



Geekbrains

Разработка веб-приложения для отслеживания актуальных мероприятий с возможностью добавления в избранное

Программа: разработчик

Специализация: Fullstack разработчик

Алексеев Дмитрий Михайлович

Санкт-Петербург

2025

Оглавление

Оглавление.....	2
Введение.....	4
1 Теоретический экскурс	5
1.1 Введение в веб-разработку.....	5
1.2 React: библиотека для создания пользовательских интерфейсов	5
1.2.1 Введение в React.....	5
1.2.2 Основные концепции и принципы React.....	5
1.2.3 CreateAsyncThunk и Store в React.....	8
1.2.4 Преимущества использования React.....	10
1.2.5 Выводы о React.....	10
1.3 Laravel: PHP-фреймворк для бэкенд-разработки.....	11
1.3.1 Введение в Laravel	11
1.3.2 Основные концепции и принципы Laravel.....	11
1.3.3 Inertia.js в Laravel.....	13
1.3.4 Преимущества использования Laravel.....	15
1.3.5 Выводы о Laravel.....	15
1.4 Tailwind CSS	16
1.5 Архитектура веб-приложения.....	17
1.6 Выводы по теоретической части	18
2 Практическая реализация	19
2.1 Принцип разворачивания среды разработки приложения.....	19
2.2 Подключение React к Laravel приложению	20
2.3 Функциональность приложения	21

2.3.1	Общая функциональность	21
2.3.2	Функциональность авторизации и добавления в избранное	23
2.3.3	Хранение информации в базе данных	27
2.3.4	Взаимодействие модулей программы	28
2.4	Выводы о практической реализации	31
	Заключение	32
	Список используемой литературы	34
	Приложения	35
	Приложение 1. Конфигурация Docker	35
	Приложение 2. Листинг модулей программы	40

Введение

В современном мире, насыщенном информацией и событиями, важность эффективного управления временем и ресурсами становится все более актуальной. С каждым днем увеличивается количество мероприятий, которые могут заинтересовать пользователей, будь то культурные события, спортивные соревнования, выставки или образовательные программы. Однако, несмотря на обилие информации, пользователям часто сложно отслеживать актуальные события и находить те, которые соответствуют их интересам.

В связи с этим, разработка веб-приложения для отслеживания актуальных мероприятий представляет собой важную задачу, способствующую упрощению доступа к информации и улучшению пользовательского опыта. Данное приложение будет предоставлять пользователям возможность не только просматривать события, но и добавлять их в избранное, что позволит легко управлять своим временем и планами.

Целью данной работы является создание интуитивно понятного и функционального веб-приложения, которое будет включать в себя такие ключевые функции, как поиск мероприятий и возможность добавления событий в избранное. В ходе работы будут рассмотрены современные технологии веб-разработки и подходы к проектированию пользовательского интерфейса.

Таким образом, данное исследование направлено на решение актуальной проблемы поиска и управления мероприятиями, что, в свою очередь, будет способствовать более эффективному использованию времени и ресурсов пользователями.

Используемые инструменты: React, Laravel, Tailwind CSS, Postman, Git, VsCode, Docker, Google Chrome, Composer, Node.js.

Состав команды: Алексеев Дмитрий Михайлович (Fullstack-Разработчик).

1 Теоретический экскурс

1.1 Введение в веб-разработку

Веб-разработка представляет собой процесс создания веб-приложений и сайтов, который включает в себя как фронтенд (клиентская часть), так и бэкенд (серверная часть). С развитием технологий и увеличением потребностей пользователей, веб-приложения становятся все более сложными и функциональными. В данной работе рассматривается разработка веб-приложения для отслеживания актуальных мероприятий с использованием технологий React и Laravel.

1.2 React: библиотека для создания пользовательских интерфейсов

1.2.1 Введение в React.

React — это библиотека JavaScript, разработанная компанией Facebook для создания пользовательских интерфейсов. Она была выпущена в 2013 году и быстро завоевала популярность благодаря удобству и мощи, что сделало ее одной из основных технологий во фронтенд-разработке. React позволяет разработчикам создавать сложные приложения, которые могут эффективно обновлять и отображать данные в реальном времени.

1.2.2 Основные концепции и принципы React

1.2.2.1 Компоненты.

Компоненты являются основным строительным блоком приложений на React. Каждый компонент инкапсулирует свою логику, состояние и структуру, что позволяет повторно использовать код и разрабатывать приложения модульно.

Компоненты могут быть функциональными или классовыми [2]:

- **Функциональные компоненты:** это простые JavaScript-функции, которые возвращают JSX (расширенный синтаксис JavaScript, используемый для описания интерфейса). Они легче и быстрее, а с введением React Hooks (в версии 16.8) они также могут управлять состоянием.

```
const MyComponent = () => {  
  return <div>Hello, World!</div>;  
};
```

Рисунок 1 – Функциональный компонент

- **Классовые компоненты:** они создаются с использованием ES6-классов и дают больше возможностей для работы со состоянием и жизненным циклом компонента.

```
class MyComponent extends React.Component {  
  render() {  
    return <div>Hello, World!</div>;  
  }  
}
```

Рисунок 2 – Классовый компонент

1.2.2.2 JSX

JSX (JavaScript XML) — это расширение синтаксиса JavaScript, которое позволяет писать HTML-подобную структуру в коде React. JSX делает код более читаемым и удобным для разработки пользовательских интерфейсов.

JSX выражения могут содержать как статические, так и динамические данные через фигурные скобки {}:

```
const name = 'John';  
const element = <h1>Hello, {name}!</h1>;
```

Рисунок 3 – Динамические данные в JSX

1.2.2.3 Состояние и свойства

В React концепции состояния и свойств (props) играют ключевую роль в управлении данными компонентами:

- Свойства (props): это неизменяемые данные, которые передаются от родительских компонентов к дочерним и используются для передачи информации и конфигурации компонента.

```
const Greeting = ({ name }) => <h1>Hello, {name}!</h1>;
```

Рисунок 4 – Свойства props

- Состояние (state): это изменяемое, локальное состояние компонента, которое может изменяться в ответ на взаимодействие пользователя или другие события. Изменение состояния вызывает повторный рендер компонента.

```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  increment = () => {
    this.setState({ count: this.state.count + 1 });
  };

  render() {
    return (
      <div>
        <p>{this.state.count}</p>
        <button onClick={this.increment}>Increment</button>
      </div>
    );
  }
}
```

Рисунок 5 – Состояние state

1.2.2.4 Жизненный цикл компонента

React предоставляет жизненный цикл компонента, который состоит из нескольких методов, выполняемых в разные моменты времени:

- **mounting**: когда компонент создается и добавляется в DOM (например, методы `componentDidMount`, `componentWillMount`).
- **updating**: когда компонент обновляется в результате изменений состояния или свойств (например, методы `componentDidUpdate`).
- **unmounting**: когда компонент удаляется из DOM (например, метод `componentWillUnmount`).

С введением функциональных компонентов и React Hooks, управление жизненным циклом можно осуществлять через хуки, что значительно упрощает написание кода.

```
useEffect(() => {  
  // Код, выполняемый при монтировании и обновлении компонента  
  return () => {  
    // Код, выполняемый при размонтировании компонента  
  };  
}, [dependencies]);
```

Рисунок 6 – Hook UseEffect

1.2.3 CreateAsyncThunk и Store в React

В React, `createAsyncThunk` — это часть стандартной библиотеки `Redux Toolkit`, которая позволяет управлять асинхронными действиями (например, запросами к API) удобным и декларативным способом. Вместе с `store` они позволяют организовать состояние приложения и взаимодействие с ним.

`CreateAsyncThunk` — это функция, которая помогает создавать асинхронные действия (`thunk`) в `Redux`. Она принимает два аргумента: тип действия и функцию, которая будет выполнять асинхронный код. Функция автоматически обрабатывает три состояния асинхронного запроса: `pending`, `fulfilled` и `rejected`. Это упрощает управление состоянием загрузки и обработки ошибок. Внутри этой функции можно выполнять любые асинхронные операции,

такие как запросы к серверу, а затем возвращать результат. Пример использования `createAsyncThunk` на рисунке 7:

```
import { createAsyncThunk } from '@reduxjs/toolkit';

export const fetchEvents = createAsyncThunk(
  'events/fetchEvents',
  async () => {
    const response = await fetch('/api/events');
    return response.json();
  }
);
```

Рисунок 7 – Пример использования `createAsyncThunk`

В этом примере создается асинхронное действие `fetchEvents`, которое выполняет запрос к API для получения списка мероприятий. Redux Toolkit автоматически создает соответствующие действия для обработки состояний запроса.

Store в Redux — это объект, который хранит состояние приложения. Он управляет состоянием и предоставляет методы для доступа к нему, подписки на изменения и отправки действий. В Redux Toolkit создание Store стало проще благодаря функции `configureStore`, которая автоматически настраивает `middleware` и `DevTools`. Пример создания Store на рисунке 8:

```
import { configureStore } from '@reduxjs/toolkit';
import eventsReducer from './eventsSlice';

const store = configureStore({
  reducer: {
    events: eventsReducer,
  },
});

export default store;
```

Рисунок 8 – Пример создания Store

В этом примере создается Store, который использует редьюсер `eventsReducer` для управления состоянием мероприятий. `ConfigureStore` автоматически добавляет

необходимые middleware, такие как `redux-thunk`, что позволяет использовать асинхронные действия.

Использование `createAsyncThunk` и `Store` в `Redux Toolkit` значительно упрощает процесс управления состоянием в приложениях на `React`. Это позволяет разработчикам сосредоточиться на логике приложения, не беспокоясь о сложностях, связанных с асинхронными операциями и управлением состоянием.

1.2.4 Преимущества использования React

- **Производительность:** `React` использует виртуальный `DOM` для оптимизации обновлений интерфейса. Это снижает количество изменений в реальном `DOM`, что улучшает производительность приложения.
- **Компонентный подход:** компоненты упрощают повторное использование и тестирование кода, что делает проектирование более управляемым и модульным.
- **Широкое сообщество и экосистема:** `React` имеет огромное сообщество разработчиков и богатую экосистему библиотек и инструментов, что упрощает интеграцию и расширение функциональности.
- **Поддержка различных платформ:** `React` может использоваться для создания веб-приложений, мобильных приложений (`React Native`) и даже приложений для настольных ПК через `Electron`.

1.2.5 Выводы о React

`React` — это мощный инструмент для создания пользовательских интерфейсов, который предлагает множество возможностей для разработки высококачественных и отзывчивых приложений. Его концепции компонентов, состояния и свойств лежат в основе современного подхода к разработке, что делает `React` одной из самых популярных библиотек во фронтенд-разработке.

Совместно с другими технологиями, такими как Redux для управления состоянием или React Router для управления маршрутизации, React предоставляет мощный набор инструментов для создания сложных и масштабируемых веб-приложений.

1.3 Laravel: PHP-фреймворк для бэкенд-разработки

1.3.1 Введение в Laravel

Laravel — это популярный PHP-фреймворк, разработанный для создания веб-приложений с использованием архитектурного паттерна MVC (Model-View-Controller). Он был создан Тейлором Отвеллом и впервые выпущен в 2011 году. Laravel стремится сделать процесс разработки более удобным и продуктивным, предоставляя разработчикам элегантный синтаксис и мощные инструменты для решения распространенных задач [3].

1.3.2 Основные концепции и принципы Laravel

1.3.2.1 Архитектура MVC

Архитектура MVC разделяет приложение на три основных компонента[1]:

- Модель (Model): обрабатывает данные приложения и взаимодействует с базой данных. Она отвечает за бизнес-логику и представление данных.
- Представление (View): отвечает за отображение данных пользователю. Оно получает информацию из модели и отображает её в удобном для пользователя виде.
- Контроллер (Controller): обрабатывает входящие запросы и управляет взаимодействием между моделью и представлением. Он получает данные от модели и передает их представлению для отображения.

Такой подход упрощает поддержку и тестирование кода, позволяя разрабатывать приложение модулями.

1.3.2.2 Роутинг

Laravel предоставляет гибкую систему маршрутизации, позволяя разработчикам легко определять URL и связывать их с контроллерами или замыканиями. Это позволяет создавать чистые и понятные URL, что улучшает восприятие и SEO приложения.

Пример определения маршрута на рисунке 9:

```
Route::get('/users', [UserController::class, 'index']);
```

Рисунок 9 – Пример определения маршрута

1.3.2.3 ORM Eloquent

Eloquent — это встроенная ORM (Object-Relational Mapping) в Laravel, позволяющая работать с базой данных, используя объектно-ориентированный подход. Eloquent облегчает манипуляцию с записями базы данных, позволяя использовать методы для выполнения запросов без необходимости писать SQL-код напрямую.

Пример использования Eloquent на рисунке 10:

```
$users = User::where('active', 1)->get();
```

Рисунок 10 – Пример использования Eloquent

Eloquent поддерживает отношения между моделями (один-к-одному, один-ко-многим и многие-ко-многим), что позволяет легко управлять связями в приложении.

1.3.2.4 Миграции и сиды

Laravel предлагает удобный механизм миграций для управления структурой баз данных. Миграции позволяют контролировать изменения в базе данных с помощью версионирования, что упрощает работу в команде и развертывание приложения на разных окружениях.

Пример создания миграции на рисунке 11:

```
php artisan make:migration create_users_table
```

Рисунок 11 – Пример создания миграции

Сиды (seeder) позволяют заполнять базу данных тестовыми данными, что может быть полезно при разработке и тестировании.

```
php artisan make:seeder UsersTableSeeder
```

Рисунок 12 – Пример создания seeder

1.3.2.5 Контейнер внедрения зависимостей

Laravel использует контейнер внедрения зависимостей для управления зависимостями в приложении. Это позволяет облегчить тестирование и следовать принципам объектно-ориентированного программирования, таким как инверсия управления (IoC) [10].

Пример инъекции зависимостей в контроллере на рисунке 13:

```
public function __construct(UserRepository $userRepository)
{
    $this->userRepository = $userRepository;
}
```

Рисунок 13 – Пример инъекции зависимости

Это позволяет использовать интерфейсы и реализовывать тестирование, не зависимо от реальных реализаций классов.

