Geekbrains

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

Программа: разработчик

Специализация: Fullstack разработчик

Алексеев Дмитрий Михайлович

Санкт-Петербург

2025

**Основные требования к оформлению дипломного проекта**

1. Текст проекта оформляйте только с одной стороны листа формата А4. При наборе используйте межстрочный интервал 1,5.  
   Объемные таблицы/иллюстрации возможно размещать на листах А3, А2 и выносить в приложения.
2. При наборе информации, используйте шрифт Times New Roman с выравниванием текста по ширине. Рекомендуем кегль — 12–14, для некоторых структурных единиц работы можно выбрать полужирное выделение.
3. Поля страницы, где размещается текст, выставляйте по определенным показателям: нижнее/верхнее – 20 мм, правое – 10 мм, левое (для последующей прошивки) – 30 мм.
4. Все листы дипломного проекта нумеруются по порядку, начиная с введения (используется сквозная нумерация). Номер указывайте в центре нижней части страницы без точки.
5. Заголовок раздела с одним подразделом или пунктом не нумеруется.
6. После текста перед следующим заголовком установите одиночный интервал.
7. После заголовка перед подзаголовком проставьте двойной интервал.
8. Имена собственные — наименования компаний/изделий, фамилии учёных и т. д., указывайте на языке оригинала.
9. Каждый структурный элемент начинается с нового листа.

*Вся работа должна составлять 50+ страниц формата А4, включая приложения.*

|  |
| --- |
| 💡 Инструкция по работе с шаблоном.   1. Создайте копию данного шаблона. Файл - Создать копию. 2. Изучите шаблон и начинайте работу. |

Оглавление

[1. Введение 6](#_Toc186466349)

[2. Теоретическая часть 7](#_Toc186466350)

[2.1. Введение в веб-разработку 7](#_Toc186466351)

[2.2. React: библиотека для создания пользовательских интерфейсов 7](#_Toc186466352)

[2.2.1. Введение в React. 7](#_Toc186466353)

[2.2.2. Основные концепции и принципы React 7](#_Toc186466354)

[2.2.3. Преимущества использования React 10](#_Toc186466355)

[2.2.4. Заключение 11](#_Toc186466356)

[2.3. Laravel: PHP-фреймворк для бэкенд-разработки 11](#_Toc186466357)

[2.3.1. Введение в Laravel 11](#_Toc186466358)

[2.3.2. Основные концепции и принципы Laravel 11](#_Toc186466359)

[2.3.3. Преимущества использования Laravel 14](#_Toc186466360)

[2.3.4. Заключение 14](#_Toc186466361)

[2.4. Архитектура веб-приложения 15](#_Toc186466362)

[2.5. Основные функции приложения 15](#_Toc186466363)

[2.6. Заключение 15](#_Toc186466364)

[Заключение 16](#_Toc186466365)

[Список используемой литературы 17](#_Toc186466366)

[Пример 17](#_Toc186466367)

[Приложения 18](#_Toc186466368)

Пример

*Данный пример не является эталонным, лишь один из вариаций.*

Введение (2–3 стр.)

Глава 1. Основы ручного тестирования веб-приложений (~15 стр.)

1.1 Что такое веб-приложение, его особенности

1.2 Что такое тестирование, кто такой тестировщик. Зачем нужно тестирование и когда оно начинается

1.3 Этапы ручного тестирования веб-приложений

1.4 Источники требований: что такое техническое задание и как проводить ручное тестирование, если техническое задание отсутствует

Глава 2. Подготовка к ручному тестированию веб-приложения (~15 стр)

2.1 Составление чек-листов

2.2 Составление тест-кейсов

2.3 Разработка тест-плана

2.4 Тест-дизайн и тест-аналитика

2.5 Баг-репорты и багтрекинговая система

Глава 3. Проведение тестирования и разработка предложений по улучшению тестирования Интернет-магазина РИВ ГОШ (~20 стр.)

3.1 Тестирование функциональности

3.2 Тестирование юзабилити

3.3 Тестирование интерфейса

3.4 Тестирование совместимости

3.5 Тестирование производительности

3.6 Тестирование безопасности

3.7 Подведение итогов. Разработка предложений по улучшению ручного тестирования веб-приложения РИВ ГОШ

Заключение (~ 4 стр.)

