя
  + Дата и время
  + Расположение
  + Цена
  + Количество свободных мест
  + Общие условия погоды (явления: дождь, снег, солнце, облачность)

#### **Бронирование спортивных мероприятий**

* Процесс бронирования: Пользователи могут забронировать место для выбранного мероприятия, нажав кнопку «Подробнее».
* В форме бронирования пользователь может выбрать необходимое количество мест.

#### **Управление бронированиями**

* Раздел «Мои бронирования»: Пользователи могут видеть свои собственные бронирования.

#### **Безопасность и защита данных**

* **Хэширование паролей:** Пароли пользователей хэшируются такими алгоритмами, как bcrypt или argon2.
* **Система авторизации:** Пользователи могут просматривать и управлять данными только в своих аккаунтах.

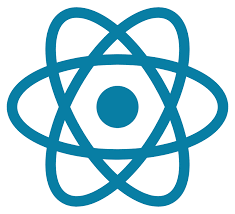
#### **Работа с базой данных**

* **База данных SQLite:** Данные пользователей, мероприятия и бронирования хранятся в базе данных SQLite.
* **Python:** Серверная часть написана на языке Python, что облегчает работу с базой данных.

# **3. Документация SportFlow**

#### **1. React**

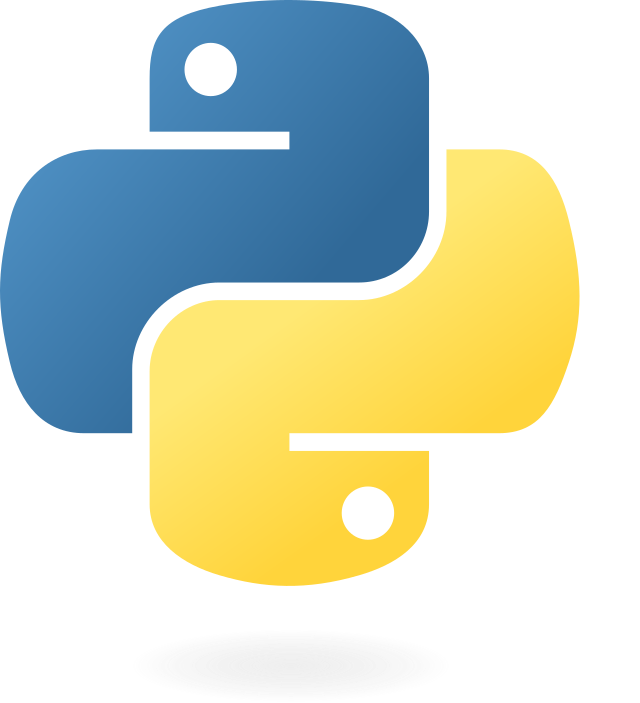
React - библиотека JavaScript, эффективное решение для создания компонентов UI. Он работает с компонентной структурой и использует виртуальный механизм DOM, что повышает скорость и производительность интерфейса. Эффективные функции и модульная структура React позволяют эффективно и масштабируемо создавать пользовательский интерфейс. (рисунок 3.1). [1]



Сурет 3.1 - React

#### **2. Python**

Python - язык программирования, используемый для создания серверной части. Это язык высокого уровня, интерпретируемый, он известен своей простотой и сильной функциональностью. Через Python были созданы серверные логики и созданы REST API.. (рисунок 3.2). [2]



Сурет 3.2 - Python

#### **3. SQLite**

SQLite - система легкой реляционной базы данных, используемая для хранения данных. Он не требует никаких серверов и может хранить всю базу данных в одном файле. Сохранены данные пользователей, мероприятий и бронирований через SQLite.. (рисунок 3.3). [3]



Сурет 3.4 - SQLite

#### **4. HTML/CSS**

HTML (HyperText Markup Language) - язык, определяющий структуру веб-страниц. CSS (Cascading Style Sheets) - язык, определяющий дизайн веб-страниц. Эти технологии составляют основу компонентов React и определяют представление пользовательского интерфейса.

#### **5. Swiper**

Swiper — это современный JavaScript слайдер с аппаратным ускорением переходов и отличной нативной поддержкой мобильных устройств. Swiper особенно полезен, когда требуется добавить интерактивные слайд-галереи, карусели изображений или пролистываемый контент. В рамках проекта Sports Booking, Swiper используется для создания интерактивных карточек спортивных мероприятий на главной странице, что значительно улучшает пользовательский опыт и делает интерфейс более динамичным. (Рисунок 3.5)[4]



#### **6. Bcrypt.js**

Bcrypt.js – паролдарды хэштеу үшін қолданылатын JavaScript кітапханасы. Ол пайдаланушылардың құпия сөздерін қауіпсіз түрде сақтауға мүмкіндік береді. Тіркелу және кіру процестерінде құпия сөздері хэштеу арқылы сақталады.(сурет 3.6)[5]