1.3.3 Inertia.js в Laravel

Inertia.js в Laravel — это интеграция, которая позволяет разработчикам создавать современные одностраничные приложения (SPA), используя привычные серверные подходы и инструменты Laravel. Inertia.js служит связующим звеном между серверной и клиентской частями приложения, позволяя использовать Laravel для обработки маршрутов и бизнес-логики, а

также современные JavaScript-фреймворки (такие как Vue.js или React) для создания интерактивных пользовательских интерфейсов [9].

Основные концепции Inertia в Laravel:

- Серверный рендеринг: inertia позволяет Laravel рендерить страницы на сервере и отправлять их на клиент в виде JSON. Это упрощает интеграцию с существующими приложениями и позволяет использовать привычные методы работы с данными.
- Маршрутизация: можно использовать стандартные маршруты Laravel для определения URL-адресов вашего приложения. Inertia обрабатывает переходы между страницами, не перезагружая страницу, что обеспечивает плавный пользовательский опыт.
- Контроллеры: в контроллерах Laravel можно возвращать представления через Inertia, передавая данные, которые будут использоваться на клиенте.
- Компоненты: на стороне клиента создаются компоненты с использованием JavaScript-фреймворков, таких как Vue.js или React. Эти компоненты могут динамически обновляться в ответ на действия пользователя.

Преимущества Inertia в Laravel:

- Простота интеграции: inertia.js легко интегрируется в существующие приложения Laravel, не требуя значительной переработки инфраструктуры.
- UX без перезагрузки: пользователи могут взаимодействовать с приложением быстро и плавно, без необходимости постоянной перезагрузки страницы.
- Использование серверных инструментов: можно использовать все возможности Laravel, включая маршрутизацию, валидацию и работу с базами данных.
- Меньше инструментов: inertia позволяет избежать необходимости использования дополнительных инструментов, таких как API для взаимодействия между клиентом и сервером.

Inertia.js в Laravel предоставляет мощный инструмент для создания современных веб-приложений, комбинируя преимущества серверного рендеринга и интерактивного клиентского интерфейса. Это позволяет разработчикам сосредоточиться на логике приложения, используя привычные подходы к разработке, и создавать приложения, которые работают быстро и эффективно.

1.3.4 Преимущества использования Laravel

- **Элегантный синтаксис:** Laravel предлагает чистый и понятный API, что делает код более читаемым и поддерживаемым.
- **Безопасность:** Laravel предоставляет защиту от множества распространенных уязвимостей, включая SQL-инъекции, XSS и CSRF, что позволяет разработчикам сосредоточиться на функциональности, не беспокоясь о безопасности.
- **Сообщество и документация:** Laravel имеет активное сообщество и обширную документацию, что облегчает обучение и решение возникающих проблем.
- **Экосистема:** Laravel предлагает множество пакетов и инструментов (таких как Laravel Forge, Envoyer и Nova) для упрощения процессов разработки и развертывания.

1.3.5 Выводы о Laravel

Laravel — это мощный и удобный фреймворк для разработки веб-приложений на PHP. Его архитектурные подходы, такие как MVC и внедрение зависимостей, обеспечивают гибкость и модульность приложений. Удобные инструменты для работы с базами данных, маршрутизации и аутентификации делают процесс разработки более продуктивным. Благодаря широкой экосистеме, поддержке сообществом и элегантному синтаксису, Laravel стал одним из наиболее популярных выборов для веб-разработки, позволяя создавать сложные и масштабируемые приложения.

1.4 Tailwind CSS

Tailwind CSS — это утилитарная CSS-библиотека, которая позволяет разработчикам строить пользовательские интерфейсы быстро и эффективно. Отличительная особенность Tailwind заключается в том, что вместо использования готовых компонентов, как в большинстве фреймворков, он предоставляет множество классов, которые можно комбинировать для создания уникальных дизайнов.

Основные концепции Tailwind CSS:

- Утилитарный подход: Tailwind предлагает классы, которые соответствуют наиболее распространенным CSS-свойствам, таким как отступы, шрифты, цвета и размеры. Например, классы *p-4* и *text-center* позволяют легко управлять отступами и выравниванием текста без необходимости писать собственные CSS-правила.
- Конфигурация: Tailwind CSS основан на файле конфигурации (*tailwind.config.js*), что позволяет разработчикам настраивать тему, добавлять собственные цвета, шрифты и размеры, а также настраивать поведение утилит. Это дает возможность гибко адаптировать библиотеку под требования проекта.
- Псевдоклассы и адаптивность: в Tailwind используются псевдоклассы, которые позволяют легко управлять состоянием элементов. Например, класс *hover:bg-blue-500* изменит цвет фона на синий при наведении курсора. Также предусмотрена адаптивность - можете использовать классы, такие как *md:p-6* для задания отступов только на устройствах с «средними» экранами и выше.
- Оптимизация и производительность: Tailwind CSS включает в себя инструмент, называемый PurgeCSS, который удаляет неиспользуемые CSS-классы из конечного файла стилей, что позволяет значительно сократить размер CSS и улучшить производительность загружаемой страницы.

- Компоненты: несмотря на утилитарный подход, разработчики могут создавать собственные компоненты, комбинируя утилиты. Это облегчает повторное использование кода и улучшает структуру проекта.

Почему выбирают Tailwind CSS:

- Гибкость: пользователи могут создавать уникальный дизайн без ограничений, которые накладывают некоторые другие фреймворки.
- Скорость разработки: использование предопределенных классов значительно ускоряет процесс верстки.
- Персонализированная стилизация: настройка темы и стилизация делают ваш проект уникальным и красивым.
- Общая производительность: оптимизация CSS гарантирует, что страницы загружаются быстрее.
- Сообщество и экосистема: Tailwind имеет активное сообщество, множество плагинов и готовых шаблонов, что упрощает поиск решений и библиотек для расширения возможностей.

В целом, Tailwind CSS представляет собой мощный инструмент для веб-разработчиков, стремящихся к скорости и гибкости в создании пользовательских интерфейсов.

1.5 Архитектура веб-приложения

Разработка веб-приложения для отслеживания мероприятий будет включать в себя следующие ключевые компоненты:

- Фронтенд: реализованный на React, фронтенд будет отвечать за отображение информации о мероприятиях, взаимодействие с пользователем и отправку запросов на сервер.
- Стилизация: реализована с помощью фреймворка Tailwind CSS
- Бэкенд: реализованный на Laravel, бэкенд будет обрабатывать запросы от фронтенда, управлять базой данных и обеспечивать безопасность данных.

- База данных: для хранения информации о мероприятиях, пользователях и избранных событиях, будет использоваться реляционная база данных, такая как MySQL.

1.6 Выводы по теоретической части

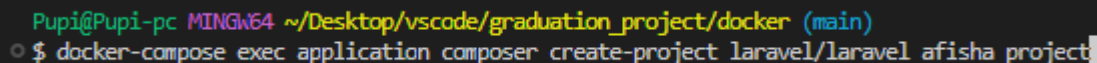
Разработка веб-приложения для отслеживания актуальных мероприятий с использованием React и Laravel представляет собой актуальную задачу, направленную на упрощение доступа к информации и улучшение пользовательского опыта. В данной теоретической части были рассмотрены ключевые технологии и архитектура приложения, что послужит основой для дальнейшей практической реализации проекта.

2 Практическая реализация

2.1 Принцип разворачивания среды разработки приложения.

Поднятие Laravel приложения с использованием Docker позволяет создать изолированную и воспроизводимую среду для разработки и развертывания. Ниже представлены шаги, позволившие развернуть Laravel приложение с Docker для создания веб-приложения для отслеживания актуальных мероприятий с возможностью добавления в избранное.

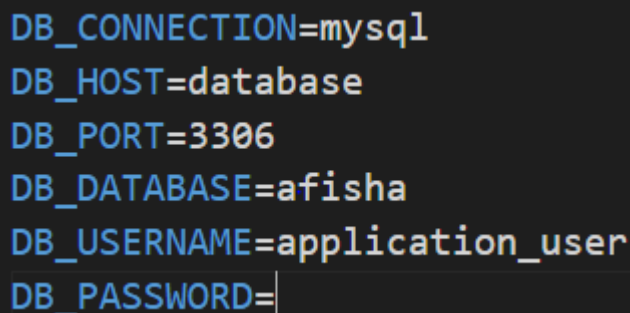
- Установка Docker
- Создание файлов Dockerfile для каждого из контейнеров. Dockerfile будут представлены в Приложении 1
- Создание файла docker-compose.yaml (см. Приложение 1)
- Запуск docker-среды
- Создание нового проекта Laravel. Для нового проекта используем следующую команду:



```
Pupi@Pupi-pc MINGW64 ~/Desktop/vscode/graduation_project/docker (main)
$ docker-compose exec application composer create-project laravel/laravel afisha_project
```

Рисунок 14 – Создание Laravel приложения

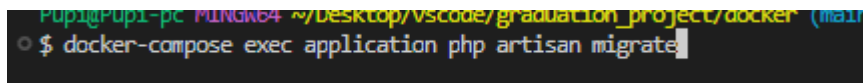
- Редактирование файла .env Laravel приложения, чтобы указать настройки подключения к базе данных:



```
DB_CONNECTION=mysql
DB_HOST=database
DB_PORT=3306
DB_DATABASE=afisha
DB_USERNAME=application_user
DB_PASSWORD=
```

Рисунок 15 – Настройки подключения к Базе данных

- Запуск миграций:



```
root@fup1-pe: /home/minkov4 ~/Desktop/vscode/graduation_project/docker (main)
$ docker-compose exec application php artisan migrate
```

Рисунок 16 – Запуск миграций приложения

- Теперь Laravel приложение доступно по адресу <http://afisha:8081/>, так как была внесена запись в файл hosts.

Использование Docker для поднятия Laravel приложения обеспечивает полную изоляцию и управление зависимостями, что делает процесс разработки более эффективным и предсказуемым. С помощью Docker и Docker Compose можно легко масштабировать приложение, добавляя новые сервисы, такие как Redis, Nginx или другие базы данных, по мере необходимости.

2.2 Подключение React к Laravel приложению

Для подключения библиотеки React к Laravel приложению была установлена библиотека Laravel Breeze [4].

Laravel Breeze предоставляет средства сборки для React и Vue через фронтенд-реализацию Inertia. Inertia позволяет создавать современные одностраничные приложения React и Vue с использованием классической маршрутизации на стороне сервера и контроллеров.

Inertia позволяет наслаждаться возможностями React и Vue на стороне фронтенда, объединенными с невероятной производительностью Laravel на стороне бэкенда и быстрой компиляцией Vite. Чтобы использовать стек Inertia, необходимо выбрать стек Vue или React при выполнении команды Artisan breeze:install.

При выборе стека фронтенда Vue или React, установщик Breeze также предложит определить, хотите ли вы Inertia SSR (серверный рендеринг) или поддержку TypeScript. После установки шаблонов Breeze также следует скомпилировать фронтенд-ресурсы вашего приложения:

```
php artisan breeze:install

php artisan migrate
npm install
npm run dev
```

Рисунок 17 – Установка Laravel Breeze

```
$ docker-compose exec application php artisan breeze:install

Which Breeze stack would you like to install? _____
React with Inertia

Would you like any optional features? _____
> ☐ Dark mode
☐ Inertia SSR
☐ TypeScript
☐ ESLint with Prettier

Use the space bar to select options.
```

Рисунок 18 – Установка Laravel Breeze в приложение с выбором React + Inertia

Затем можно перейти по адресу /login или /register приложения в веб-браузере. Все маршруты Breeze определены в файле routes/auth.php [5].

2.3 Функциональность приложения

2.3.1 Общая функциональность

Веб-приложение представляет собой сайт, для просмотра в веб-браузере, предлагающий пользователю ознакомиться со списком предстоящих мероприятий, в одном из предложенных городов РФ. В данной версии приложения предлагается на выбор 4 города: Санкт-Петербург, Москва, Краснодар и Сочи.

Приложение берет информацию с публичного API KudaGo [8] и выводит ее для пользователя в специально заданном порядке, так что пользователь будет видеть только предстоящие события, отсортированные по дате добавления на сайт <https://kudago.com/> [7]. Также пользователь может перейти на интересующую его карточку и посмотреть более подробную информацию о событии или перейти на сайт мероприятия.

При желании, пользователь может зарегистрироваться и авторизоваться в приложении, после чего ему будет доступен функционал добавления событий в избранное и просмотр избранных событий на отдельной странице веб-приложения. Главная страница приложения представлена на рисунке 19.

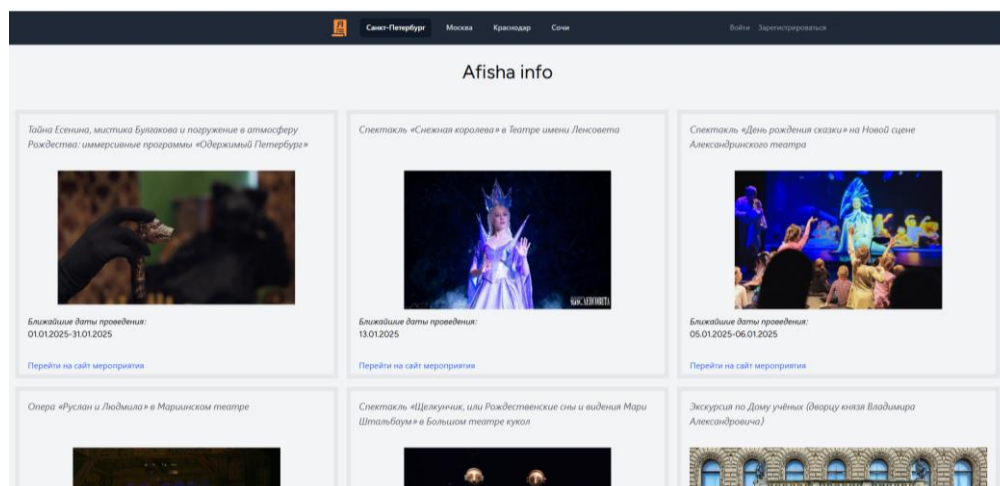


Рисунок 19 – Общий вид веб-приложения

Информация о датах выводится периодами, если событие проходит несколько дней, одним днем, если событие проходит только один день или появляется надпись *проводится постоянно*, если мероприятие долгосрочное. Также в выдаче пользователь не будет видеть прошедшие мероприятия, при этом в избранном они будут сохранены, а вместо даты будет надпись *событие прошло* (см. рисунок 20).