Список используемой литературы

Приложения

# 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:** Postman, Git, VsCode, Docker, React, Laravel.

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

# Теоретическая часть

## 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.

## React: библиотека для создания пользовательских интерфейсов

### 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.

### Основные концепции и принципы React

#### 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:

* 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.

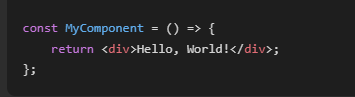


Рисунок 1 – Функциональный компонент

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

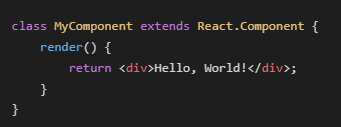


Рисунок 2 – Классовый компонент

#### 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 {}:

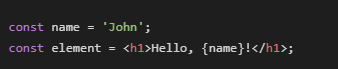


Рисунок 3 – Динамические данные в JSX

#### 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.



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.

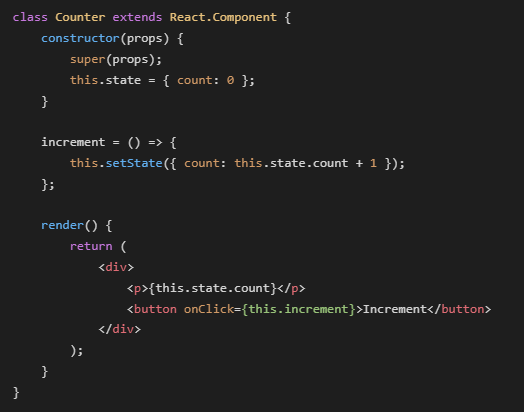


Рисунок 5 – Состояние state

#### 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.

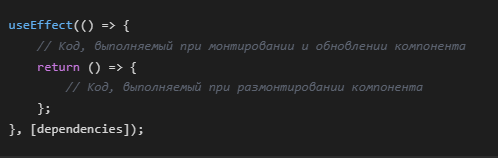


Рисунок 6 – Hook UseEffect

### 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.

### Заключение

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.

## Laravel: PHP-фреймворк для бэкенд-разработки

### 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 [1].

### Основные концепции и принципы Laravel

#### Архитектура MVC

Архитектура MVC разделяет приложение на три основных компонента:

* 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.

#### 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 7:

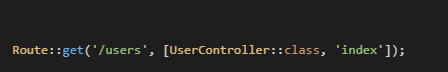


Рисунок 7 – Пример определения маршрута

#### 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 8:



Рисунок 8 – 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.

#### 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 9:



Рисунок 9 – Пример создания миграции

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



Рисунок 10 – Пример создания seeder

#### 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).

Пример инъекции зависимостей в контроллере на рисунке 11:

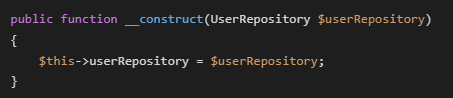


Рисунок 11 – Пример инъекции зависимости

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

### 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.

### Заключение

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.

## 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.

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.

## Основные функции приложения

Веб-приложение будет включать в себя следующие ключевые функции:

Поиск мероприятий: пользователи будут получать выдачу мероприятий по одному из предложенных городов. При этом пользователь, который имеет аккаунт и произвел вход в личный кабинет, будет иметь возможность добавлять событие в избранное и просматривать список событий в своём избранном.

## Заключение

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.

# Практическая реализация

## 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 (cм. Приложение 1)
* Запуск docker-среды
* Создание нового проекта Laravel. Для нового проекта используем следующую команду:



Рисунок 12 – Создание Laravel приложения

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

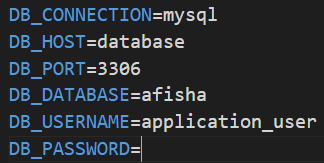


Рисунок 13 – Настройки подключения к Базе данных

* Запуск миграций:



Рисунок 14 – 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.

## Подключение React к Laravel приложению

Для подключения библиотеки React к Laravel приложению была установлена библиотека Laravel Breeze [2].