Сурет 3.6 - Bcrypt.js

**Структура файла**

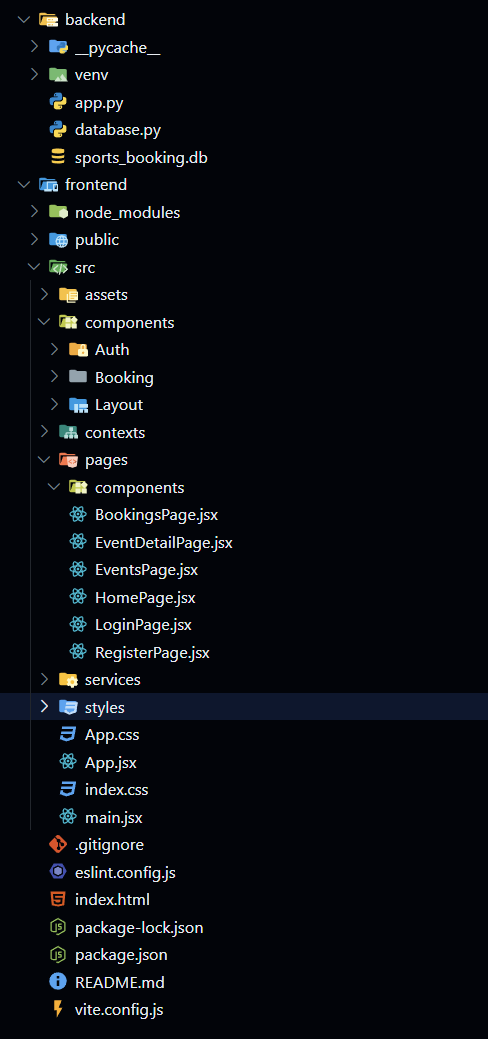


Рисунок 3.9 - Структура файла

Структура файла проекта содержит следующие разделы:

* **src/:** Включает клиентский код. Здесь расположены компоненты React (Header.jsx, Navbar.jsx, Footer.jsx), логические файлы (Auth.jsx, Forecast.jsx) и стили (styles.css).
* **server.js**: Содержит серверный код. Здесь расположены API, созданные с помощью Express.js, и логика, созданная для взаимодействия с PostgreSQL.
* **package.json**: Содержит зависимости и сценарии проекта.
* **.env**: Средний включает параметры (например, ключ API, данные базы).