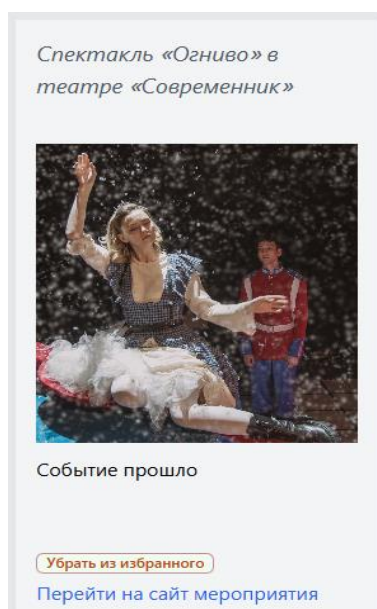


Рисунок 20 – Пример прошедшего события в избранном

2.3.2 Функциональность авторизации и добавления в избранное

Пользователь веб-приложения имеет возможность зарегистрироваться либо, авторизоваться, если уже имеет учетную запись. Для этого в правом верхнем углу приложения располагаются соответствующие кнопки.

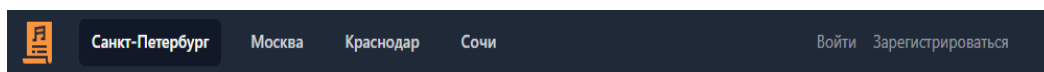


Рисунок 21 – Кнопки авторизации

При переходе на страницу создания учетной записи, пользователь попадает по адресу /register и наблюдает перед собой окно, представленное на рисунке 21.

A registration form on a light gray background. At the top center is a dark blue icon of a document with a pencil. Below it is a white rounded rectangle containing the form fields. The fields are labeled "Name", "Email", "Password", and "Confirm Password", each with a corresponding text input box. At the bottom right of the form is a dark blue button with the word "REGISTER" in white. To the left of the button is a blue link that says "Already registered?".

Рисунок 22 – Окно создания учетной записи

Пользователю предлагается ввести свое имя, адрес электронной почты, пароль и подтвердить пароль. После заполнения перечисленных полей, в случае корректно введенных данных, пользователь будет перенаправлен в свой личный кабинет на страницу избранного по адресу /dashboard (см. Рисунок 23). Также

пользователю будет отправлено сообщение на email, с благодарностью за регистрацию (см. Рисунок 24). За данный функционал отвечает класс *Welcome* и представление *welcome* в формате blade шаблона, листинг которых приведен в Приложении 2.

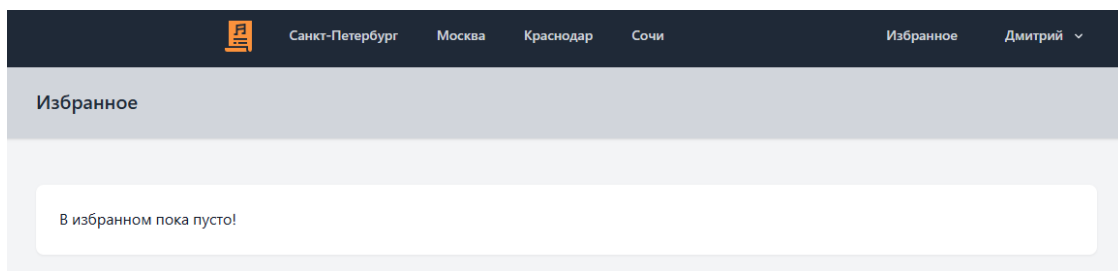


Рисунок 23 – Окно избранного зарегистрировавшегося пользователя

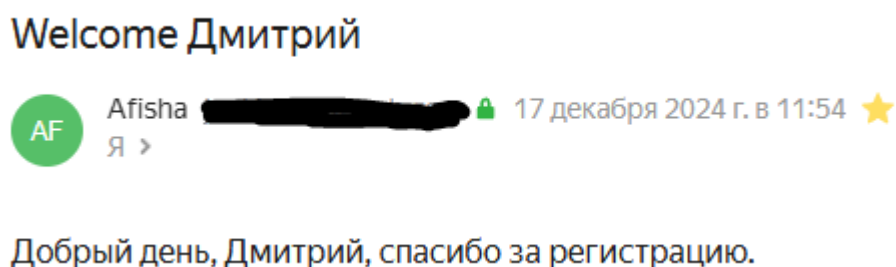


Рисунок 24 – Присланное письмо

При переходе на страницу авторизации, пользователь попадает по адресу /login и наблюдает перед собой окно, представленное на рисунке 25.

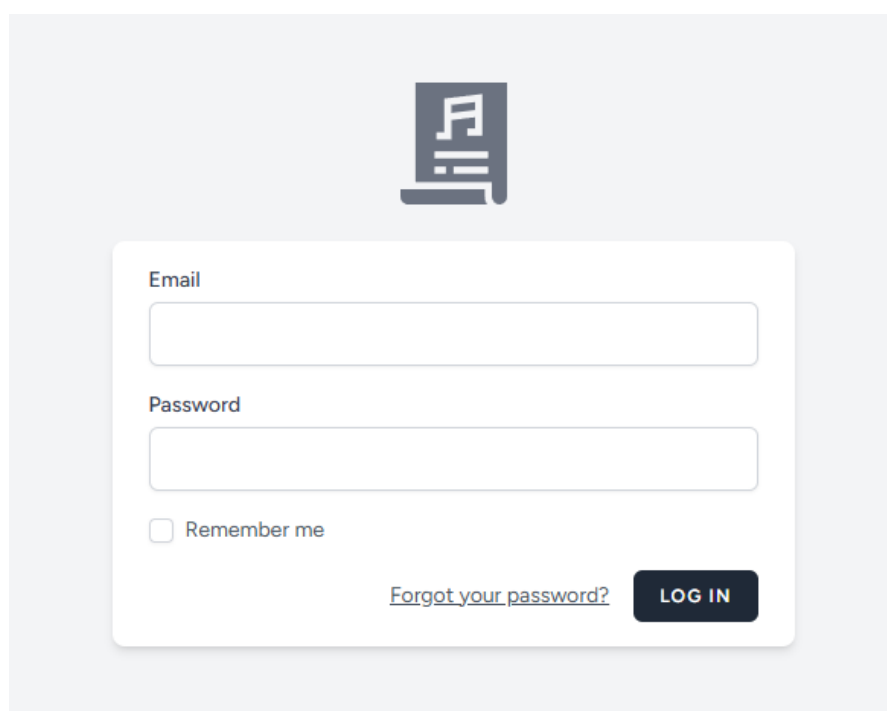


Рисунок 25 – Окно авторизации

Пользователю предлагается ввести адрес электронной почты и пароль. После заполнения перечисленных полей, в случае корректно введенных данных, пользователь будет перенаправлен в свой личный кабинет на страницу избранного по адресу /dashboard. В случае, если у пользователя есть сохранённые в избранном мероприятия, то они будут ему выведены (см. рисунок 26).

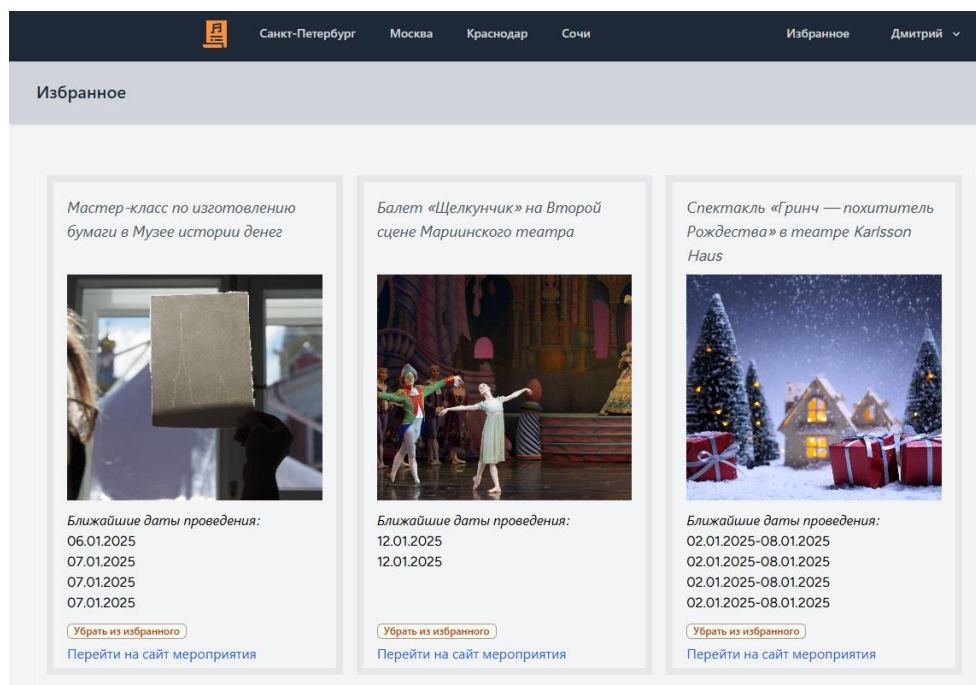


Рисунок 26 – Избранное авторизованного пользователя

Авторизованный пользователь, при просмотре событий в выбранном городе имеет возможность добавить понравившееся событие в избранное (см. рисунок 27).

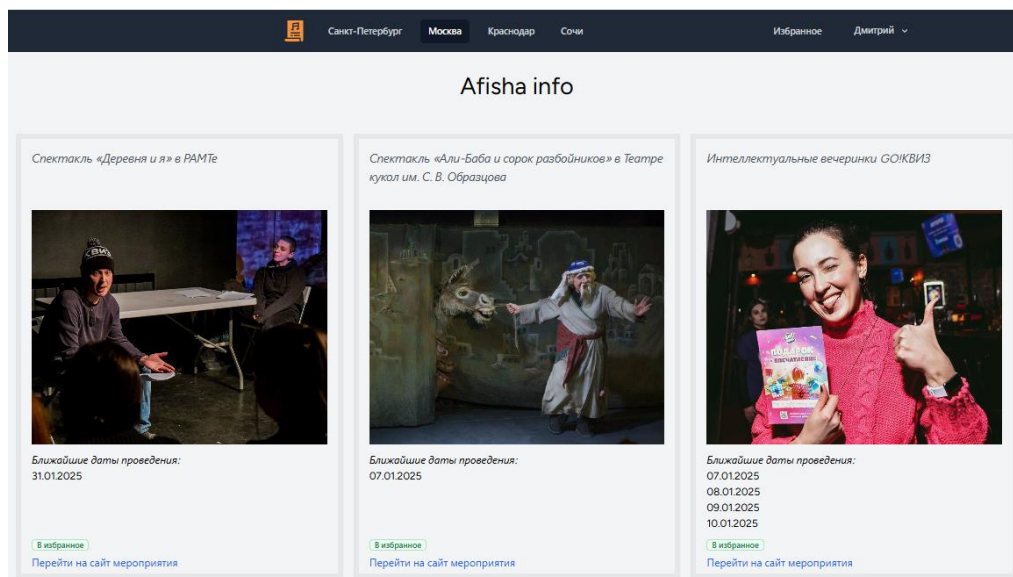


Рисунок 27 – Возможность добавления в избранное

Пользователь, при нажатии на свое имя в выпадающем меню может перейти на страницу редактирования информации о своём профиле, либо выйти из системы.

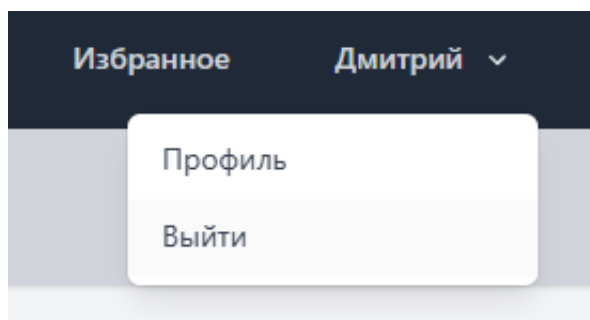


Рисунок 28 – Выпадающее меню профиля

При нажатии на кнопку “Профиль”, пользователь будет перенаправлен по адресу /profile на страницу редактирования информации, представленной на рисунке 29.

A screenshot of the Profile page. The page has a dark blue header with a logo, city names (Санкт-Петербург, Москва, Краснодар, Сочи), and user information (Избранное, Дмитрий). The main content area is white and contains three sections: "Profile Information" with fields for Name (Дмитрий) and Email, a "SAVE" button; "Update Password" with fields for Current Password, New Password, and Confirm Password, and a "SAVE" button; and "Delete Account" with a warning message and a red "DELETE ACCOUNT" button.

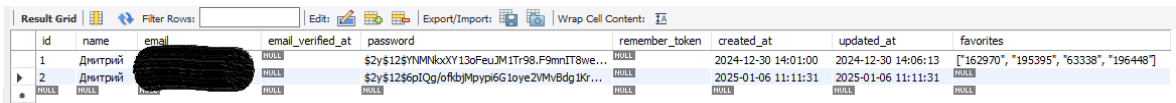
Рисунок 29 – Страница редактирования профиля

2.3.3 Хранение информации в базе данных

Для хранения информации о авторизационных данных пользователей, а также для отображения информации о избранных мероприятиях, приложение использует базу данных MySQL. Функциональность работы с базой данной реализована встроенными средствами работы с БД в Laravel – ORM-библиотекой Eloquent.

Laravel содержит ORM-библиотеку Eloquent, предоставляющую способ работы с базой данных, который часто удобнее обычного построителя запросов. При использовании Eloquent каждая таблица БД имеет соответствующую «Модель», которая используется для взаимодействия с этой таблицей. Помимо получения записей из таблицы БД, модели Eloquent также позволяют вставлять, обновлять и удалять записи из таблицы [6].