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:



Рисунок 15 – Установка Laravel Breeze

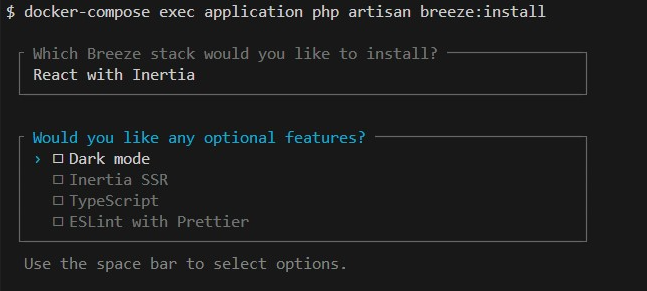


Рисунок 16 – Установка Laravel Breeze в приложение с выбором React + Inertia

Затем можно перейти по адресу /login или /register приложения в веб-браузере. Все маршруты Breeze определены в файле routes/auth.php [3].

## Функциональность приложения

### Общая функциональность

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

Приложение берет информацию с публичного API KudaGo [6] 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/> [5]. 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.

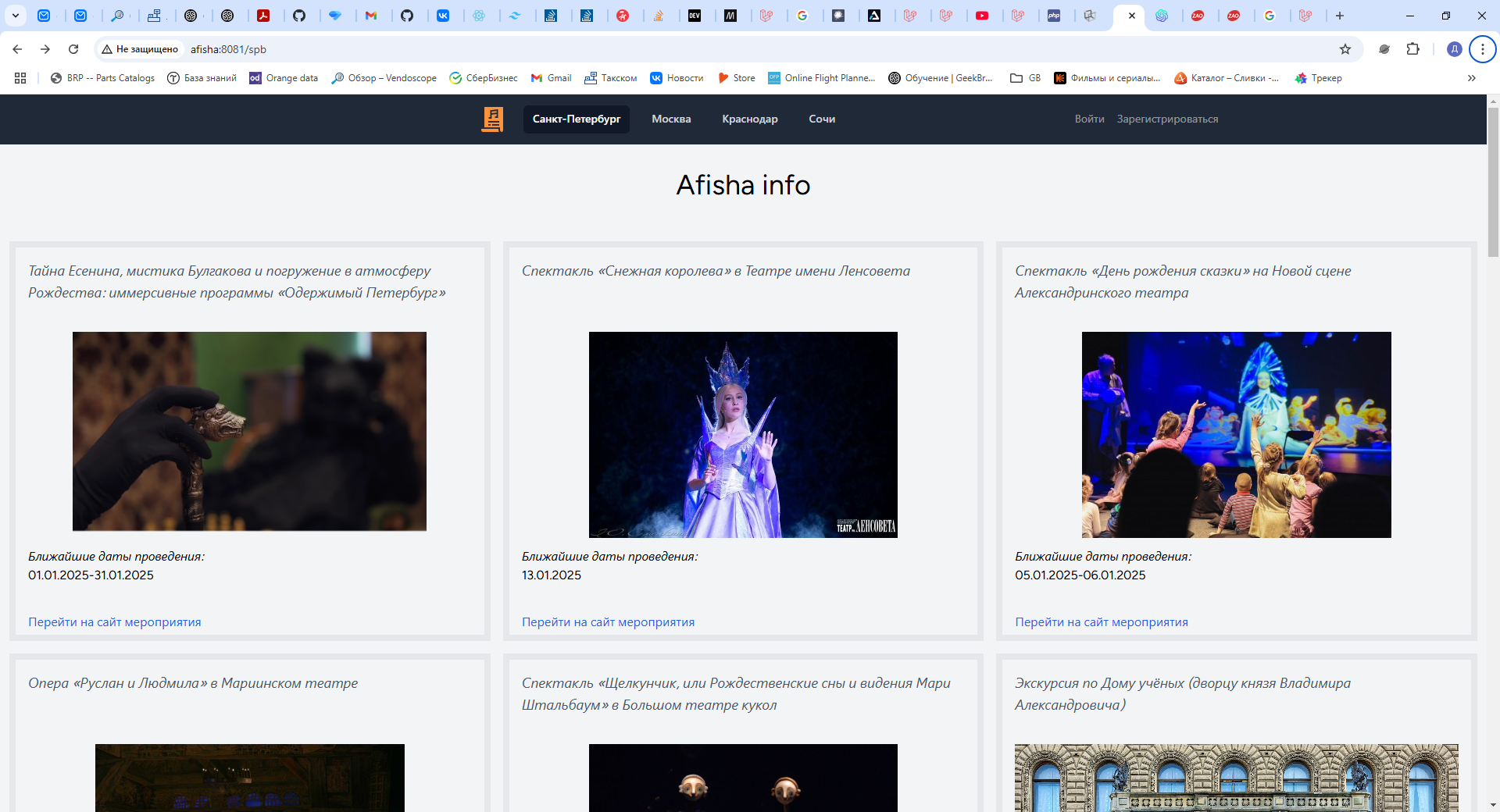


Рисунок 17 – Общий вид веб-приложения

### Функциональность авторизации

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