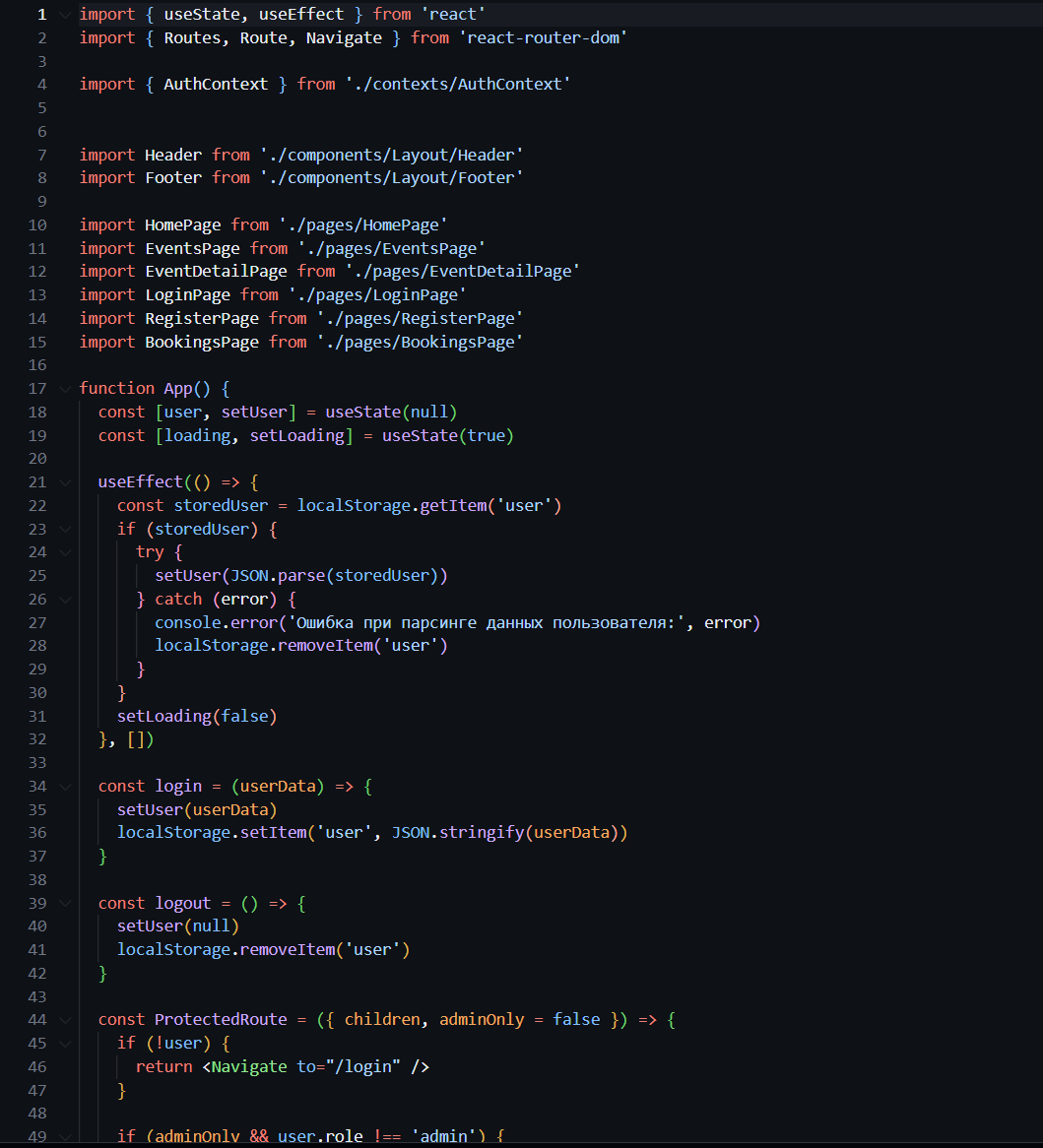
#### **Структура проекта**

1. **Клиентский раздел (Frontend):**
   1. Создан пользовательский интерфейс через компоненты React.
   2. Данные были отправлены и получены на сервер через запросы API.
2. **Серверный раздел (Backend):**
   1. Создано REST API через Python.
   2. Данные пользователя, мероприятия и бронирования сохранены через SQLite.
   3. Осуществлена бизнес-логика (авторизация пользователей, поиск мероприятий, управление бронированиями).
3. **База данных (Database):**
   1. Данные пользователя, мероприятия и бронирования сохранены через SQLite.
   2. Установлены связи между данными с помощью реляционной модели.