Laravel содержит в себе модель Users, с помощью которой идет взаимодействие с базой данных. Модель Users содержит в себе такие поля как id, name, email, email_verified_at, password, remember_token, created_at, updated_at, favorites. Листинг модели Users будет приведен в Приложении 2. На рисунке 27 представлена выборка из базы данных из таблицы users.



id	name	email	email_verified_at	password	remember_token	created_at	updated_at	favorites
1	Дмитрий	[REDACTED]	NULL	\$2y\$12\$YMN0xXY13oFeuJM1T98.F9mnIT8we...	NULL	2024-12-30 14:01:00	2024-12-30 14:06:13	["162970", "195395", "63338", "196448"]
2	Дмитрий	[REDACTED]	NULL	\$2y\$12\$6pIQg/ofkbjMypip6G1oye2VMvBdg1Kr...	NULL	2025-01-06 11:11:31	2025-01-06 11:11:31	NULL

Рисунок 30 – Выборка из таблицы users

Как можно видеть из представленного рисунка 30, информация в базе данных хранится в следующем виде (см. Таблицу 1):

Таблица 1. Поля таблицы users

Поле	Хранимый формат
ID пользователя	число
Имя	строка
email	валидированная строка
email верифицирован	timestamp
пароль	хешированная строка

токен идентификации	хешированная строка
пользователь создан	timestamp
пользователь обновлен	timestamp
избранное	массив

Для работы с таблицей users используются встроенные в Laravel контроллеры аутентификации. Для работы с полем favorites таблицы users используется кастомный контроллер FetchController, листинг которого будет представлен в Приложении 2.

2.3.4 Взаимодействие модулей программы

Приложение построено по принципу клиент-серверной архитектуры, где Laravel выполняет роль бэкенда, обрабатывающего запросы от клиента и предоставляющего данные через API, а React отвечает за отображение пользовательского интерфейса.

Взаимодействие между модулями программы реализовано с помощью createAsyncThunk, а также отрисовки представлений с использованием Inertia, которые были рассмотрены в теоретической части работы. Основная точка входа маршрутов приложения находится в файле routes/web.php (см. Приложение 2). В этом файле определяются маршруты, которые будут обрабатывать запросы и возвращать представления с использованием Inertia. Когда пользователь открывает сайт с приложением, то попадает на главную страницу. Происходит перенаправление на маршрут /spb и вызывается представление ShowAfishaInfo.jsx (см. Приложение 2), которое отрисовывает информацию по городу Санкт-Петербургу. Когда пользователь нажимает на другой город, делается запрос на бекенд часть в контроллер FetchController.php (см. Приложение 2), который подгружает информацию по требуемому городу с внешнего источника и отдает результат во фронтенд часть в представление ShowAfishaInfo.jsx.

Когда пользователь нажимает на карточку события, вызывается другое представление FetchEvent.jsx (см. Приложение 2) которое с помощью бекенд части также делает запрос к стороннему источнику и отдает уточненную информацию по событию в представление EachEvent.jsx (см. Приложение 2), которое отрисовывает информацию (см. рисунок 31).

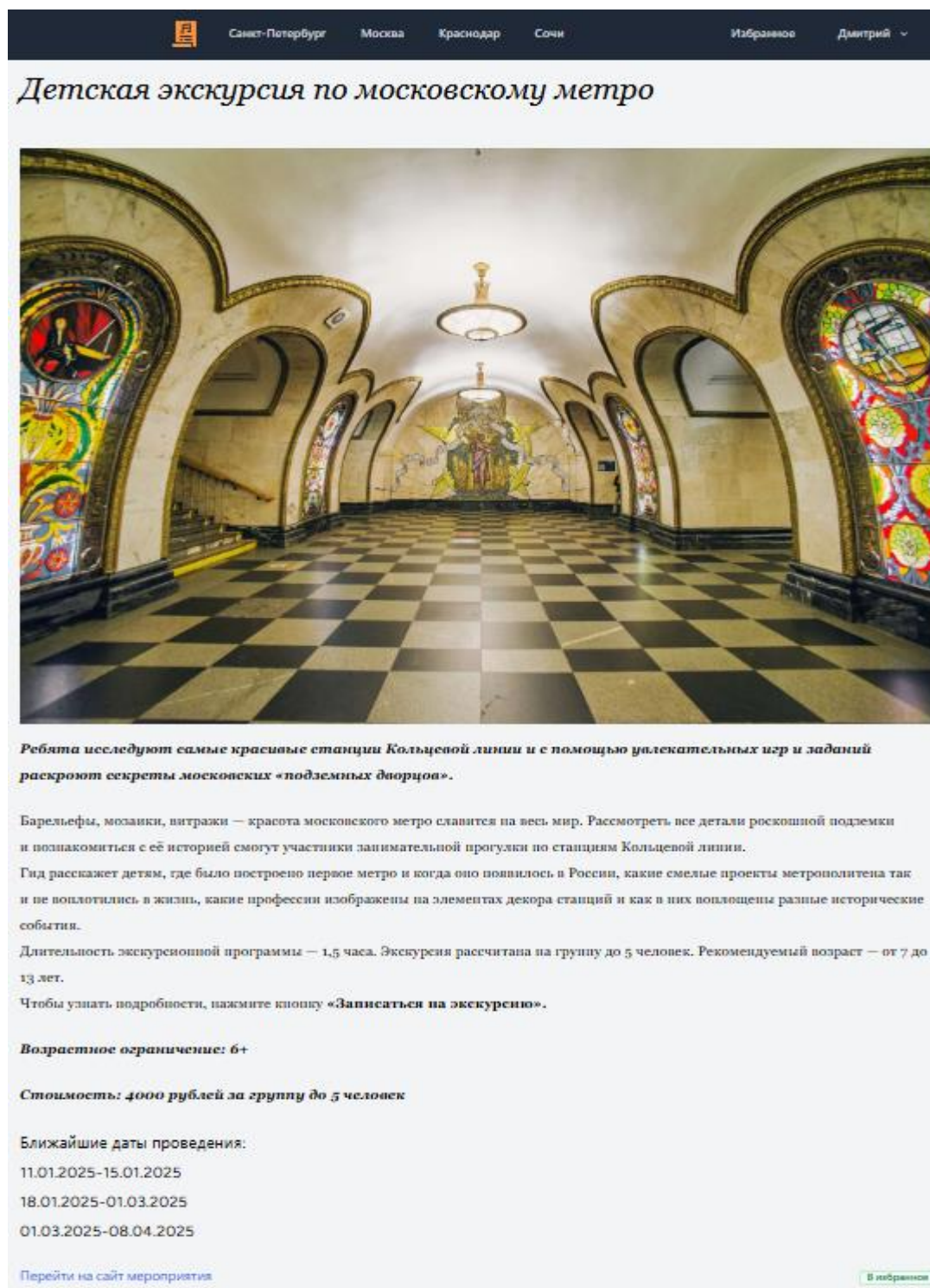


Рисунок 31 – Детализация события

Каждая страница включает в себя шапку сайта, который вынесен в отдельный функциональный компонент Header.jsx (см. Приложение 2).

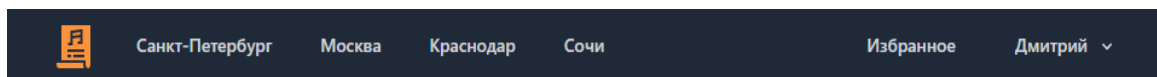


Рисунок 32 – Шапка сайта

Также в отдельный компонент Pagation.jsx (см. Приложение 2) вынесены элементы пагинации, необходимые на некоторых страницах приложения.

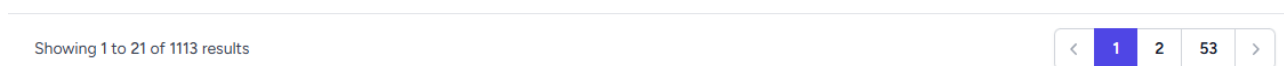


Рисунок 33 – Элементы пагинации

Пагинация в левом углу показывает количество отображаемых событий на странице и общее количество событий, а в правом углу расположено меню перемещения по страницам.

Каждая карточка события EachCard.jsx (см. Приложение 2) кнопка добавления в избранное FavoritesButton.jsx (см. Приложение 2), детализация события EachEvent.jsx (см. Приложение 2), также являются отдельными функциональными компонентами. Все остальные компоненты, не являющиеся стандартными при разворачивании приложения также представлены в Приложении 2.

При нажатии на кнопки добавления и удаления из избранного делается запрос в бекенд часть в контроллер FetchController.php, в котором вызывается соответствующий метод и производятся действия с базой данных. В свою очередь, в этот момент, фронтенд меняет дизайн и надпись внутри кнопки. При перезагрузке страницы данные о наличии карточки в избранном берутся всегда из базы данных.

2.4 Выводы о практической реализации

В результате практической реализации было разработано полноценное полностью работоспособное веб-приложение, позволяющее пользователям отслеживать актуальные мероприятия и управлять списком избранных. Использование Laravel и React обеспечило эффективное разделение логики приложения, улучшая как производительность, так и удобство работы. Приложение имеет хороший потенциал для развития, модернизации и масштабирования, а также для внедрения новой функциональности.

Таким образом, данное приложение не только демонстрирует применения современных технологий, но и предоставляет пользователям удобный интерфейс и функциональные возможности для отслеживания мероприятий.

Заключение

В ходе разработки веб-приложения для отслеживания актуальных мероприятий с возможностью добавления в избранное, было реализовано эффективное решение, отвечающее всем заявленным требованиям и функциональным характеристикам. Использование стека технологий Laravel и React обеспечило стабильность и высокую отзывчивость системы, а также позволило создать современный и интуитивно понятный интерфейс для конечного пользователя.

В процессе работы над проектом были применены лучшие практики разработки, включая создание модульной структуры, оптимизацию производительности и быструю скорость загрузки страниц. Также уделено внимание вопросам безопасности данных и защите личной информации пользователей, что является важным аспектом в современных веб-приложениях.

Созданное веб-приложение позволяет пользователям легко находить и отслеживать различные мероприятия, а главной особенностью разработанного приложения, является возможность добавления мероприятий в избранное, что позволяет пользователям быстро получать доступ к интересующей информации. Эта функция была реализована с учетом современных подходов к пользовательскому интерфейсу, что значительно улучшает пользовательский опыт. Такой подход повышает вовлеченность пользователей и делает процесс поиска информации более удобным и персонализированным.

Проведённые тестирования показали высокую степень стабильности и функционирования приложения в условиях реальной эксплуатации. Полученные результаты подтверждают правильность выбранных архитектурных подходов и технологий, на основе которых было реализовано веб-приложение.

В дальнейшем планируется продолжить развивать проект и внедрить в него такие функции как:

- Расширить список предлагаемых на выбор пользователю городов, вынести список в селектор, с возможностью поиска по первым введенным буквам.
- Показ списка событий дня.
- Добавление функциональности просмотра комментариев к событию.
- Добавить функции фильтрации мероприятий по разным критериям, а также функции выбора параметров отображения на странице.
- Расширить функции системы уведомлений.
- Добавить возможность выбора конкретной страницы в пагинации.

Также планируется рассмотреть возможность интеграции с различными социальными сетями.

В заключение, проект продемонстрировал не только пользовательское удобство, но и перспективность использования современных технологий в разработке веб-приложений.

Таким образом, выполненная работа представляет собой достойный вклад в сферу веб-разработки и может быть использована как основа для дальнейших исследований и улучшений.

Список используемой литературы

1. Kott, N. (2019). Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. O'Reilly Media. 478 pages.
2. Sass, K. (2020). Web Development with React and Laravel: Building applications using RESTful APIs. Packt Publishing. 289 pages.
3. Официальная документация фреймворка Laravel: [Электронный ресурс] — URL: <https://laravel.com/docs/11.x/> (дата обращения: 6.11.2024).
4. Инструкция по установке и настройке Laravel Breeze с сайта официальной документации фреймворка Laravel: [Электронный ресурс] — URL: <https://laravel.com/docs/11.x/starter-kits> (дата обращения: 15.11.2024).
5. Инструкция по установке и настройке Laravel Breeze с сайта официальной документации фреймворка Laravel на русском языке: [Электронный ресурс] — URL: <https://laravel.su/docs/11.x/starter-kits> (дата обращения: 15.11.2024).
6. Модель Eloquent с сайта официальной документации фреймворка Laravel на русском языке: [Электронный ресурс] — URL: <https://laravel.su/docs/11.x/eloquent> (дата обращения: 16.11.2024).
7. Сайт по поиску мероприятий в городах РФ: [Электронный ресурс] — URL: <https://kudago.com/> (дата обращения: 16.11.2024).
8. API для доступа к информации сайта по поиску мероприятий в городах РФ: [Электронный ресурс] — URL: <https://docs.kudago.com/api/> (дата обращения: 16.11.2024).
9. Официальная документация Inertia.js: [Электронный ресурс] — URL: <https://inertiajs.com> (дата обращения: 20.11.2024).
10. Трепачёв Дмитрий. Онлайн учебник по веб-разработке: [Электронный ресурс] — <https://code.mu/ru/> (дата обращения: 25.11.2024).