Рисунок 18 – Кнопки авторизации

При переходе на страницу создания учетной записи, пользователь попадает по адресу /register и наблюдает перед собой окно, представленное на рисунке 19.

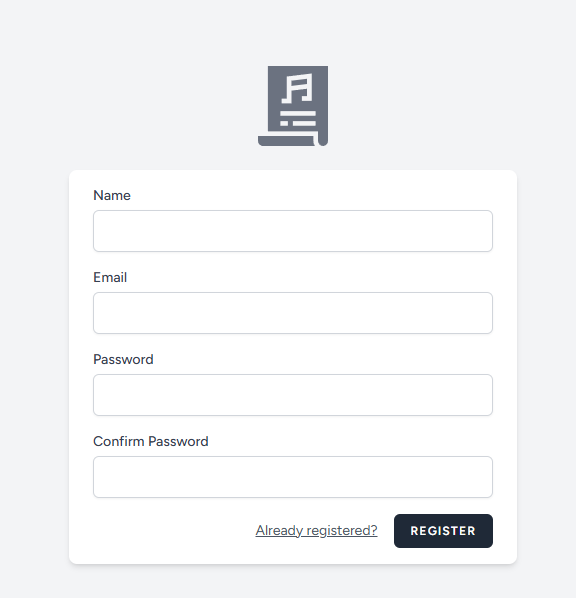


Рисунок 19 – Окно создания учетной записи

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 (cм. Рисунок 20). Также пользователю придет сообщение на email, с благодарностью за регистрацию (cм. Рисунок 21).

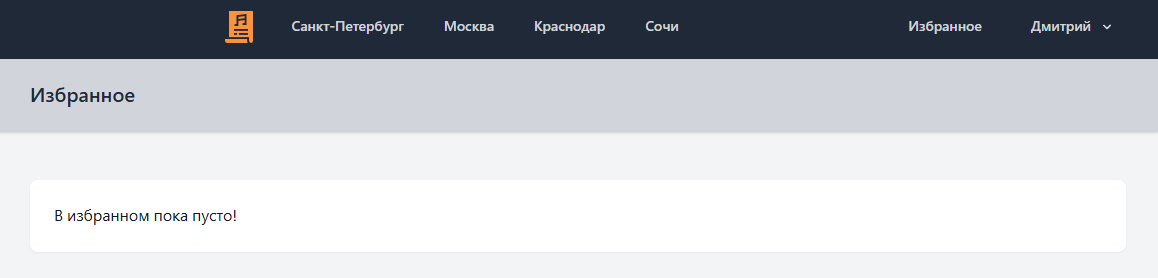


Рисунок 20 – Окно избранного зарегистрировавшегося пользователя

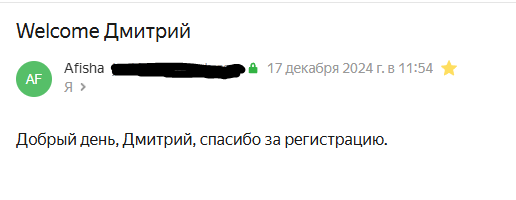


Рисунок 21 – Присланное письмо

При переходе на страницу авторизации, пользователь попадает по адресу /login и наблюдает перед собой окно, представленное на рисунке 22.