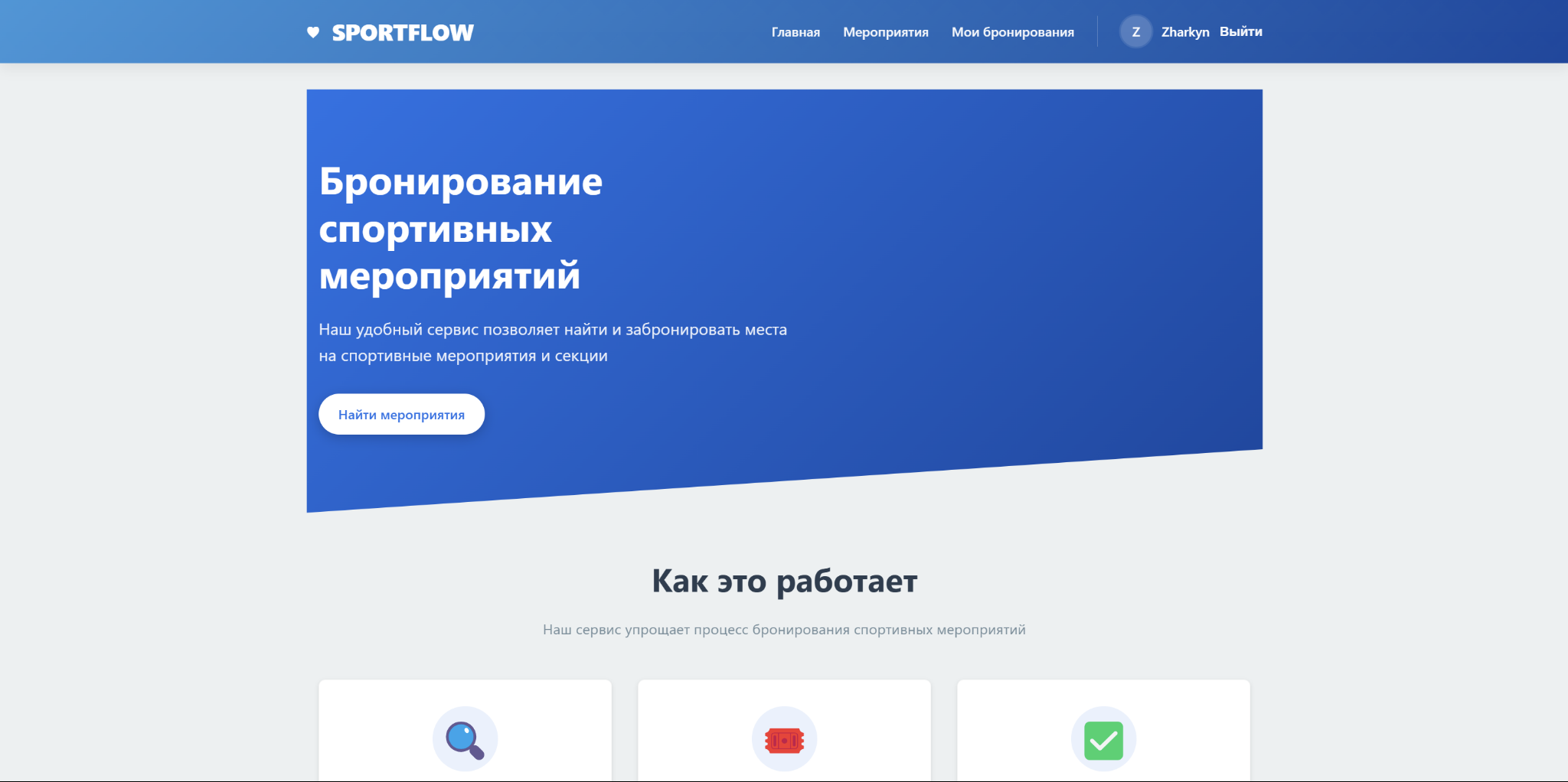
# **Разработка конструкции приложения**

### **4.1 Файл App.js**

Файл App.js - основной файл приложения React. Он определяет основную структуру приложения и управляет маршрутами. С помощью этого файла будет проверена авторизация пользователя и перенаправлена на соответствующие страницы.. (рисунок 4.1)



Сурет 4.1.A - Бастапқы бет

Сурет 4.1.1 - Бастапқы бет

#### **1. Импортированные модули**

import React from "react";  
import { BrowserRouter as Router, Route, Routes, Navigate } from "react-router-dom";  
import Home\_page from "./HomePage";  
import EventsPage from "./EventsPage";  
import EventDetailPage from "./EventDetailPage";  
import LoginPage from "./LoginPage";   
import RegisterPage from "./RegisterPage";  
import BookingsPage from "./BookingsPage";

* **React:** Включает основные функции React.
  + Route: Позволяет указать компонент, соответствующий URL-адресу пользователя.
  + Routes: Позволяет управлять всеми маршрутами в одном месте.
  + Navigate: Используется для автоматического отправки пользователя на другой маршрут.
* **HomePage, EventsPage, EventsDetailPage, RegisterPage, BookingsPage**: Каждый компонент содержит определенную часть приложения.

#### **2. Функция приложения**

import { useState, useEffect } from 'react'

import { Routes, Route, Navigate } from 'react-router-dom'

import { AuthContext } from './contexts/AuthContext'

import Header from './components/Layout/Header'

import Footer from './components/Layout/Footer'

import HomePage from './pages/HomePage'

import EventsPage from './pages/EventsPage'

import EventDetailPage from './pages/EventDetailPage'

import LoginPage from './pages/LoginPage'

import RegisterPage from './pages/RegisterPage'

import BookingsPage from './pages/BookingsPage'

function App() {

const [user, setUser] = useState(null)

const [loading, setLoading] = useState(true)

useEffect(() => {

const storedUser = localStorage.getItem('user')

if (storedUser) {

try {

setUser(JSON.parse(storedUser))

} catch (error) {

console.error('Ошибка при парсинге данных пользователя:', error)

localStorage.removeItem('user')

}

}

setLoading(false)

}, [])

const login = (userData) => {

setUser(userData)

localStorage.setItem('user', JSON.stringify(userData))

}

const logout = () => {

setUser(null)

localStorage.removeItem('user')

}

const ProtectedRoute = ({ children, adminOnly = false }) => {

if (!user) {

return <Navigate to="/login" />

}

if (adminOnly && user.role !== 'admin') {

return <Navigate to="/" />

}

return children

}

if (loading) {

return <div className="container text-center p-5">Загрузка...</div>

}

return (

<AuthContext.Provider value={{ user, login, logout }}>

<div className="app">

<Header />

<main className="container mt-4 mb-4">

<Routes>

<Route path="/" element={<HomePage />} />

<Route path="/events" element={<EventsPage />} />

<Route path="/events/:id" element={<EventDetailPage />} />

<Route path="/login" element={<LoginPage />} />

<Route path="/register" element={<RegisterPage />} />

<Route

path="/bookings"

element={

<ProtectedRoute>

<BookingsPage />

</ProtectedRoute>

}

/>

<Route

path="/admin"

element={

<ProtectedRoute adminOnly={true}>

</ProtectedRoute>

}

/>

<Route path="\*" element={<Navigate to="/" />} />

</Routes>

</main>

<Footer />

</div>

</AuthContext.Provider>

)

}

export default App

#### **2.1. Элемент < Router >**

* **Назначение:** Применяется для управления маршрутами Приложения.
* **Принцип работы:** URL мекенжайына сәйкес компонентті көрсетеді.