Приложения

Расположение практической части дипломной работы:

https://github.com/aleksdima01/graduation_project.git

Приложение 1. Конфигурация Docker

Листинг 1 – Dockerfile php-fpm

```
FROM php:8.2-fpm

COPY ./php.ini /usr/local/etc/php/conf.d/php-custom.ini

RUN curl -sS https://getcomposer.org/installer | php -- --install-  
dir=/usr/local/bin --filename=composer

ENV COMPOSER_ALLOW_SUPERUSER 1

RUN apt-get update && apt-get install -y zip \  
unzip \  
build-essential \  
libpng-dev \  
libjpeg62-turbo-dev \  
libfreetype6-dev \  
locales \  
zip \  
jpegoptim optipng pngquant gifsicle \  
vim \  
unzip \  
git \  
curl
```

```
RUN docker-php-ext-install mysqli pdo pdo_mysql

RUN curl -fsSL https://deb.nodesource.com/setup_22.x | bash
RUN apt-get install -y nodejs

# Add user for laravel application
WORKDIR /data/afisha
RUN groupadd -g 1000 www
RUN useradd -u 1000 -ms /bin/bash -g www www
# RUN chown -R www:www ../mysite.local

USER www

#VOLUME /data/mysite.local

EXPOSE 9000

CMD ["php-fpm"]
```

Листинг 2 – Dockerfile nginx

```
FROM nginx:latest

COPY ./hosts/afisha.conf /etc/nginx/conf.d/afisha.conf

WORKDIR /data/afisha

VOLUME /data/afisha

EXPOSE 80
```

```
CMD ["nginx", "-g", "daemon off;"]
```

Листинг 3 – Config nginx

```
server {  
    # указываем 80 порт для соединения  
    listen 80;  
  
    # нужно указать, какому доменному имени принадлежит конфиг  
    server_name afisha;  
  
    # задаём корневую директорию  
    root /data/afisha/public;  
  
    # стартовый файл  
    index index.php index.html;  
  
    location ~* \.(jpg|jpeg|gif|css|png|js|ico|html)$ {  
        access_log off;  
        expires max;  
    }  
  
    location / {  
        try_files $uri $uri/ /index.php?$query_string;  
    }  
  
    location ~* \.php$ {  
        try_files $uri = 404;  
        fastcgi_split_path_info ^(.+\.php)(/.+)\$;  
        fastcgi_pass application:9000;  
        #fastcgi_pass unix:/var/run/php8.2-fpm.sock;
```

```

    fastcgi_index index.php;

    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    include fastcgi_params;
}

# ошибки
# fastcgi_intercept_errors on;
# error_page 404 403 500 503 /404.html;
}

```

Листинг 4 – Docker-compose.yaml

```

#в этом блоке описываем контейнеры, которые будут запускаться
services:
  #контейнер с Nginx
  nginx:
    build:
      context: ./nginx
      dockerfile: Dockerfile
    image: myapp/nginx
    container_name: webserver
    # проброс портов
    ports:
      - "8081:80"
    volumes:
      - ../afisha_project:/data/afisha
    networks:
      - app-network

```

```

#Контейнер с PHP-FPM, назовём его application
application:
  build:
    context: ./fpm
    dockerfile: Dockerfile
  image: myapp/php # имя будущего образа
  container_name: application # имя контейнера после запуска
  #tty: true
  ports:
    - "5173:5173"
  volumes:
    - ../afisha_project:/data/afisha
  networks:
    - app-network

#Контейнер с БД
database:
  image: mysql:5.7
  container_name: database # имя контейнера после запуска
  environment:
    MYSQL_DATABASE: ${DB_NAME} # имя БД
    MYSQL_USER: ${DB_USER} # имя пользователя, с которым будет
подключаться
    MYSQL_PASSWORD: ${DB_PASSWORD} # пароль для пользователя
    MYSQL_ROOT_PASSWORD: ${DB_ROOT_PASSWORD} # администраторский пароль
  ports:
    - "3307:3306"
  # создаю для контейнеров внутреннюю сеть
  networks:
    - app-network

```

```
#Docker Networks

networks:

  app-network:

    driver: bridge
```

Приложение 2. Листинг модулей программы

Листинг 1 – Контроллер FetchController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;

class FetchController extends Controller
{
    public function getFetchInfo(Request $request)
    {
        $response = Http::get("https://kudago.com/public-
api/v1.4/events/?lang=&fields=id,dates,publication_date,title,short_title
,place,location,images,site_url&expand=&order_by=-
publication_date&text_format=&ids=&page={$request-
>page}&page_size=21&location={$request->location}&actual_since={$request-
>actual_since}");
        return $response->json();
    }
    public function fetchEachEvent(Request $request)
    {
        $response = Http::get("https://kudago.com/public-
api/v1.4/events/{$request->id}/?lang=&fields=");
        return $response->json();
    }
    public function fetchFavorites(Request $request)
    {
        $user = User::findOrFail($request->user);
        $favorites = $user->favorites;
        $stringFavorites = implode(",", $favorites);
        $response = Http::get("https://kudago.com/public-
api/v1.4/events/?lang=&fields=id,dates,publication_date,title,short_title
,place,location,images,site_url&ids={$stringFavorites}");
```



```

        return $response->json();
    }
    public function saveFavorites(Request $request)
    {
        $user = User::findOrFail($request->user);
        if (collect($user->favorites)->contains($request->favoriteid)) {
            return "Элемент уже в избранном";
        } else {
            $user->favorites = (collect($user->favorites)->push($request->favoriteid));
            $user->save();
            return $user->favorites;
        }
    }
    public function deleteFavorites(Request $request)
    {
        $user = User::findOrFail($request->user);
        $favorites = $user->favorites;
        $newFavorites = collect($favorites)->filter(function ($value) use ($request) {
            return $value !== $request->favoriteid;
        });
        $user->favorites = $newFavorites;
        $user->save();
        return $user->favorites;
    }
}

```

Листинг 2 – Модель User.php

```

<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    /** @use HasFactory<\Database\Factories\UserFactory> */
    use HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var list<string>
     */
    protected $fillable = [

```

```

        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var list<string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * Get the attributes that should be cast.
     *
     * @return array<string, string>
     */
    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
            'favorites' => 'array',
        ];
    }
}

```

Листинг 3 – Класс Welcome.php

```

<?php

namespace App\Mail;

use App\Models\User;
use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Mail\Mailable;
use Illuminate\Mail\Mailables\Content;
use Illuminate\Mail\Mailables\Envelope;
use Illuminate\Queue\SerializesModels;

class Welcome extends Mailable
{
    use Queueable, SerializesModels;
}

```

```

public User $user;
/**
 * Create a new message instance.
 */
public function __construct(User $user)
{
    $this->user = $user;
}

/**
 * Get the message envelope.
 */
public function envelope(): Envelope
{
    return new Envelope(
        subject: 'Welcome' . ' ' . $this->user->name,
    );
}

/**
 * Get the message content definition.
 */
public function content(): Content
{
    return new Content(
        view: 'mails.welcome',
    );
}

/**
 * Get the attachments for the message.
 *
 * @return array<int, \Illuminate\Mail\Mailables\Attachment>
 */
public function attachments(): array
{
    return [];
}
}

```

Листинг 4 – Представление welcome.blade.php

```
<body>
  <div>Добрый день, {{ $user->name }}, спасибо за регистрацию.</div>
</body>
```

Листинг 5 – store.jsx

```
import { configureStore } from "@reduxjs/toolkit";
import showAfishaReducer from './afishaReducer'
import showEventReducer from './eventReducer'
import showFavoritesReducer from './favoritesReducer'

const store = configureStore({
  reducer: {
    afisha: showAfishaReducer,
    event: showEventReducer,
    favorites: showFavoritesReducer,
  },
});

export default store;
```

Листинг 6 – favoritesReducer.jsx

```
import { createAsyncThunk, createSlice } from "@reduxjs/toolkit";

export const fetchFavorites = createAsyncThunk(
  'favorites/fetchFavorites',
  async (data, thunkApi) => {
    try {
      const response = await
fetch(`http://127.0.0.1:8081/api/fetchfavorites/?user=${data.userId}`, {
        method: "GET",
      });
      if (!response.ok) {
        throw new Error("Something go wrong");
      }
      return await response.json();
    } catch (error) {
      thunkApi.rejectWithValue(error.message)
    }
  })

const initialState = {
  favorites: [],
```

```

    loading: false,
    error: null,
  };

const favoritesSlice = createSlice({
  name: "favorites",
  initialState,
  reducers: {},
  extraReducers: (builder) => {
    builder
      .addCase(fetchFavorites.pending, (state) => {
        state.loading = true;
        state.error = null;
      })
      .addCase(fetchFavorites.fulfilled, (state, action) => {
        state.loading = false;
        state.favorites = action.payload;
      })
      .addCase(fetchFavorites.rejected, (state, action) => {
        state.loading = false;
        state.error = action.payload;
      })
  }
});

export default favoritesSlice.reducer;

```

Листинг 7 – eventReducer.jsx

```

import { createAsyncThunk, createSlice } from "@reduxjs/toolkit";

export const fetchEvent = createAsyncThunk(
  'event/fetchEvent',
  async (data, thunkApi) => {
    try {
      const response = await
fetch(`http://127.0.0.1:8081/api/fetcheachevent/?id=${data.id}&lang=&fiel
ds=&expand=images`, {
        method: "GET",
      });
      if (!response.ok) {
        throw new Error("Something go wrong");
      }
    }
  }
);

```

```

        return await response.json();
    } catch (error) {
        thunkApi.rejectWithValue(error.message)
    }
})

const initialState = {
  event: {},
  loading: false,
  error: null,
  // city: 'spb',
};

const eventSlice = createSlice({
  name: "event",
  initialState,
  reducers: {},
  extraReducers: (builder) => {
    builder
      .addCase(fetchEvent.pending, (state) => {
        state.loading = true;
        state.error = null;
        state.event = [];
      })
      .addCase(fetchEvent.fulfilled, (state, action) => {
        state.loading = false;
        state.event = action.payload;
        // state.city = action.payload.results[0].location.slug
      })
      .addCase(fetchEvent.rejected, (state, action) => {
        state.loading = false;
        state.error = action.payload;
      })
  }
})

export default eventSlice.reducer;

```

Листинг 8 – deleteFromFavorites.jsx

```

import { createAsyncThunk, } from "@reduxjs/toolkit";

export const fetchDeleteFromFavorites = createAsyncThunk(

```

```

    'favorites/fetchDeleteFromFavorites',
    async (data, thunkApi) => {
      try {
        const response = await
fetch(`http://127.0.0.1:8081/api/deletefavorites/?user=${data.user}&favor
iteid=${data.id}`, {
          method: "GET",
        });
        if (!response.ok) {
          throw new Error("Something go wrong");
        }
        return await response.json();
      } catch (error) {
        thunkApi.rejectWithValue(error.message)
      }
    })

```

Листинг 9 – afishaReducer.jsx

```

import { createAsyncThunk, createSlice } from "@reduxjs/toolkit";

export const fetchAfisha = createAsyncThunk(
  'afisha/fetchAfisha',
  async (data, thunkApi) => {
    //const { page, city } = data
    try {
      const response = await
fetch(`http://127.0.0.1:8081/api/getfetchinfo?lang=&fields=id,dates,publi
cation_date,title,short_title,place,location,images,site_url&expand=&orde
r_by=-
publication_date&text_format=&ids=&page=${data.page}&page_size=21&locatio
n=${data.city}&actual_since=${Date.now() / 1000}`, {
        method: "GET",
      });
      if (!response.ok) {
        throw new Error("Something go wrong");
      }
      return await response.json();
    } catch (error) {
      thunkApi.rejectWithValue(error.message)
    }
  })

```

```

const initialState = {
  afisha: [],
  loading: false,
  error: null,
  // city: 'spb',
};

const afishaSlice = createSlice({
  name: "afisha",
  initialState,
  reducers: {},
  extraReducers: (builder) => {
    builder
      .addCase(fetchAfisha.pending, (state) => {
        state.loading = true;
        state.error = null;
      })
      .addCase(fetchAfisha.fulfilled, (state, action) => {
        state.loading = false;
        state.afisha = action.payload;
        state.city = action.payload.results[0].location.slug;
      })
      .addCase(fetchAfisha.rejected, (state, action) => {
        state.loading = false;
        state.error = action.payload;
      })
  }
});

export default afishaSlice.reducer;

```

Листинг 10 – addToFavorites.jsx

```

import { createAsyncThunk } from "@reduxjs/toolkit";

export const fetchAddToFavorites = createAsyncThunk(
  'favorites/fetchAddToFavorites',
  async (data, thunkApi) => {
    try {
      const response = await
fetch(`http://127.0.0.1:8081/api/savefavorites/?user=${data.user}&favorit
eid=${data.id}`, {
        method: "GET",

```



```

    });
    if (!response.ok) {
      throw new Error("Something go wrong");
    }
    return await response.json();
  } catch (error) {
    thunkApi.rejectWithValue(error.message)
  }
})

```

Листинг 11 – ShowAfishaInfo.jsx

```

import { useSelector } from "react-redux";
import EachCard from "../Components/Custom/EachCard";
import { useDispatch } from "react-redux";
import { fetchAfisha } from '../Store/afishaReducer';
import { useEffect } from 'react';
import Pagination from "../Components/Custom/Pagination";
import Header from "@/Layouts/Header";
import { Head, Link } from '@inertiajs/react';
import FavoritesButton from "@/Components/Custom/FavoritesButton";

export default function ShowAfishaInfo({ city }) {
  const { afisha, loading, error } = useSelector((state) =>
state.afisha);
  const dispatch = useDispatch();
  useEffect(() => {
    dispatch(fetchAfisha({ page: 1, city: city }))
  }, [dispatch, city])

  return (
    <>
      <Header />
      <Head title="Welcome" />
      <div className="bg-gray-100 min-h-screen">
        <h2 className="text-4xl mb-10 pt-8 text-center">Afisha
info</h2>
        {loading && <div role="status" className="flex items-
center justify-center">
          <svg aria-hidden="true" className="w-8 h-8 text-gray-
200 animate-spin dark:text-gray-600 fill-blue-600" viewBox="0 0 100 101"
fill="none" xmlns="http://www.w3.org/2000/svg">
            <path d="M100 50.5908C100 78.2051 77.6142 100.591
50 100.591C22.3858 100.591 0 78.2051 0 50.5908C0 22.9766 22.3858 0.59082

```