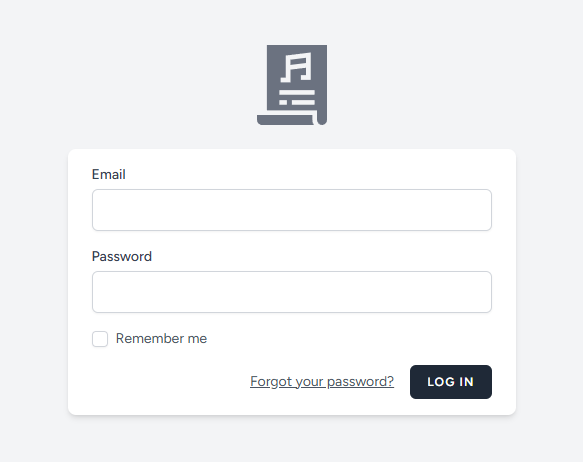


Рисунок 22 – Окно авторизации

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.

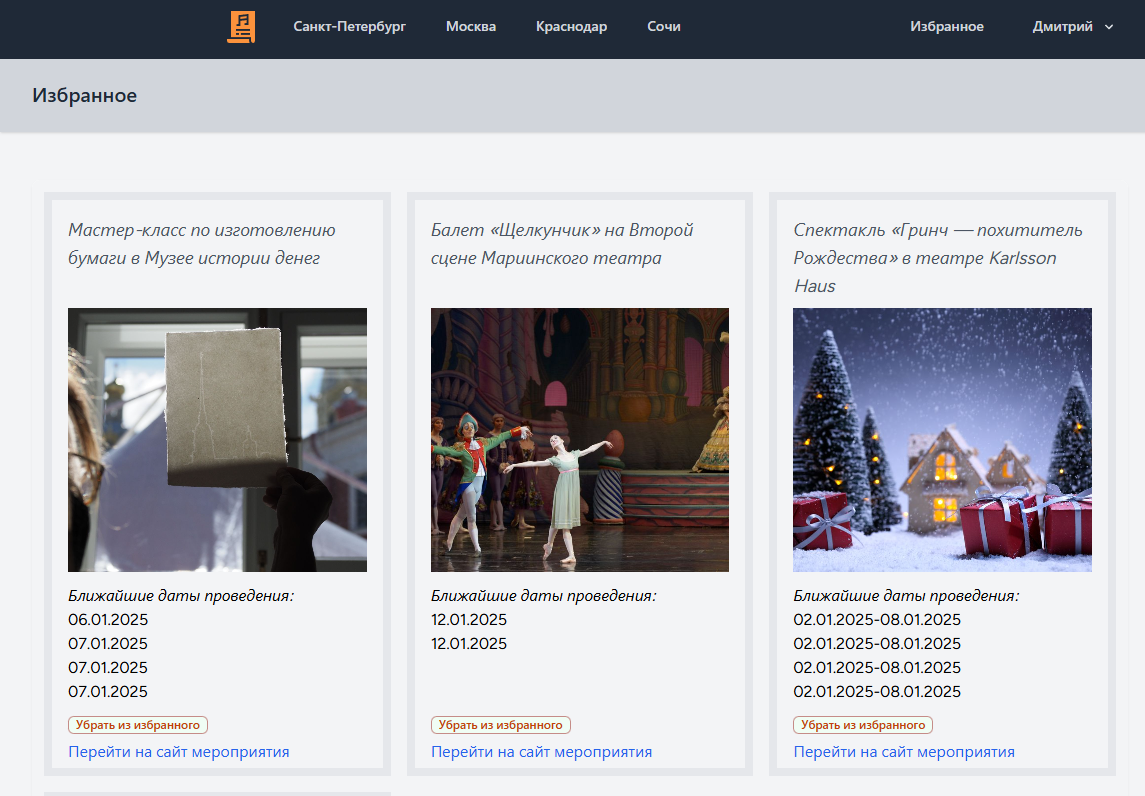


Рисунок 23 – Избранное авторизованного пользователя

Авторизованный пользователь, при просмотре событий в выбранном городе имеет возможность добавить понравившееся событие в избранное.