#### **Принцип работы проекта**

1. **Если пользователь не авторизован:**
   1. Запуск маршрута при входе пользователя в приложение.
2. **При авторизации пользователя:**
   1. После авторизации пользователя/home отправляется на маршрут.
   2. / homepage доступен на внутренних страницах маршрута (contact, profile, logout).
3. **Недопустимые маршруты:**
   1. Если пользователь заходит на недопустимый маршрут, то отображается сообщение «Страница не найдена».

### **4.2 Header.jsx**

#### **Краткое описание: файл Header.jsx**

Данный код является компонентом, представляющим верхнюю страницу веб-приложения, созданного в React. Он принимает поисковые запросы пользователя и отображает начало интерфейса. (рисунок 4.2)

import { useContext, useState, useEffect, useRef } from 'react';

import { Link, useNavigate } from 'react-router-dom';

import { AuthContext } from '../../contexts/AuthContext';

import '../../styles/Header.css';

const Header = () => {

const { user, logout } = useContext(AuthContext);

const navigate = useNavigate();

const [mobileMenuOpen, setMobileMenuOpen] = useState(false);

const menuRef = useRef(null);

const buttonRef = useRef(null);

const handleLogout = () => {

logout();

navigate('/');

};

const toggleMobileMenu = () => {

setMobileMenuOpen(!mobileMenuOpen);

};

useEffect(() => {

const handleClickOutside = (event) => {

if (

mobileMenuOpen &&

menuRef.current &&

!menuRef.current.contains(event.target) &&

buttonRef.current &&

!buttonRef.current.contains(event.target)

) {

setMobileMenuOpen(false);

}

};

document.addEventListener('mousedown', handleClickOutside);

return () => {

document.removeEventListener('mousedown', handleClickOutside);

};

}, [mobileMenuOpen]);

const getInitial = (name) => {

return name ? name.charAt(0).toUpperCase() : 'П';

};

return (

<header className="header">

<div className="container header-container">

<div className="logo">

<Link to="/">

<span className="logo-icon">&#9829;</span>

&nbsp;SportFlow

</Link>

</div>

<div className="nav-container">

<button

ref={buttonRef}

className="mobile-menu-button"

onClick={toggleMobileMenu}

aria-label="Открыть навигационное меню"

>

&#9776;

</button>

<ul

ref={menuRef}

className={`nav-links ${mobileMenuOpen ? 'active' : ''}`}

>

<li>

<Link to="/">Главная</Link>

</li>

<li>

<Link to="/events">Мероприятия</Link>

</li>

{user ? (

<>

<li>

<Link to="/bookings">Мои бронирования</Link>

</li>

<li className="user-info">

<div className="user-avatar">

{getInitial(user.username)}

</div>

<span className="user-name">{user.username}</span>

<button onClick={handleLogout}>

Выйти

</button>

</li>

</>

) : (

<>

<li>

<Link to="/login">Войти</Link>

</li>

<li>

<Link to="/register">Зарегистрироваться</Link>

</li>

</>

)}

</ul>

</div>

</div>

</header>

);

};

export default Header;

Рисунок 4.2.1 - Header.jsx

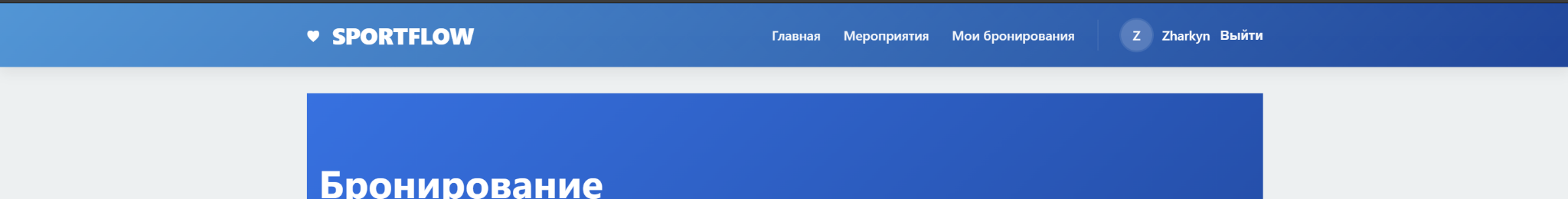


Рисунок 4.2.2 - Header.jsx

#### **1. Основной функционал**

* **useContext**: Используется для получения данных пользователя (user) и функции вывода (logout) из AuthContext.
* **useState**:Используется для сохранения состояния мобильного меню (mobileMenuOpen).
* **useNavigate**:Используется для перенаправления пользователя на другие страницы.
* **React Router**: Использует компоненты Link для навигации.

#### **2. Управление состоянием авторизации**

* Пользователь отображает различные элементы в зависимости от того, вошли они или нет.
* Авторизованным пользователям предоставляется доступ к разделу «Мои бронирования».

#### **3. Стилизация**

* Стили используются с помощью модулей CSS.
* Придерживается принципов адаптивного дизайна.