```

50 0.59082C77.6142 0.59082 100 22.9766 100 50.5908ZM9.08144
50.5908C9.08144 73.1895 27.4013 91.5094 50 91.5094C72.5987 91.5094
90.9186 73.1895 90.9186 50.5908C90.9186 27.9921 72.5987 9.67226 50
9.67226C27.4013 9.67226 9.08144 27.9921 9.08144 50.5908Z"
fill="currentColor" />
      <path d="M93.9676 39.0409C96.393 38.4038 97.8624
35.9116 97.0079 33.5539C95.2932 28.8227 92.871 24.3692 89.8167
20.348C85.8452 15.1192 80.8826 10.7238 75.2124 7.41289C69.5422 4.10194
63.2754 1.94025 56.7698 1.05124C51.7666 0.367541 46.6976 0.446843 41.7345
1.27873C39.2613 1.69328 37.813 4.19778 38.4501 6.62326C39.0873 9.04874
41.5694 10.4717 44.0505 10.1071C47.8511 9.54855 51.7191 9.52689 55.5402
10.0491C60.8642 10.7766 65.9928 12.5457 70.6331 15.2552C75.2735 17.9648
79.3347 21.5619 82.5849 25.841C84.9175 28.9121 86.7997 32.2913 88.1811
35.8758C89.083 38.2158 91.5421 39.6781 93.9676 39.0409Z"
fill="currentFill" />
    </svg>
    <span className="sr-only">Loading...</span>
  </div>}
  {error && <p>Ошибка {error}</p>}
  {Object.keys(afisha).length !== 0 ? <div className="grid
gap-4 grid-cols-3 p-3">
    {
      afisha.results.map(el => (
        <div className="border-8 relative"
key={el.id}>

          <Link href={el.id}
            className=" h-full w-full flex
justify-center"

            >
              <EachCard event={el} />
            </Link>
            <div >
              <FavoritesButton id={el.id}
                addClasses={'absolute bottom-8
px-4 '}

              />
            </div>

            <a href={el.site_url} className="px-4
absolute bottom-1 text-blue-600 visited:text-purple-600 pointer-events-
auto hover:text-orange-600 text-[7px] sm:text-sm/[10px] lg:text-
base">Перейти на сайт мероприятия</a>

```

```

        </div>
      ))
    }
  </div>
  : null}
</div >
<footer className={loading ? "hidden" : ''}>
  <Pagination city={city} />
</footer>
</>
);
}

```

Листинг 12 – FetchEvent.jsx

```

import EachEvent from "@/Components/Custom/EachEvent";
import { fetchEvent } from "@/Store/eventReducer";
import { useEffect } from "react";
import { useDispatch } from "react-redux";

const FetchEvent = ({ id }) => {
  const dispatch = useDispatch();
  useEffect(() => {
    dispatch(fetchEvent({ id: id }))
  }, [dispatch, id])
  return (
    <>
      <EachEvent />
    </>
  )
}
export default FetchEvent

```

Листинг 13 – Dashboard.jsx

```

import EachCard from '@/Components/Custom/EachCard';
import FavoritesButton from '@/Components/Custom/FavoritesButton';
import AuthenticatedLayout from '@/Layouts/AuthenticatedLayout';
import Header from '@/Layouts/Header';
import { fetchFavorites } from '@/Store/favoritesReducer';
import { Head, Link, usePage } from '@inertiajs/react';
import { useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';

```

```

export default function Dashboard() {
  const page = usePage();
  const { favorites, loading, error } = useSelector((state) =>
state.favorites);
  const dispatch = useDispatch();
  useEffect(() => {
    dispatch(fetchFavorites({ userId: page.props.auth.user.id }))
  }, [dispatch])
  return (
    <>
      <Head title="Избранное" />
      <Header />
      <AuthenticatedLayout
        header={
          <h2 className="text-xl font-semibold leading-tight
text-gray-800">
            Избранное
          </h2>
        }
      >
        <div className="py-12 ">
          <div className="mx-auto max-w-7xl sm:px-6 lg:px-8 ">
            {loading && <div role="status" className="flex
items-center justify-center bg-gray-100 shadow-none">
              <svg aria-hidden="true" className="w-8 h-8
text-gray-200 animate-spin dark:text-gray-600 fill-blue-600" viewBox="0 0
100 101" fill="none" xmlns="http://www.w3.org/2000/svg">
                <path d="M100 50.5908C100 78.2051 77.6142
100.591 50 100.591C22.3858 100.591 0 78.2051 0 50.5908C0 22.9766 22.3858
0.59082 50 0.59082C77.6142 0.59082 100 22.9766 100 50.5908ZM9.08144
50.5908C9.08144 73.1895 27.4013 91.5094 50 91.5094C72.5987 91.5094
90.9186 73.1895 90.9186 50.5908C90.9186 27.9921 72.5987 9.67226 50
9.67226C27.4013 9.67226 9.08144 27.9921 9.08144 50.5908Z"
fill="currentColor" />
                <path d="M93.9676 39.0409C96.393 38.4038
97.8624 35.9116 97.0079 33.5539C95.2932 28.8227 92.871 24.3692 89.8167
20.348C85.8452 15.1192 80.8826 10.7238 75.2124 7.41289C69.5422 4.10194
63.2754 1.94025 56.7698 1.05124C51.7666 0.367541 46.6976 0.446843 41.7345
1.27873C39.2613 1.69328 37.813 4.19778 38.4501 6.62326C39.0873 9.04874
41.5694 10.4717 44.0505 10.1071C47.8511 9.54855 51.7191 9.52689 55.5402
10.0491C60.8642 10.7766 65.9928 12.5457 70.6331 15.2552C75.2735 17.9648
79.3347 21.5619 82.5849 25.841C84.9175 28.9121 86.7997 32.2913 88.1811

```

```

35.8758C89.083 38.2158 91.5421 39.6781 93.9676 39.0409Z"
fill="currentFill" />
        </svg>
        <span className="sr-only">Loading...</span>
    </div>}
    {error && <p>Ошибка {error}</p>}
    <div className="overflow-hidden bg-white shadow-
sm sm:rounded-lg bg-gray-100">
        {(page.props.auth.user.favorites) ?
            <div className="p-6 text-gray-900">
                В избранном пока пусто!
            </div> :
            <>
                <div>
                    {Object.keys(favorites).length
!== 0 ? <div className=" bg-gray-100 grid gap-4 grid-cols-3 p-3">
                        {
                            favorites.results.map(el
=> (
                                <div key={el.id}
                                className="border-8 relative">
                                    <Link
                                        href={el.id}
                                        className="
                                        hover:bg-slate-200 h-full flex"
                                    >
                                        <EachCard
                                            event={el} />
                                        </Link>
                                        <FavoritesButton
                                            id={el.id} addClasses={'absolute bottom-8 px-4'} />
                                        <a
                                            href={el.site_url} className="px-4 absolute bottom-1 text-blue-600
visited:text-purple-600 pointer-events-auto hover:text-orange-600 text-
[7px] sm:text-sm/[10px] lg:text-base">Перейти на сайт мероприятия</a>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                    )
                </div>
            </div>
        : null}
    </div>
</>
}

```

```

        </div>
      </div>
    </div>
  </AuthenticatedLayout>
</>
);
}

```

Листинг 14 – AuthenticatedLayout.jsx

```

export default function AuthenticatedLayout({ header, children }) {

  return (
    <div className="min-h-screen bg-gray-100">
      {header && (
        <header className="bg-white shadow bg-gray-200">
          <div className="mx-auto px-4 py-6 sm:px-6 lg:px-8 bg-
gray-300">
            {header}
          </div>
        </header>
      )}
      <main>{children}</main>
    </div>
  );
}

```

Листинг 15 – GuestLayout.jsx

```

import { Link } from '@inertiajs/react';

export default function GuestLayout({ children }) {
  return (
    <div className="flex min-h-screen flex-col items-center bg-gray-
100 pt-6 sm:justify-center sm:pt-0">
      <div>
        <Link href="/">
          <div className="flex shrink-0 items-center stroke-
2">
            <svg className="h-20 w-auto fill-current text-
gray-500" id="Capa_1" enableBackground="new 0 0 512 512" viewBox="0 0 512
512" xmlns="http://www.w3.org/2000/svg"><g><path d="m345.631 115.263v-
33.5091-99.594 12.245v33.509z" /><path d="m95.602

```

```

418.266h320.797v61.867c0 17.6 14.268 31.867 31.867 31.867 17.6 0 31.867-
14.267 31.867-31.867v-480.133h-384.531zm304.863-35.731h-145.527v-
30h145.527zm-217.229-173.402h32.801v-141.671l1159.594-19.622v175.359h-
62.801v-30h32.801v-47.711l-99.594 12.245v81.399h-62.801zm-7.966
79.668h225.195v30h-225.195zm0 63.734h47.801v30h-47.801z" /><path
d="m386.398 448.266h-354.531v31.867c0 17.6 14.267 31.867 31.867
31.867h331.52c-5.619-9.313-8.856-20.219-8.856-31.867z" /></g></svg>
    </div>
  </Link>
</div>

    <div className="mt-6 w-full overflow-hidden bg-white px-6 py-
4 shadow-md sm:max-w-md sm:rounded-lg">
      {children}
    </div>
  </div>
);
}

```

Листинг 16 – Header.jsx

```

import { Disclosure, DisclosureButton, DisclosurePanel } from
'@headlessui/react'
import { Bars3Icon, XMarkIcon } from '@heroicons/react/24/outline'
import { Link, usePage } from '@inertiajs/react';
import Dropdown from '@Components/Dropdown';

import { useState } from 'react';
const navigation = [
  { name: 'Санкт-Петербург', city: 'spb', current: true, href: '/spb'
},
  { name: 'Москва', city: 'msk', current: false, href: '/msk' },
  { name: 'Краснодар', city: 'krd', current: false, href: '/krd' },
  { name: 'Сочи', city: 'sochi', current: false, href: '/sochi' },
]

export default function Header() {
  const [showingNavigationDropdown, setShowingNavigationDropdown] =
    useState(false);
  const page = usePage();
  return (
    <Disclosure as="nav" className="bg-gray-800">
      <div className="mx-auto max-w-7xl px-2 sm:px-6 lg:px-8">

```

```

        <div className="relative flex h-16 items-center justify-
between">
            <div className="absolute inset-y-0 left-0 flex items-
center sm:hidden">
                { /* Mobile right menu button */ }
                <DisclosureButton className="group relative
inline-flex items-center justify-center rounded-md p-2 text-gray-400
hover:bg-gray-700 hover:text-white focus:outline-none focus:ring-2
focus:ring-inset focus:ring-white">
                    <span className="absolute -inset-0.5" />
                    <span className="sr-only">Меню</span>
                    <Bars3Icon aria-hidden="true"
className="block size-6 group-data-[open]:hidden" />
                    <XMarkIcon aria-hidden="true"
className="hidden size-6 group-data-[open]:block" />
                </DisclosureButton>
            </div>
            <div className="flex flex-1 items-center justify-left
pl-12 sm:justify-start sm:justify-center sm:pl-0 ">
                <Link href="/">
                    <div className="flex shrink-0 items-center
stroke-2">
                        <svg className="h-8 w-auto fill-orange-
400 " id="Capa_1" enableBackground="new 0 0 512 512" viewBox="0 0 512
512" xmlns="http://www.w3.org/2000/svg"><g><path d="m345.631 115.263v-
33.509l-99.594 12.245v33.509z" /><path d="m95.602
418.266h320.797v61.867c0 17.6 14.268 31.867 31.867 31.867 17.6 0 31.867-
14.267 31.867-31.867v-480.133h-384.531zm304.863-35.731h-145.527v-
30h145.527zm-217.229-173.402h32.801v-141.671l159.594-19.622v175.359h-
62.801v-30h32.801v-47.711l-99.594 12.245v81.399h-62.801zm-7.966
79.668h225.195v30h-225.195zm0 63.734h47.801v30h-47.801z" /><path
d="m386.398 448.266h-354.531v31.867c0 17.6 14.267 31.867 31.867
31.867h331.52c-5.619-9.313-8.856-20.219-8.856-31.867z" /></g></svg>
                    </div>
                </Link>
                <div className="hidden sm:ml-6 sm:block">
                    <div className="flex space-x-4">
                        {navigation.map((item) => (
                            <Link href={route(item.city)}
                                key={item.name}
                                className={
                                    (page.url ===
`/${item.city}`) ? 'bg-gray-900 text-white rounded-md px-3 py-2 text-sm
font-medium flex items-center' : 'text-gray-300 hover:bg-gray-700

```



```

hover:text-white rounded-md px-3 py-2 text-sm font-medium align-bottom
flex items-center'}
        >
            {item.name}
        </Link>
    )))
</div>
</div>
<div className="absolute inset-y-0 right-0 flex
items-center pr-2 sm:static sm:inset-auto sm:ml-6 sm:pr-0">

    {/* Profile dropdown */}
    <nav className="-mx-3 flex flex-1 items-center
justify-end">
        {page.props.auth.user ? (
            <Link
                href={route('dashboard')}
                onClick={() => changeLocation()}
                className="rounded-md px-3 py-2 text-
sm font-medium text-gray-300 ring-1 ring-transparent transition
hover:text-white focus:outline-none focus-visible:ring-[#FF2D20]
dark:text-white dark:hover:text-white/80 dark:focus-visible:ring-white
hover:bg-gray-700 text-sm"
            >
                Избранное
            </Link>
        ) : (
            <>
                <Link
                    href={route('login')}
                    onClick={() => changeLocation()}
                    className="rounded-md pr-2 pl-3
py-2 text-white/50 ring-1 ring-transparent transition hover:text-white/70
focus:outline-none focus-visible:ring-[#FF2D20] dark:text-white
dark:hover:text-white/80 dark:focus-visible:ring-white align-middle text-
sm"
                >
                    Войти
                </Link>
                <Link
                    href={route('register')}
                    onClick={() => changeLocation()}

```

```

        className="rounded-md pr-3 pl-2
py-2 text-white/50 ring-1 ring-transparent transition hover:text-white/70
focus:outline-none focus-visible:ring-[#FF2D20] dark:text-white
dark:hover:text-white/80 dark:focus-visible:ring-white text-sm"
      >
        Зарегистрироваться
      </Link>
    </>
  )}
</nav>
{page.props.auth.user &&
  <div className="hidden sm:ms-6 sm:flex
sm:items-center">
    <div className="relative ms-3">
      <Dropdown>
        <Dropdown.Trigger>
          <span className="inline-flex
rounded-md">
            <button
              type="button"
              className="inline-
flex items-center rounded-md border border-transparent px-3 py-2 text-sm
font-medium leading-4 text-gray-300 transition duration-150 ease-in-out
hover:bg-gray-700 focus:outline-none hover:text-white"
            >
              {page.props.auth.user
.name}
            <svg
              className="-me-
0.5 ms-2 h-4 w-4"
              xmlns="http://www
.w3.org/2000/svg"
              viewBox="0 0 20
20"
              fill="currentColo
r"
            >
              <path
                fillRule="eve
nodd"

```