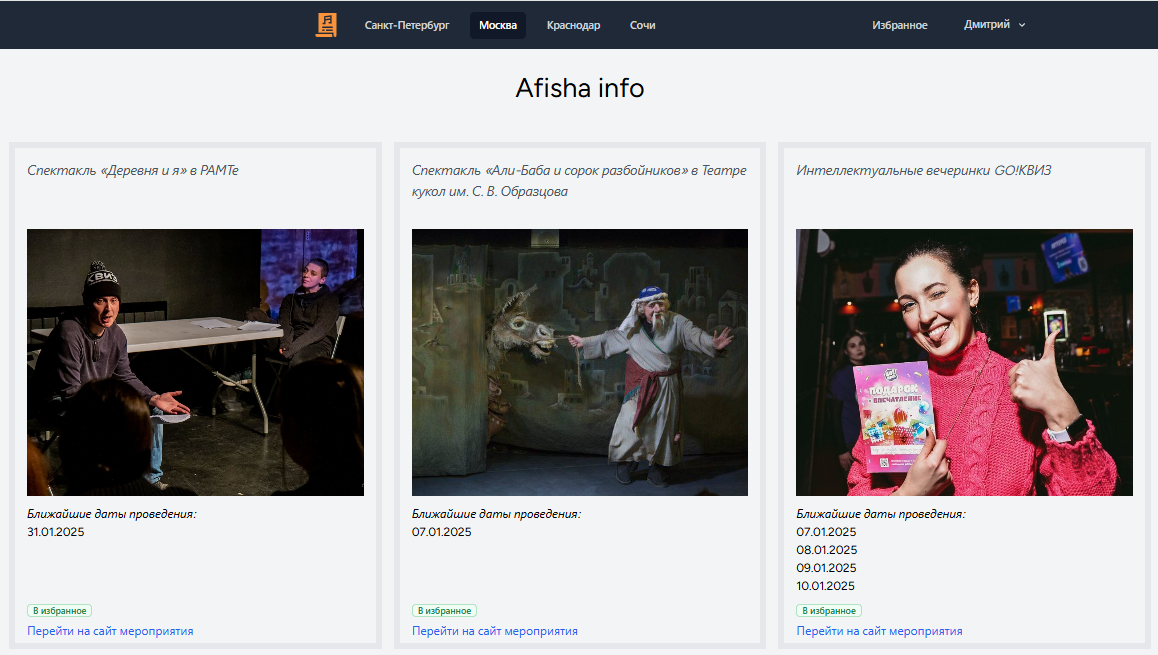


Рисунок 24 – Возможность добавления в избранное

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

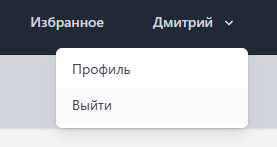


Рисунок 25 – Выпадающее меню профиля

При нажатии на кнопку “Профиль”, пользователь будет перенаправлен по адресу /profile на страницу редактирования информации, представленной на рисунке 26.