#### **4. Элементы интерфейса**

* **Логотип**: Логотип и название SportFlow.
* **Навигационное меню:** Ссылки на основные страницы (Главная, Мероприятия) и пользовательские страницы (Мои брони).
* **Кнопка мобильного меню:**Кнопка гамбургер

# **5. База данных**

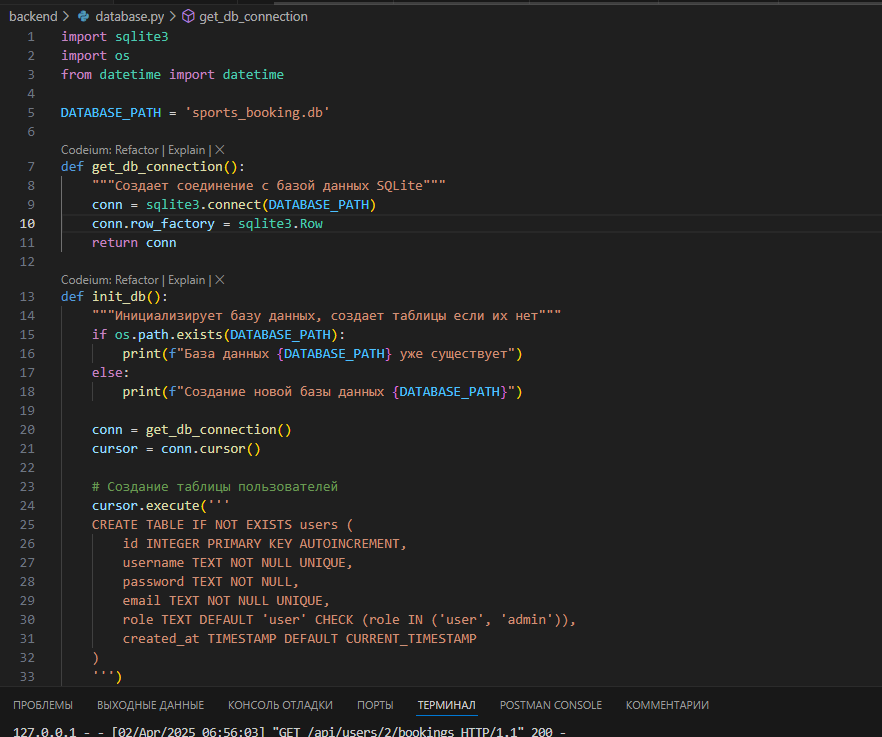


Рисунок 5.1.1 - База данных



Рисунок 5.2.2 - Структура базы данных

**База данных**

В проекте используется база данных SQLite, которая является легкой и эффективной реляционной системой базы данных. Его главное преимущество в том, что он хранит всю базу данных в одном файле и не требует сервера. Ниже приведены подробные сведения об основных таблицах и их важных графах в базе данных проекта. (рисунок 5.1, 5.2)

**1. users**

**Цель:** Сохранить данные пользователей.

| **Столбец** | **Тип** | **Цель** |
| --- | --- | --- |
| id | INTEGER PRIMARY KEY AUTOINCREMENT | Уникальный идентификатор пользователя. |
| username | TEXT NOT NULL UNIQUE | Имя пользователя,обязательно должно быть уникальным |
| role | TEXT DEFAULT ‘user’ | Роль пользователя,по умолчанию “Пользователь” |
| password | TEXT NOT NULL | Пароль учетной записи |
| email | TEXT NOT NULL UNIQUE | Электронная почта пользователя,обязательно должно быть уникальным |

**2. bookings**

**Цель:** Сохранение бронирования пользователей на мероприятия.

| **Баған** | **Тип** | **Мақсат** |
| --- | --- | --- |
| id | INTEGER PRIMARY KEY AUTOINCREMENT | Уникальный идентификатор брони (генерируется автоматически). |
| seats | INTEGER NOT NULL | Количество забронированных мест. |
| total\_price | INTEGER NOT NULL | Общая стоимость бронирования. |
| status | VARCHAR(50) DEFAULT 'confirmed' | Состояние бронирования (подтверждено, отменено). |
| created\_at | TIMESTAMP DEFAULT CURRENT\_TIMESTAMP | Время бронирования. |

# **Заключение**

**SportFlow -** это инновационное веб-приложение для бронирования мест для спортивных мероприятий и секций. Его основная цель - дать возможность пользователям активно заниматься спортом, облегчив процесс участия в спортивных мероприятиях и подписки на секции.

#### **1. Основные функции проекта**

**Поиск и бронирование мероприятий:** Пользователи могут искать спортивные мероприятия и бронировать необходимые места.

**Интерфейс на русском языке:** Интерфейс на казахском языке удобен для местных пользователей.

**Система регистрации и входа:** Пользователи могут зарегистрироваться и управлять своими бронированиями

**Управление бронированиями:** Пользователи имеют возможность просматривать, изменять и отменять свои бронирования.

**Адаптивный интерфейс:** Приложение предлагает адаптивный дизайн, который хорошо отображается на различных устройствах.

#### **2. Технологии и инструменты**

В развитии проекта использованы следующие технологии и инструменты:

React: библиотека JavaScript, используемая для создания клиентского интерфейса.