```

d="M5.293
7.293a1 1 0 011.414 0L10 10.586l3.293-3.293a1 1 0 111.414 1.414l-4 4a1 1
0 01-1.414 0l-4-4a1 1 0 010-1.414z"
clipRule="evenodd"

      />
    </svg>
  </button>
</span>
</Dropdown.Trigger>

  <Dropdown.Content>
    <Dropdown.Link
      href={route('profile.edit')}
    >
      Профиль
    </Dropdown.Link>
    <Dropdown.Link
      href={route('logout')}
      method="post"
      as="button"
    >
      Выйти
    </Dropdown.Link>
  </Dropdown.Content>
</Dropdown>
</div>
</div>}

{page.props.auth.user &&
  <div className="-me-2 flex items-center
sm:hidden">

    <Dropdown>
      <Dropdown.Trigger>
        <button
          className="inline-flex items-
center justify-center rounded-md p-2 text-gray-400 transition duration-
150 ease-in-out hover:bg-gray-100 hover:text-gray-500 focus:bg-gray-100
focus:text-gray-500 focus:outline-none ml-2"
        >
          <svg

```

```

        className="h-6 w-6"
        stroke="currentColor"
        fill="none"
        viewBox="0 0 24 24"
      >
        <path
          className={
            !showingNavigation
              ? 'inline-
flex'
              : 'hidden'
          }
          strokeLinecap="round"
          strokeLinejoin="round"
          strokeWidth="2"
          d="M4 6h16M4 12h16"
        />
        <path
          className={
            showingNavigation
              ? 'inline-
flex'
              : 'hidden'
          }
          strokeLinecap="round"
          strokeLinejoin="round"
          strokeWidth="2"
          d="M6 18L18 6M6 12L12 6"
        />
      </svg>
    </button>

    </Dropdown.Trigger>

    <Dropdown.Content>
      <Dropdown.Link
        href={route('profile.edit')}

```

```

        >
        Профиль
      </Dropdown.Link>
      <Dropdown.Link
        href={route('logout')}
        method="post"
        as="button"
      >
        Выйти
      </Dropdown.Link>
    </Dropdown.Content>
  </Dropdown>
</div>
}
</div>
</div>
</div>

{/* Mobile left menu button*/}

<DisclosurePanel className="sm:hidden">
  <div className="space-y-1 px-2 pb-3 pt-2">
    {navigation.map((item) => (
      <DisclosureButton
        key={item.name}
        as="a"
        href={item.href}
        aria-current={item.current ? 'page' :
undefined}
        className={({page.url === `/${item.city}`) ?
'bg-gray-900 text-white block rounded-md px-3 py-2 text-base font-medium'
: 'text-gray-300 hover:bg-gray-700 hover:text-white block rounded-md px-3
py-2 text-base font-medium'
      }
    >
      {item.name}
    </DisclosureButton>
    )
  )
  )
  </div>
</DisclosurePanel>
</Disclosure >
)
}

```

Листинг 17 – EachCard.jsx

```
function EachCard({ event }) {

  function capitalizeFirstLetter(event) {
    return event.title.charAt(0).toUpperCase() +
event.title.slice(1);
  }
  function filter(event) {
    return event.dates.filter(el => ((Date.now() > el.start * 1000 &&
Date.now() < el.end * 1000) || (Date.now() < el.start * 1000)));
  }

  return (
    <>
      <div className="each_event pt-4 pr-4 pl-4 bg-gray-100 w-full
hover:bg-slate-200">
        <div className="h-10 mb-14 sm:mb-12 lg:mb-10 italic text-
gray-600 text-[8px] sm:text-base lg:text-lg
"><p>{capitalizeFirstLetter(event)}</p></div>
        <div className="justify-items-center pt-3 w-full h-[10vh]
sm:h-[20vh] lg:h-[30vh]" >
          <img className="w-auto h-full object-cover"
src={event.images[0].image} alt="event_image" />
        </div>

        {filter(event).slice(0, 4).length === 0 ?
          <div><p className="mt-3 mb-12 text-xs sm:text-sm
lg:text-base not-italic">Событие прошло</p></div>
          :
          <>
            {
              filter(event).slice(0, 4)[0].start <= 0 ?
                <div><p className="mt-3 mb-12 text-xs
sm:text-sm lg:text-base not-italic">Проводится постоянно</p></div>
                :
                <div className='mt-3 mb-16 italic text-
[8px] sm:text-sm lg:text-base'>Ближайшие даты проведения:
{filter(event).slice(0, 4).map((el, index) => (
```

```

        <div key={index}>
            {(new Date(el.start *
1000).toLocaleDateString("ru-RU")) === (new Date(el.end *
1000).toLocaleDateString("ru-RU")) ? <p className="text-[8px] sm:text-sm
lg:text-base not-italic"> {new Date(el.start *
1000).toLocaleDateString("ru-RU")}</p>
: <p className="text-[8px]
sm:text-sm lg:text-base not-italic "> {new Date(el.start *
1000).toLocaleDateString("ru-RU")}-{new Date(el.end *
1000).toLocaleDateString("ru-RU")}</p>}}
        </div>
    ))}</div>
    }</>
    }
    </div >
    </>);
}

export default EachCard;

```

Листинг 18 – EachEvent.jsx

```

import Header from "@/Layouts/Header";
import { useSelector } from "react-redux";
import FavoritesButton from "./FavoritesButton";

function EachEvent() {
    const { event, loading, error } = useSelector((state) =>
state.event);

    function capitalizeFirstLetter(event) {
        return event.title.charAt(0).toUpperCase() +
event.title.slice(1);
    }
    function cleanTextFromTags(text) {
        return text.replace(/<[>]*>/g, '');
    }
    function filter(event) {
        return event.dates.filter(el => ((Date.now() > el.start * 1000 &&
Date.now() < el.end * 1000) || (Date.now() < el.start * 1000)));
    }

    return (
        <>

```

```

    <Header />
    {loading && loading && <div role="status" className="flex
items-center justify-center bg-gray-100 !shadow-none">
      <svg aria-hidden="true" className="w-8 h-8 text-gray-200
animate-spin dark:text-gray-600 fill-blue-600" viewBox="0 0 100 101"
fill="none" xmlns="http://www.w3.org/2000/svg">
        <path d="M100 50.5908C100 78.2051 77.6142 100.591 50
100.591C22.3858 100.591 0 78.2051 0 50.5908C0 22.9766 22.3858 0.59082 50
0.59082C77.6142 0.59082 100 22.9766 100 50.5908ZM9.08144 50.5908C9.08144
73.1895 27.4013 91.5094 50 91.5094C72.5987 91.5094 90.9186 73.1895
90.9186 50.5908C90.9186 27.9921 72.5987 9.67226 50 9.67226C27.4013
9.67226 9.08144 27.9921 9.08144 50.5908Z" fill="currentColor" />
        <path d="M93.9676 39.0409C96.393 38.4038 97.8624
35.9116 97.0079 33.5539C95.2932 28.8227 92.871 24.3692 89.8167
20.348C85.8452 15.1192 80.8826 10.7238 75.2124 7.41289C69.5422 4.10194
63.2754 1.94025 56.7698 1.05124C51.7666 0.367541 46.6976 0.446843 41.7345
1.27873C39.2613 1.69328 37.813 4.19778 38.4501 6.62326C39.0873 9.04874
41.5694 10.4717 44.0505 10.1071C47.8511 9.54855 51.7191 9.52689 55.5402
10.0491C60.8642 10.7766 65.9928 12.5457 70.6331 15.2552C75.2735 17.9648
79.3347 21.5619 82.5849 25.841C84.9175 28.9121 86.7997 32.2913 88.1811
35.8758C89.083 38.2158 91.5421 39.6781 93.9676 39.0409Z"
fill="currentFill" />
      </svg>
      <span className="sr-only">Loading...</span>
    </div>}
    {error && <p>Ошибка {error}</p>}
    {Object.keys(event).length !== 0 ?
      <div className="each_event p-4 bg-gray-100 ">
        <div className="h-10 mb-10 italic text-4xl font-serif
"><p>{!loading && capitalizeFirstLetter(event)}</p></div>
        <div className="grid justify-items-center pt-3 pb-4"
><img src={event.images[0].image} alt="event_image" /></div>
        <div className="text-left mb-6 font-serif tracking-
wide leading-loose font-semibold italic">
          {cleanTextFromTags(event.description)}
        </div>
        <div className="text-left mb-6 font-serif tracking-
wide leading-loose" dangerouslySetInnerHTML={{ __html: event.body_text
}}>
        </div>
        <div className="text-left mb-6 font-serif tracking-
wide leading-loose font-semibold italic">

```



```

        {event.age_restriction === 0 ||
event.age_restriction === null ? <p>
        Возрастное ограничение: Без ограничений
    </p> :
    <p>
        Возрастное ограничение:
{event.age_restriction}
    </p>
    }

</div>
<div className="text-left mb-6 font-serif tracking-
wide leading-loose font-semibold italic">
    {event.is_free ? <p>Стоимость: Бесплатное
мероприятие</p>
    :
    <p>Стоимость: {event.price}</p>
    }
</div>
{filter(event).slice(0, 4).length === 0 ?
    <div><p className="text-left mb-6 font-serif
tracking-wide leading-loose">Событие прошло</p></div>
    :
    <>
        {
            filter(event).slice(0, 4)[0].start <= 0 ?
                <div className="text-left mb-6 font-
serif tracking-wide leading-loose "><p className="mt-3">Проводится
постоянно</p></div>
                :
                <div className="text-left mb-6
tracking-wide leading-loose text-lg">Ближайшие даты проведения:
{filter(event).slice(0, 4).map((el, index) => (
                    <div key={index}>
                        {(new Date(el.start *
1000).toLocaleDateString("ru-RU")) === (new Date(el.end *
1000).toLocaleDateString("ru-RU")) ? <p> {new Date(el.start *
1000).toLocaleDateString("ru-RU")}</p>
                        : <p> {new Date(el.start
* 1000).toLocaleDateString("ru-RU")}-{new Date(el.end *
1000).toLocaleDateString("ru-RU")}</p>}
                    </div>
                ))}</div>
        }
    </>
}

```

```

        }</>
    }

    <div className="flex justify-between">
        <a href={event.site_url} className="text-blue-600
visited:text-purple-600 pointer-events-auto hover:text-orange-
600">Перейти на сайт мероприятия</a>
        <FavoritesButton id={event.id} />
    </div>
</div> :
null}

</>);
}

export default EachEvent;

```

Листинг 19 – FavoritesButton.jsx

```

import { fetchAddToFavorites } from "@Store/addToFavorites";
import { fetchDeleteFromFavorites } from "@Store/deleteFromFavorites";
import { usePage } from "@inertiajs/react";
import { useState } from "react";
import { useDispatch } from "react-redux";

function FavoritesButton({ id, addClasses }) {
    const userProps = usePage().props.auth.user;
    let favorites = '';
    if (userProps) {
        favorites = usePage().props.auth.user.favorites;
    }
    if (favorites !== null) {
        favorites = Object.values(favorites);
    }

    const [inFavoritesTrue, setInFavoritesTrue] = useState(true);
    const [inFavoritesFalse, setInFavoritesFalse] = useState(true);
    const dispatch = useDispatch();
    function addToFavorites(flag) {
        dispatch(fetchAddToFavorites({ id: id, user: userProps.id }));
        if (flag) {
            setInFavoritesFalse(!inFavoritesFalse);
        }
        else {

```

```

        setInFavoritesTrue(!inFavoritesTrue);
    }
}
function deleteFromFavorites(flag) {
    dispatch(fetchDeleteFromFavorites({ id: id, user: userProps.id
})));
    if (flag) {
        setInFavoritesFalse(!inFavoritesFalse);
    }
    else {
        setInFavoritesTrue(!inFavoritesTrue);
    }
}

return (
    <>
        {userProps &&
            <button className={`transition duration-150 ${addClasses}
`}
                >
                    {favorites === null ? <span onClick={() =>
inFavoritesFalse ? addToFavorites(true) : deleteFromFavorites(true)}
                        className={`${inFavoritesFalse ? 'inline-flex
items-center rounded-md bg-green-50 px-2 py-1 text-[6px] sm:text-
xs/[10px] font-medium text-green-700 ring-1 ring-inset ring-green-600/40
hover:text-green-900 hover:bg-green-100' : 'inline-flex items-center
rounded-md bg-green-50 px-2 py-1 text-[6px] sm:text-xs/[10px] font-
medium text-orange-700 ring-1 ring-inset ring-red-800/50 hover:text-red-
900 hover:bg-red-100'}`}>
                        {inFavoritesFalse ? <>В избранное</> : <>Убрать
из избранного</>}
                    </span> :
                        <>
                            {(!(favorites.includes(String(id)))) ?
                                <span onClick={() => inFavoritesFalse ?
addToFavorites(true) : deleteFromFavorites(true)}
                                    className={`${inFavoritesFalse ?
'inline-flex items-center rounded-md bg-green-50 px-2 py-1 text-[6px]
sm:text-xs/[10px] font-medium text-green-700 ring-1 ring-inset ring-
green-600/40 hover:text-green-900 hover:bg-green-100' : 'inline-flex

```

```

items-center rounded-md bg-green-50 px-2 py-1 text-[6px] sm:text-
xs/[10px] font-medium text-orange-700 ring-1 ring-inset ring-red-800/50
hover:text-red-900 hover:bg-red-100'}``}>
                                {inFavoritesFalse ? <>В избранное</>
: <>Убрать из избранного</>}

                                </span> : <span

                                onClick={() => inFavoritesTrue ?
deleteFromFavorites(false) : addToFavorites(false)}
                                className={`${inFavoritesTrue ?
"favorite inline-flex items-center rounded-md bg-green-50 px-2 py-1 text-
[6px] sm:text-xs/[10px] font-medium text-orange-700 ring-1 ring-inset
ring-red-800/50 hover:text-red-900 hover:bg-red-100" : 'inline-flex
items-center rounded-md bg-green-50 px-2 py-1 text-[6px] sm:text-
xs/[10px] font-medium text-green-700 ring-1 ring-inset ring-green-600/40
hover:text-green-900 hover:bg-green-100'}`}>
                                {inFavoritesTrue ? <>Убрать из
избранного</> : <>В избранное</>}
                                </span></>}