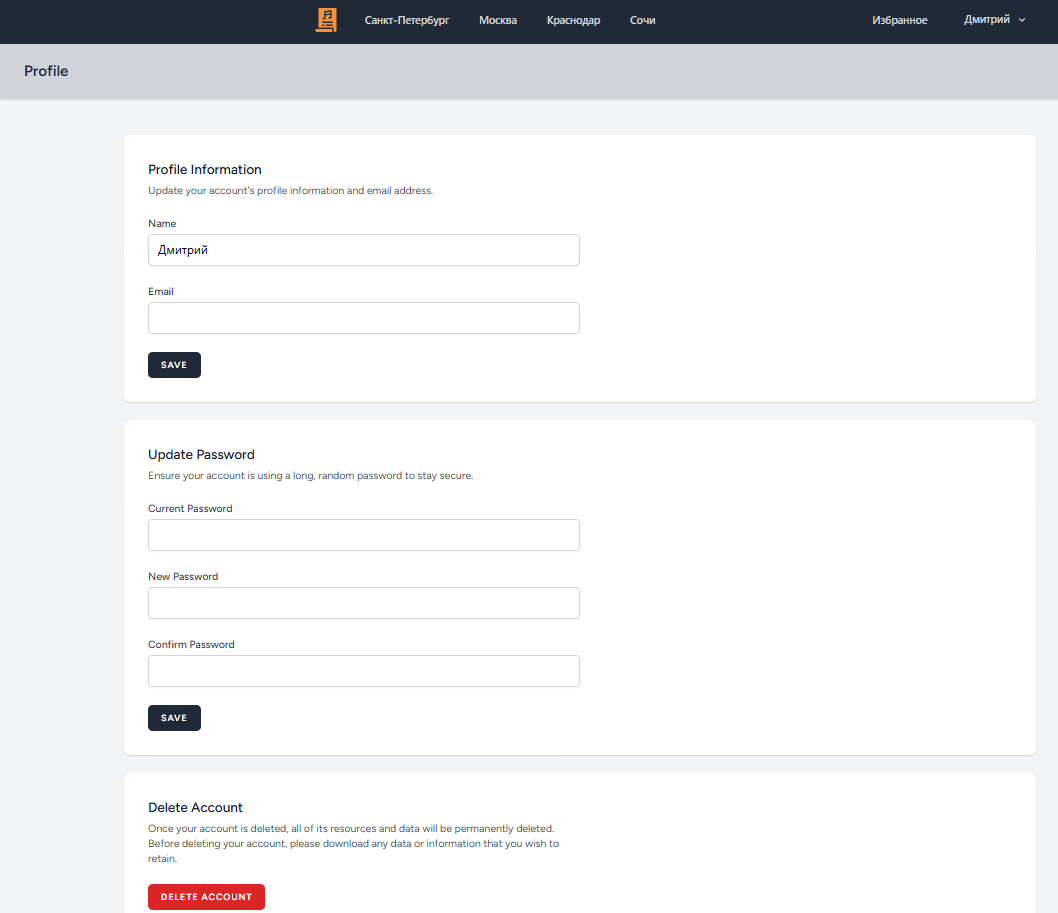


Рисунок 26 – Страница редактирования профиля

### 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 [4].

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. На рисунке 27 представлена выборка из базы данных из таблицы users.

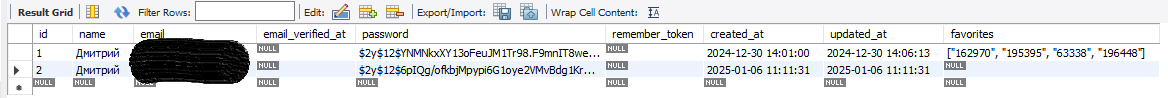


Рисунок 27 – Выборка из таблицы users

Как можно видеть из представленного рисунка 27, 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 таблицой 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.

# Заключение

В заключение необходимо включить следующее:

1. Краткие и ёмкие теоретические и практические выводы, которые были получены во время анализа теоретической базы и практического исследования.
2. Оценка проведённого исследования, описание его результатов.
3. Практическая значимость работы, рекомендации и планы на дальнейшие исследования.
4. Общий итог — достижение цели, выполнение задач, доказательство гипотезы.
5. Предложения по совершенствованию объекта исследования.

# Список используемой литературы

1. Официальная документация фреймворка Laravel: [Электронный ресурс] — URL: https://laravel.com/docs/11.x/ (дата обращения: 6.11.2024).
2. 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).
3. 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).
4. 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).
5. Сайт по поиску мероприятий в городах РФ: [Электронный ресурс] — URL: https://kudago.com/ (дата обращения: 16.11.2024).
6. 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).

Здесь нужно будет указатьсписок используемой литературы, ссылки на все ресурсы, которые нужны были для создания проектной работы.

Основные правила оформления использованной литературы и ресурсов:

1. Каждый источник упоминается единожды, независимо от того, насколько часто на него ссылаются.
2. Список литературы оформляется в алфавитном порядке по фамилии автора, сначала русскоязычная литература, затем иностранная, далее интернет-сайты.
3. Библиографическая запись обязательно включает:
   * Фамилию автора или фамилии их группы, инициалы (при наличии).
   * Название статьи, книги, справочника, закона, иного документа.
   * Населённый пункт, в котором был издан источник, наименование издательства.
   * Год публикации.
   * Число страниц.

## Пример

* *Книга: Автор. Название книги. Город: Издательство, Год.*
* *Статья: Автор. "Заголовок статьи." Название журнала Том, номер (Год): страницы.*

# Приложения

## 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

В **приложения** обычно входят артефакты, получившиеся в процессе создания проекта:

1. Объёмные графики и таблицы, которые не помещаются на лист А4.
2. Длинные математические формулы и расчёты по ним.
3. Характеристики аппаратуры, которая использовалась для проведения исследования.
4. Авторские методики.
5. Вспомогательный материал: тесты, карточки, схемы, рисунки.
6. Материалы, полученные на предприятии: отчёты, прочие документы.