Python: язык программирования, используемый для создания серверной части [1].

SQLite: легкая реляционная база данных, используемая для хранения и управления данными.

HTML/CSS: Технологии, используемые для создания структуры и дизайна пользовательского интерфейса.

React Router: Библиотека, используемая для управления маршрутами в приложении.

Context API: механизм React, используемый для управления состоянием приложения.

Visual Studio Code: Редактор, используемый для записи и редактирования кода проекта.

#### **3. Структура базы данных**

База данных SQLite обеспечивает основные функционалы проекта:

**users:** сохраняет данные пользователей (логин, пароль, роль).

**events:** Сохраняет данные о спортивных мероприятиях (название, время, место, цена).

**venues:** Сохраняет данные о местах проведения спортивных мероприятий.

**bookings**: Сохраняет бронирования пользователей на мероприятия.

#### **4. Структура проекта**

##### **4.1. Клиентский раздел (Frontend)**

**App.jsx:** Управляет основными маршрутами ( **/** , **/events** , **/login** , **/bookings** ) и авторизацией.

**Header.jsx:** отображает навигационное меню и профиль пользователя.

**Footer.jsx:** Отображает контактные данные и навигационные ссылки.

**EventsPage.jsx:** Показывает все доступные мероприятия.

**EventDetailPage.jsx:** Отображает подробную информацию о выбранном мероприятии и форму бронирования.

**BookingsPage.jsx:** Показывает и позволяет управлять бронированиями пользователя.

**LoginPage.jsx** и **RegisterPage.jsx:** управляет регистрацией и доступом пользователей.

##### **4.2. Серверный раздел (Backend)**

**Python:** RESTful используется для создания API.

**SQLite:** Используется для хранения данных пользователей, мероприятий и бронирований.

**API Service:** Обеспечивает обмен данными между клиентом и сервером.

**5. Особенности**

**Интуитивно понятный пользовательский интерфейс:** Простой и понятный интерфейс позволяет пользователям легко освоить приложение.

**Адаптивный дизайн:** адаптивный дизайн, который хорошо работает как на смартфонах, так и на планшетах.

**Безопасность:** для защиты данных пользователей внедрена система авторизации.

**Эффективное управление бронированиями:** Пользователи могут легко управлять своими бронированиями, включая возможность отмены.

#### **5. База данных**

/:Общая информация о главной странице, приложении.

**/events**: Список всех доступных мероприятий.

**/logout:** Выход из системы.

/ **login**: Форма входа в систему.

**/register**: Форма регистрации.

**/bookings**:Бронирование пользователем

SportFlow - это инновационное решение, которое облегчает пользователям участие в спортивных мероприятиях. Его интерфейс на казахском языке, удобный дизайн, эффективная система управления бронированиями и меры безопасности предоставляют пользователям удобство для занятий спортом. Применение React, SQLite, Python и других технологий обеспечивает эффективность и масштабирование проекта.

# **Список использованной литературы**

React. Официальная документация [Электронный ресурс]. URL: <https://reactjs.org/docs/getting-started.html>

Python. Документация [Электронный ресурс]. URL: <https://docs.python.org/3/>

SQLite. Официальный сайт [Электронный ресурс]. URL: <https://www.sqlite.org/docs.html>

HTML/CSS. MDN Web Docs [Электронный ресурс]. URL: <https://developer.mozilla.org/ru/docs/Web>

React Router. Документация [Электронный ресурс]. URL: <https://reactrouter.com/docs/en/v6>

Bcrypt.js. Репозиторий на GitHub [Электронный ресурс]. URL: <https://github.com/kelektiv/node.bcrypt.js>

REST API Design. Best Practices [Электронный ресурс]. URL: <https://restfulapi.net/>

Swiper [Электронный ресурс] URL:

<https://swiperjs.com/>

1. Visual Studio Code. Документация [Электронный ресурс]. URL: <https://code.visualstudio.com/docs>

# **ПРИЛОЖЕНИЕ 1**

**App.jsx**

**import { useState, useEffect } from 'react'**

**import { Routes, Route, Navigate } from 'react-router-dom'**

**import { AuthContext } from './contexts/AuthContext'**

**import Header from './components/Layout/Header'**

**import Footer from './components/Layout/Footer'**

**import HomePage from './pages/HomePage'**

**import EventsPage from './pages/EventsPage'**

**import EventDetailPage from './pages/EventDetailPage'**

**import LoginPage from './pages/LoginPage'**

**import RegisterPage from './pages/RegisterPage'**

**import BookingsPage from './pages/BookingsPage'**

**function App() {**

**const [user, setUser] = useState(null)**

**const [loading, setLoading] = useState(true)**

**useEffect(() => {**

**const storedUser = localStorage.getItem('user')**

**if (storedUser) {**

**try {**

**setUser(JSON.parse(storedUser))**

**} catch (error) {**

**console.error('Ошибка при парсинге данных пользователя:', error)**

**localStorage.removeItem('user')**

**}**

**}**

**setLoading(false)**

**}, [])**

**const login = (userData) => {**

**setUser(userData)**

**localStorage.setItem('user', JSON.stringify(userData))**

**}**

**const logout = () => {**

**setUser(null)**

**localStorage.removeItem('user')**

**}**

**const ProtectedRoute = ({ children, adminOnly = false }) => {**

**if (!user) {**

**return <Navigate to="/login" />**

**}**

**if (adminOnly && user.role !== 'admin') {**

**return <Navigate to="/" />**

**}**

**return children**

**}**

**if (loading) {**

**return <div className="container text-center p-5">Загрузка...</div>**

**}**

**return (**

**<AuthContext.Provider value={{ user, login, logout }}>**

**<div className="app">**

**<Header />**

**<main className="container mt-4 mb-4">**

**<Routes>**

**<Route path="/" element={<HomePage />} />**

**<Route path="/events" element={<EventsPage />} />**

**<Route path="/events/:id" element={<EventDetailPage />} />**

**<Route path="/login" element={<LoginPage />} />**

**<Route path="/register" element={<RegisterPage />} />**

**<Route**

**path="/bookings"**

**element={**

**<ProtectedRoute>**

**<BookingsPage />**

**</ProtectedRoute>**

**}**

**/>**

**<Route**

**path="/admin"**

**element={**

**<ProtectedRoute adminOnly={true}>**

**</ProtectedRoute>**

**}**

**/>**

**<Route path="\*" element={<Navigate to="/" />} />**

**</Routes>**

**</main>**

**<Footer />**

**</div>**

**</AuthContext.Provider>**

**)**

**}**

**export default App**

# **ПРИЛОЖЕНИЕ 2**

**Main.jsx**

**import React from 'react'**

**import ReactDOM from 'react-dom/client'**

**import { BrowserRouter } from 'react-router-dom'**

**import App from './App.jsx'**

**import './styles/global.css'**

**import './index.css'**

**ReactDOM.createRoot(document.getElementById('root')).render(**

**<React.StrictMode>**

**<BrowserRouter>**

**<App />**

**</BrowserRouter>**

**</React.StrictMode>,**

**)**

**Header.jsx**

import { useContext, useState, useEffect, useRef } from 'react';

import { Link, useNavigate } from 'react-router-dom';

import { AuthContext } from '../../contexts/AuthContext';

import '../../styles/Header.css';

const Header = () => {

const { user, logout } = useContext(AuthContext);

const navigate = useNavigate();

const [mobileMenuOpen, setMobileMenuOpen] = useState(false);

const menuRef = useRef(null);

const buttonRef = useRef(null);

const handleLogout = () => {

logout();

navigate('/');

};

const toggleMobileMenu = () => {

setMobileMenuOpen(!mobileMenuOpen);

};

useEffect(() => {

const handleClickOutside = (event) => {

if (

mobileMenuOpen &&

menuRef.current &&

!menuRef.current.contains(event.target) &&

buttonRef.current &&

!buttonRef.current.contains(event.target)

) {

setMobileMenuOpen(false);

}

};

document.addEventListener('mousedown', handleClickOutside);

return () => {

document.removeEventListener('mousedown', handleClickOutside);

};

}, [mobileMenuOpen]);

const getInitial = (name) => {

return name ? name.charAt(0).toUpperCase() : 'П';

};

return (

<header className="header">

<div className="container header-container">

<div className="logo">

<Link to="/">

<span className="logo-icon">&#9829;</span>

&nbsp;SportFlow

</Link>

</div>

<div className="nav-container">

<button

ref={buttonRef}

className="mobile-menu-button"

onClick={toggleMobileMenu}

aria-label="Открыть навигационное меню"

>

&#9776;

</button>

<ul

ref={menuRef}

className={`nav-links ${mobileMenuOpen ? 'active' : ''}`}

>

<li>

<Link to="/">Главная</Link>

</li>

<li>

<Link to="/events">Мероприятия</Link>

</li>

{user ? (

<>

<li>

<Link to="/bookings">Мои бронирования</Link>

</li>

<li className="user-info">

<div className="user-avatar">

{getInitial(user.username)}

</div>

<span className="user-name">{user.username}</span>

<button onClick={handleLogout}>

Выйти

</button>

</li>

</>

) : (

<>

<li>

<Link to="/login">Войти</Link>

</li>

<li>

<Link to="/register">Зарегистрироваться</Link>

</li>

</>

)}

</ul>

</div>

</div>

</header>

);

};

export default Header;