                                </button >

                                }

                                </>

                                );
}
export default FavoritesButton;

```

Листинг 20 – Pagination.jsx

```

import { ChevronLeftIcon, ChevronRightIcon } from
'@heroicons/react/20/solid'
import { useSelector } from 'react-redux';
import { useDispatch } from "react-redux";
import { fetchAfisha } from '../store/afishaReducer';
import { useEffect, useState } from 'react';

export default function Pagination({ city }) {
  const { afisha } = useSelector((state) => state.afisha);
  const [page, setPage] = useState(1);
  const [show_result_from, setShow_result_from] = useState(1);
  const [show_result_to, setShow_result_to] = useState(21);
  const count_results = afisha.count;

```

```

let count_pages = 0;
if (afisha.count) {
  count_pages = Math.ceil(afisha.count / 21);
}
useEffect(() => {
  setPage(1);
  setShow_result_from(1);
  if (count_results < 21) {
    setShow_result_to(count_results)
  }
  setShow_result_to(21);

}, [city, count_results])
const dispatch = useDispatch();
const handleChangePage = (page) => {
  window.scrollTo(0, 0);
  setPage(page);
  dispatch(fetchAfisha({ page: page, city: city }));
  setShow_result_from(page * 21 - 20);
  setShow_result_to(page * 21);
}
return (
  <div className="flex items-center justify-between border-t
border-gray-200 bg-white px-4 py-3 sm:px-6">
    <div className="flex flex-1 justify-around sm:hidden">
      <button
        onClick={() => handleChangePage(page - 1)}
        className={page !== 1 ? "relative inline-flex items-
center rounded-md border border-gray-300 bg-white px-4 py-2 text-sm font-
medium text-gray-700 hover:bg-gray-50" :
          "hidden"}
      >
        Previous
      </button>
      <button
        onClick={() => handleChangePage(page + 1)}
        className={page !== count_pages ? "relative ml-3
inline-flex items-center rounded-md border border-gray-300 bg-white px-4
py-2 text-sm font-medium text-gray-700 hover:bg-gray-50" :
          "hidden"}
      >
        Next
      </button>

```

```

        </div>
        <div className="hidden sm:flex sm:flex-1 sm:items-center
sm:justify-between">
            <div>
                <p className="text-sm text-gray-700">
                    Showing <span className="font-
medium">{show_result_from}</span> to <span className="font-medium">{page
=== count_pages ? count_results : show_result_to}</span> of{' '}
                    <span className="font-
medium">{count_results}</span> results
                </p>
            </div>
            <div>
                <nav aria-label="Pagination" className="isolate
inline-flex -space-x-px rounded-md shadow-sm">
                    <button
                        onClick={() => handleChangePage(page - 1)}
                        className={page !== 1 ? "relative inline-flex
items-center rounded-l-md px-2 py-2 text-gray-400 ring-1 ring-inset ring-
gray-300 hover:bg-gray-50 focus:z-20 focus:outline-offset-0" : "relative
inline-flex items-center rounded-l-md px-2 py-2 text-gray-400 ring-1
ring-inset ring-gray-300 hover:bg-gray-50 focus:z-20 focus:outline-
offset-0 pointer-events-none"}
                    >
                        <span className="sr-only">Previous</span>
                        <ChevronLeftIcon aria-hidden="true"
className="size-5" />
                    </button>
                    <button
                        onClick={() => handleChangePage(1)}
                        className={page === 1 ? "relative z-10
inline-flex items-center bg-indigo-600 px-4 py-2 text-sm font-semibold
text-white focus:z-20 focus-visible:outline focus-visible:outline-2
focus-visible:outline-offset-2 focus-visible:outline-indigo-600" :
"relative items-center px-4 py-2 text-sm font-semibold text-gray-900
ring-1 ring-inset ring-gray-300 hover:bg-gray-50 focus:z-20
focus:outline-offset-0 md:inline-flex"}
                    >
                        1
                    </button>
                    {page >= 3 &&
                        <button

```

```

onClick={() => handleChangePage(page -
1)}}

        className="relative items-center px-4 py-
2 text-sm font-semibold text-gray-900 ring-1 ring-inset ring-gray-300
hover:bg-gray-50 focus:z-20 focus:outline-offset-0 md:inline-flex"
        >
            {page - 1}
        </button>
    }
    {page !== 1 && page !== count_pages &&
        <span className="relative z-10 inline-flex
items-center bg-indigo-600 px-4 py-2 text-sm font-semibold text-white
focus:z-20 focus-visible:outline focus-visible:outline-2 focus-
visible:outline-offset-2 focus-visible:outline-indigo-600">
            {page}
        </span>
    }
    {page <= count_pages - 2 &&
        <button
            onClick={() => handleChangePage(page +
1)}}

            className="relative items-center px-4 py-
2 text-sm font-semibold text-gray-900 ring-1 ring-inset ring-gray-300
hover:bg-gray-50 focus:z-20 focus:outline-offset-0 md:inline-flex"
            >
                {page + 1}
            </button>
        }
        {count_pages !== 1 &&
            <button
                onClick={() =>
handleChangePage(count_pages)}

                className={page === count_pages ?
"relative z-10 inline-flex items-center bg-indigo-600 px-4 py-2 text-sm
font-semibold text-white focus:z-20 focus-visible:outline focus-
visible:outline-2 focus-visible:outline-offset-2 focus-visible:outline-
indigo-600" : "relative items-center px-4 py-2 text-sm font-semibold
text-gray-900 ring-1 ring-inset ring-gray-300 hover:bg-gray-50 focus:z-20
focus:outline-offset-0 md:inline-flex"}
            >

                {count_pages}
            </button>

```

```

        }
        <button
            onClick={() => handleChangePage(page + 1)}
            className={page !== count_pages ? "relative
inline-flex items-center rounded-r-md px-2 py-2 text-gray-400 ring-1
ring-inset ring-gray-300 hover:bg-gray-50 focus:z-20 focus:outline-
offset-0" : "relative inline-flex items-center rounded-r-md px-2 py-2
text-gray-400 ring-1 ring-inset ring-gray-300 hover:bg-gray-50 focus:z-20
focus:outline-offset-0 pointer-events-none"}
            >
                <span className="sr-only">Next</span>
                <ChevronRightIcon aria-hidden="true"
className="size-5" />
            </button>
        </nav>
    </div>
</div>
</div>
)
}

```

Листинг 21 – app.blade.php

```

<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title inertia>{{ config('app.name', 'Laravel') }}</title>

    <!-- Fonts -->
    <link rel="preconnect" href="https://fonts.bunny.net">
    <link
href="https://fonts.bunny.net/css?family=figtree:400,500,600&display=swap
" rel="stylesheet" />

    <!-- Scripts -->
    @routes
    @viteReactRefresh
    @vite(['resources/js/app.jsx',
"resources/js/Pages/{$page['component']}.jsx"])

```



```

    @inertiaHead
  </head>

  <body class="font-sans antialiased">
    @inertia
  </body>

</html>

```

Листинг 22 – app.jsx

```

import './css/app.css';
import './bootstrap';
import { createInertiaApp } from '@inertiajs/react';
import { resolvePageComponent } from 'laravel-vite-plugin/inertia-helpers';
import { createRoot } from 'react-dom/client';
import { Provider } from 'react-redux';
import store from './store/store';
import { BrowserRouter } from 'react-router-dom';

const appName = import.meta.env.VITE_APP_NAME || 'Laravel';

createInertiaApp({
  title: (title) => `${title} - ${appName}`,
  resolve: (name) =>
    resolvePageComponent(
      `./Pages/${name}.jsx`,
      import.meta.glob('./Pages/**/*.jsx'),
    ),
  setup({ el, App, props }) {
    const root = createRoot(el);
    {
      root.render(
        <Provider store={store}>
          <BrowserRouter>
            <App {...props} />
          </BrowserRouter>
        </Provider>
      )
    }
  },
},

```

```

    progress: {
        color: '#4B5563',
    },
});

```

Листинг 23 – composer.json

```

{
    "$schema": "https://getcomposer.org/schema.json",
    "name": "laravel/laravel",
    "type": "project",
    "description": "The skeleton application for the Laravel framework.",
    "keywords": ["laravel", "framework"],
    "license": "MIT",
    "require": {
        "php": "^8.2",
        "inertiajs/inertia-laravel": "^1.0",
        "laravel/breeze": "^2.2",
        "laravel/framework": "^11.31",
        "laravel/sanctum": "^4.0",
        "laravel/tinker": "^2.9",
        "tightenco/ziggy": "^2.0"
    },
    "require-dev": {
        "fakerphp/faker": "^1.23",
        "laravel/pail": "^1.1",
        "laravel/pint": "^1.13",
        "laravel/sail": "^1.26",
        "mockery/mockery": "^1.6",
        "nunomaduro/collision": "^8.1",
        "phpunit/phpunit": "^11.0.1"
    },
    "autoload": {
        "psr-4": {
            "App\\": "app/",
            "Database\\Factories\\": "database/factories/",
            "Database\\Seeders\\": "database/seeders/"
        }
    },
    "autoload-dev": {
        "psr-4": {
            "Tests\\": "tests/"
        }
    },
}

```

```

"scripts": {
    "post-autoload-dump": [
        "Illuminate\\Foundation\\ComposerScripts::postAutoloadDump",
        "@php artisan package:discover --ansi"
    ],
    "post-update-cmd": [
        "@php artisan vendor:publish --tag=laravel-assets --ansi --
force"
    ],
    "post-root-package-install": [
        "@php -r \"file_exists('.env') || copy('.env.example',
'.env');\""
    ],
    "post-create-project-cmd": [
        "@php artisan key:generate --ansi",
        "@php -r \"file_exists('database/database.sqlite') ||
touch('database/database.sqlite');\"",
        "@php artisan migrate --graceful --ansi"
    ],
    "dev": [
        "Composer\\Config::disableProcessTimeout",
        "npx concurrently -c \"#93c5fd,#c4b5fd,#fb7185,#fdb74\" \"
php artisan serve\" \"php artisan queue:listen --tries=1\" \"php
artisan pail --timeout=0\" \"npm run dev\" --
names=server,queue,logs,vite"
    ]
},
"extra": {
    "laravel": {
        "dont-discover": []
    }
},
"config": {
    "optimize-autoloader": true,
    "preferred-install": "dist",
    "sort-packages": true,
    "allow-plugins": {
        "pestphp/pest-plugin": true,
        "php-http/discovery": true
    }
},
"minimum-stability": "stable",
"prefer-stable": true

```

```
}
```

Листинг 24 – package.json

```
{
  "private": true,
  "type": "module",
  "scripts": {
    "build": "vite build",
    "dev": "vite"
  },
  "devDependencies": {
    "@inertiajs/react": "^1.0.0",
    "@tailwindcss/forms": "^0.5.3",
    "@vitejs/plugin-react": "^4.2.0",
    "@reduxjs/toolkit": "^2.2.7",
    "@headlessui/react": "^2.2.0",
    "@heroicons/react": "^2.2.0",
    "autoprefixer": "^10.4.12",
    "axios": "^1.7.4",
    "concurrently": "^9.0.1",
    "laravel-vite-plugin": "^1.0",
    "postcss": "^8.4.31",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-redux": "^9.1.2",
    "react-router-dom": "^6.26.2",
    "react-scripts": "5.0.1",
    "tailwindcss": "^3.4.16",
    "vite": "^6.0"
  }
}
```

Листинг 25 – vite.config.js

```
import { defineConfig } from 'vite';
import laravel from 'laravel-vite-plugin';
import react from '@vitejs/plugin-react';

export default defineConfig({
  plugins: [
    laravel({
      input: 'resources/js/app.jsx',
```

```

        refresh: true,
      )),
      react(),
    ],
    server: {
      strictPort: true,
      port: 5173,
      host: '0.0.0.0',
      origin: 'http://localhost:5173',
      hmr: {
        host: 'localhost',
      },
    },
  }
});

```

Листинг 26 – web.php

```

<?php

use App\Http\Controllers\FetchController;
use App\Http\Controllers\ProfileController;
use Illuminate\Support\Facades\Route;
use Inertia\Inertia;

Route::redirect('/', '/spb');

Route::get('/spb', function () {
    return Inertia::render('ShowAfishaInfo', [
        'city' => 'spb'
    ]);
})->name('spb');

Route::get('/msk', function () {
    return Inertia::render('ShowAfishaInfo', [
        'city' => 'msk'
    ]);
})->name('msk');

Route::get('/krd', function () {
    return Inertia::render('ShowAfishaInfo', [
        'city' => 'krd'
    ]);
})->name('krd');

Route::get('/sochi', function () {
    return Inertia::render('ShowAfishaInfo', [

```

```

        'city' => 'sochi'
    });
})->name('sochi');

Route::get('{id?}', function ($id) {
    return Inertia::render('FetchEvent', [
        'id' => $id
    ]);
})->whereNumber('id')->name('eachEvent');

Route::get('api/getfetchinfo', [FetchController::class, 'getFetchInfo']);
Route::get('api/fetcheachevent', [FetchController::class,
'fetchEachEvent']);
Route::get('api/fetchfavorites', [FetchController::class,
'fetchFavorites']);
Route::get('api/savefavorites', [FetchController::class,
'saveFavorites']);
Route::get('api/deletefavorites', [FetchController::class,
'deleteFavorites']);

Route::get('/dashboard', function () {
    return Inertia::render('Dashboard');
})->middleware(['auth', 'verified'])->name('dashboard');

Route::middleware('auth')->group(function () {
    Route::get('/profile', [ProfileController::class, 'edit'])->
>name('profile.edit');
    Route::patch('/profile', [ProfileController::class, 'update'])->
>name('profile.update');
    Route::delete('/profile', [ProfileController::class, 'destroy'])->
>name('profile.destroy');
});

require __DIR__ . '/auth.php';

```